



HAL
open science

Distributed clocking for synchronous SoCs

Eldar Zianbetov

► **To cite this version:**

Eldar Zianbetov. Distributed clocking for synchronous SoCs. Micro and nanotechnologies/Microelectronics. Université Pierre et Marie Curie - Paris VI, 2013. English. NNT: . tel-01053729

HAL Id: tel-01053729

<https://theses.hal.science/tel-01053729v1>

Submitted on 1 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



THÈSE DE DOCTORAT DE
L'UNIVERSITÉ PIERRE ET MARIE CURIE

École Doctorale Informatique, Télécommunications et Électronique
(EDITE)

Présentée par :
Eldar ZIANBETOV

Pour obtenir le grade de :
DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

HORLOGERIE DISTRIBUÉE POUR LES SOC_s SYNCHRONES

Présentée le :
25.03.2013

Le jury est composé de :

M. Jean-Baptiste BEGUERET	IMS Bordeaux	Rapporteur
M. David NAIRN	Université de Waterloo	Rapporteur
M. Jean-Pierre SCHOELLKOPF	SiLiCieL	Rapporteur
M. Bernard COURTOIS	CNRS	Examineur
M. Alain GREINER	UPMC, LIP6	Examineur
M. Valeriy SITNIKOV	Odessa State Polytechnic University	Examineur
M. François ANCEAU	CNAM	Directeur de Thèse
M. Dimitri GALAYKO	UPMC, LIP6	Co-directeur de Thèse
M. Éric COLINET	CEA-LETI, Minatec	Invité
M. Jérôme JUILLARD	Supélec	Invité



DOCTORAL DISSERTATION
PIERRE AND MARIE CURIE UNIVERSITY

Doctoral School of Informatics, Telecommunications and Electronics
(EDITE)

Presented by:
Eldar ZIANBETOV

To obtain the degree of :
DOCTOR OF PHILOSOPHY AT UNIVERSITY OF PIERRE AND
MARIE CURIE

Thesis title :

DISTRIBUTED CLOCKING FOR SYNCHRONOUS SOC

Presented on :

25.03.2013

Members of jury :

M. Jean-Baptiste BEGUERET	IMS Bordeaux	Reviewer
M. David NAIRN	University of Waterloo	Reviewer
M. Jean-Pierre SCHOELLKOPF	SiLiCieL	Reviewer
M. Bernard COURTOIS	CNRS	Examinator
M. Alain GREINER	UPMC, LIP6	Examinator
M. Valeriy SITNIKOV	Odessa State Polytechnic University	Examinator
M. François ANCEAU	CNAM	Supervisor
M. Dimitri GALAYKO	UPMC, LIP6	Co-Supervisor
M. Éric COLINET	CEA-LETI, Minatec	Invited
M. Jérôme JUILLARD	Supélec	Invited

Abstract

This dissertation addresses the problem of global synchronization of complex SoC in the context of deeply submicron CMOS technologies.

Nowadays, to circumvent the difficulties associated with conventional clock distribution techniques (e.g. tree, mesh) in synchronous systems, the designers wishing to go on with the Globally Synchronous paradigm are turning toward clocking techniques breaking away from conventional approaches (e.g. distributed oscillators, stationary waves, coupled oscillators, programmable delays). This study is situated on this research axis.

In this research we studied and elaborated a global distributed clocking system for a highly reliable synchronous circuit. This clocking scheme is based on a network of oscillators coupled in phase. Inside each synchronous clocking domain, there is one oscillator that generates the local clock. To synchronize the oscillators (i.e. domains), each one of them is controlled by an All-Digital Phase Locked Loop (ADPLL), realizing a phase coupling between the oscillators of neighboring zones.

During this research we have developed the specifications and selected an architecture of the network. A theoretical model of the system has been established in a collaboration with CEA-LETI and Supélec laboratories in the framework of ANR HODISS project. We have analyzed the behavior of the system in simulations on different abstraction levels, investigated the stability conditions of its synchronous operation.

An All-Digital Phase Locked Loop (ADPLL) has been proposed for the role of an elementary node of distributed clocking network. The use of ADPLL permits to circumvent difficulties of implementation, which are usually associated with analog PLL. We have designed the main blocks of the ADPLL: a Digitally-Controlled Oscillator (DCO), a Phase-Frequency Detector (PFD) and an error processing block. A cell-based design technique has been adapted for the design of DCO layout. This technique significantly reduced the complexity of the oscillator's implementation. The remaining blocks have been designed in a common digital design flow.

In order to reduce the risks associated with silicon implementation, the system has been validated in a FPGA prototyping platform. The results of the measurements showed that clocking network behaves as predicted by the theory and simulations.

Two prototype circuits have been designed, implemented and tested in a 65 nm STMicroelectronics CMOS technology. The first one is a proof of concept of a designed highly linear and monotonous DCO. The measured parameters of oscillator showed the compliance with specifications. The measured performance demonstrated the <15 ps rms jitter, while consuming 6.2 mW/GHz with 1.1 V supply voltage. The tuning range of the oscillator is 999-2480 MHz under 10 bit resolution. The second chip is a 4×4 node clocking network which consists of 16 distributed ADPLLs. Each of them employs a designed earlier DCO, PFD and error processing block. The experiments showed that proposed technique of distributed clock generation is feasible in a real CMOS chip environment. The measured performance demonstrated the timing error between neighbor oscillators less than 60 ps, while power consumption is 98.47 mW/GHz.

Thesis title: Distributed clocking for synchronous SoC

Key words: synchronous clocking, multioscillator architecture, all-digital phase locked loop, digitally-controlled oscillator, bang-bang detector, time-to-digital converter

Thesis Supervisor: François ANCEAU, Professor at Conservatoire national des arts et métiers

Thesis Co-Supervisor: Dimitri GALAYKO, Associate Professor at Pierre and Marie Curie University (Paris VI)

In Memory Of My Mother, Who Passed Away In 1994

The meaning of this degree is that the recipient of instruction is examined for the last time in his life, and is pronounced completely full.

Stephen Leacock. McGill University, 1912.

Acknowledgements

With a great sincerity I wish to express my deep gratitude to all those who helped me to carry out my doctoral work. I have been surrounded by family, friends, project partners, labmates, colleagues and professors who have provided support and aid during this research.

This thesis would not have been possible without the guidance, patience and generosity of my supervisor Dimitry Galayko. I am grateful to him for giving me the opportunity to do the research in such a marvelous city as Paris. I would like to deeply thank for his continuous support, encouragement and help. I am also very grateful to my supervisor François Anceau for his ambitions and inspiring enthusiasm. During these years he has been a source of ideas, knowledge and motivation for me. It has been a honor working and collaborating with them.

I want to express my thanks to our collaborators Éric Colinet, Jérôme Juillard and Sylvain Féruglio. A special thanks to Mehdi Terosiet, Anton Korniienko and Jean-Michel Akre for precious discussions and of course for their time spent on explaining me things out of my competence.

I would like also to thank Valeriy Sitnikov for giving me an initial burst of enthusiasm on this encouraging and challenging way of research.

A big thank for all my labmates as well. They created a warm and friendly atmosphere in our team and the excellent research environment in the CIAN group. A special thank to Diomadson Belfort, who helped me a lot with CAD tools, simulations, design of layout and showed many special "magic" tricks – they saved countless hours of my efforts. Another special thank to Mohammad Javidan for his valuable help in carrying out the design of the second chip.

The specialists in CEA-LETI and CMP center helped to make the tape-out processes as easy as it was possible. In particular, the hours spent on verifying my designs with Kholdoun Torki and Olivier Billoint was very useful.

I thank my wife, Maria, for moral support and encouragement in my private life all these years. Thank you for organizing all our vacations.

Finally, my thanks go to all those who participated more or less indirectly to the success of my thesis.

Eldar Zianbetov

Contents

1	Introduction	1
1.1	Area of focus	1
1.1.1	Clocking in large digital circuits	2
1.1.2	Clock error issues	4
1.2	State of the art: synchronous clocking in modern SoC	6
1.2.1	Conventional clock trees	6
1.2.2	Optical distribution technique	8
1.2.3	Multioscillator architectures	9
1.2.4	Synchronous clocking architectures: conclusion	12
1.3	Thesis outline and contribution	14
2	Network of distributed ADPLLs	15
2.1	Introduction	15
2.2	Proposed clocking architecture	16
2.2.1	Network of coupled PLLs	16
2.2.2	Digital phase synthesis	18
2.2.3	Blocks of the ADPLL network	19
2.2.4	Specification of clocking network parameters	22
2.3	Multiplicity of synchronisation modes	25
2.3.1	Definition of the problem	25
2.3.2	Synchronization mode selection	26
2.4	Stability of the PLL networks	29
2.5	Modeling of the clocking network	30
2.5.1	The model of the phase comparator	30
2.5.2	Loop filter	31
2.5.3	DCO	31
2.5.4	Simulation of ADPLL	31
2.5.5	Simulation of network	32
2.6	Conclusion	38
3	Digitally controlled oscillator design	39
3.1	Introduction	39
3.2	Digital frequency tuning in ring oscillators	42

3.2.1	Capacitive tuning	43
3.2.2	Current/Voltage tuning	44
3.2.3	Current tuning with width modulation technique	44
3.2.4	Frequency tuning in 2D array oscillator	46
3.2.5	The choice of the coding	48
3.3	DCO architecture	52
3.3.1	Oscillator topology	52
3.3.2	Control algorithm	53
3.4	Sizing of the DCO core and tuning cells	57
3.5	VHDL modeling of the oscillator	58
3.5.1	Precise modeling of the real code-frequency characteristic	58
3.5.2	Synthesis of the DCO output frequency in the precise VHDL model	59
3.6	DCO layout desing I: the DCO floorplan	61
3.6.1	Cell based design	61
3.6.2	Power planning	62
3.6.3	Signal flow oriented layout	63
3.6.4	Guard rings	64
3.6.5	General DCO floorplan	64
3.7	DCO layout design II: cell design	65
3.7.1	Main inverters	65
3.7.2	Feedback wire and decoupling capacitors	66
3.7.3	Tuning inverters	67
3.7.4	DCO output interface	71
3.7.5	A, B and C bus generator	72
3.7.6	DCO chip	73
3.8	DCO chip test	76
3.8.1	Impact of the supply variations	76
3.8.2	Chip-to-chip variations	76
3.8.3	Power consumption	77
3.8.4	Linearity	78
3.8.5	Jitter characteristics	78
3.9	Conclusion	80
4	Digital blocks of the ADPLL	81
4.1	Introduction	81
4.2	Digital phase/frequency error measurement	82
4.2.1	Digital versus analog phase comparators	82
4.2.2	Phase comparators versus phase-frequency detectors	83
4.2.3	The digital PFD architecture	85
4.2.4	Metastability problem	87
4.3	Implementation of digital PFD	89
4.3.1	Bang-bang detector implementation	89

4.3.2	Time-to-digital converter	94
4.3.3	Implementation of PFD	96
4.4	Digital loop control of ADPLL network node	99
4.4.1	Error combining block	99
4.4.2	PI filter	100
4.4.3	Implementation	101
4.4.4	ADPLL simulation results	101
4.5	Node programming mechanism	105
4.6	Conclusion	107
5	Clock network implementation	109
5.1	Introduction	109
5.2	FPGA prototyping	111
5.2.1	Synthesizable DCO	112
5.2.2	FPGA based TDC	114
5.2.3	Experimental results	115
5.2.4	FPGA prototyping: conclusion	120
5.3	Silicon implementation of the clock network	122
5.3.1	Floorplan of the test chip	122
5.3.2	Design for test	124
5.3.3	Test chip layout	127
5.4	Measurement results	130
5.4.1	Initial frequencies of DCOs	130
5.4.2	Supply voltage sensivity	131
5.4.3	Power consumption	131
5.4.4	Bidirectional configuration	132
5.4.5	Unidirectional configuration	133
5.4.6	Corner to corner timing errors	135
5.4.7	Study of mode-locking phenomenon	136
5.4.8	Coefficient variation	137
5.5	Conclusion	141
6	Conclusion and Perspectives	143
6.1	Thesis summary and conclusions	143
6.2	Future work	145
	Appendices	147
A	VHDL models of the ADPLL blocks	149
B	Phase error sign detection theorem proof	165
C	Matlab scripts	167

D DCO test chip characterization flow	175
E FPGA prototyping of the clocking network	179
F VLSI implementation of the network	185
G Clocking network test chip characterization flow	187
Bibliography	195

List of Figures

1.1	Amount of transistors in microprocessors and SoC	2
1.2	Frequency of the Intel microprocessors	2
1.3	Clock domains in a SoC	3
1.4	Flow of the data in clocked system and its timing digram	4
1.5	Definition of the skew and jitter	5
1.6	Examples of conventional clock distribution tree structures	6
1.7	Typical mesh clock network	7
1.8	Centralized skew compensation technique	8
1.9	Example of decentralized skew compensation technique	9
1.10	An optical clock distribution network	9
1.11	Network of oscillators coupled by voltage	10
1.12	Standing-wave clocking principle	11
1.13	Distributed PLL	11
1.14	Network node	12
1.15	16-node distributed clocking network	12
2.1	Basic idea of multioscillator clocking approach	16
2.2	Topology of the proposed clock network and architecture of the network node	17
2.3	Phase coupling between two oscillators	17
2.4	Block diagram of the ADPLL	18
2.5	Block diagram of the PI filter	21
2.6	Transfer function of the digital phase comparator	22
2.7	Definition of two main design constraints for the clocking network	23
2.8	Frequency tuning limits consideration	24
2.9	Cyclic nature of the conventional analog linear phase comparator	25
2.10	Illustration of the mode-locking phenomenon in a 2×2 mesh network	26
2.11	Transfer function of the phase comparator to avoid undesirable states	26
2.12	Propagation of the phase information in unidirectional network	27
2.13	Dynamic reconfiguration of the network from uni- to bidirectional	28
2.14	Representation of the PLL network for stability study in [59]	29
2.15	Block diagram of the modeled phase comparator	30
2.16	Principle of the VHDL implementation of digital period-controlled oscillator	31
2.17	Simulation of the developed VHDL model of ADPLL	32
2.18	Simulated 16-node clocking network	33

2.19	Simulation of the developed VHDL model of 16-node network	35
2.20	Errors of the PFDs connected to the node 11 in first experiment	35
2.21	Simulation of the developed VHDL model of 16-node network	36
2.22	Simulation results of dynamically reconfigurable network	37
3.1	Representation of DCO as a combination of DAC and VCO	40
3.2	Inverter-based ring oscillator	42
3.3	CMOS inverter with parasitic components	43
3.4	Charge and discharge of parasitic capacitance in CMOS inverter	43
3.5	Capacitive tuning in a ring oscillator	44
3.6	Current tuning in a ring oscillator	44
3.7	Current tuning technique with direct digital control	45
3.8	Digital delay control principle	46
3.9	Frequency tuning in 2D array oscillator	47
3.10	Thermometer current coding	49
3.11	Binary current coding	50
3.12	Core of the proposed oscillator	52
3.13	Virtual extension of number of stages	54
3.14	Oscillator control grid	55
3.15	Approximation of the code-frequency characteristic in VHDL model of DCO	58
3.16	Structure of the implemented VHDL model	59
3.17	Period-controlled oscillator with jitter	59
3.18	Simulation of the DCO behavioral model	60
3.19	Interdigitated multi-finger power routing	61
3.20	Cell layout template	62
3.21	Supply decoupling polysilicon CMOS capacitor	63
3.22	Signal flow driven placement of the tuning cells in DCO	64
3.23	Guard rings around oscillator core	64
3.24	Floorplan of the designed oscillator	65
3.25	Schematic of a main inverter of oscillator	66
3.26	Layout of the feedback line/decoupling capacitance service cell	67
3.27	Feedback/decoupling capacitor cell in the power distribution network	67
3.28	Single tuning cell layout template	68
3.29	Schematic diagram of the coarse tuning cells	68
3.30	Layout of the coarse tuning cells	69
3.31	Schematic diagram of the additional coarse tuning cells	69
3.32	Additional coarse tuning cells layout	70
3.33	Schematic diagram of the fine tuning cells	70
3.34	Layout of the FTI cell	71
3.35	Improvement of duty cycle by divider	71
3.36	Schematic diagram of the feedback frequency divider	71
3.37	Layout of the block containing the main inverters of oscillator, dividers and buffers	72

3.38	Schematic of the <i>A</i> , <i>B</i> and <i>C</i> signal generation circuit	73
3.39	Layout of the B2T decoders	73
3.40	Schematic diagram of the DCO test chip	74
3.41	Microphotograph of the fabricated chip	75
3.42	Measured and simulated output frequency versus input control code for different supply voltages	76
3.43	Measured tuning curves for 7 samples of fabricated circuits	77
3.44	Measured and simulated power consumption versus input control code	77
3.45	Measured differential nonlinearity of the circuit	78
3.46	Measured jitter versus control code	78
4.1	XOR phase comparator	82
4.2	Conventional phase comparator	83
4.3	The phase/frequency detector	84
4.4	Two cases of initial conditions in phase/frequency detector	85
4.5	Detection of the phase error sign variation	85
4.6	Proposed phase/frequency detector for clock network	86
4.7	Principle of operation of proposed PFD	86
4.8	Simplest trigger circuit	87
4.9	Proposed in [67] arbiter circuit	87
4.10	Schematic diagram of the bang-bang phase/frequency detector	89
4.11	Circuit diagram of the reduced flip-flop	91
4.12	Circuit diagram of the metastability filter with RS-trigger	91
4.13	Circuit diagram of the Muller C-element	91
4.14	Delay of the arbiter as a function of timing error	92
4.15	Degradation of the bang-bang loop delay with smaller input errors	93
4.16	Layout of the bang-bang detector	93
4.17	Time-to-digital converter	95
4.18	Block diagram of proposed time-to-digital converter	95
4.19	Simulated transfer function of the designed flash time-to-digital converter	96
4.20	Layout of the TDC employing standard cells	97
4.21	Simulation of the designed multi-bit PFD	98
4.22	Signal flow diagram for proposed PFD	98
4.23	Layout of the proposed PFD	98
4.24	Error signal processing block	99
4.25	Layout of the digital signal processing block	102
4.26	Layout of the test ADPLL	103
4.27	Simulation of the designed ADPLL with proposed oscillator, PFD and signal processing block	104
4.28	Schematic of the programming interface	106
4.29	Cascading the programming interfaces of several blocks	106

5.1	Structure of the implemented clock network	110
5.2	Repeating discrete ramp function in the DDFS	112
5.3	Schematic diagram of the proposed FPGA implementation of the oscillator	113
5.4	Conventional phase detector	115
5.5	Block diagram of the node in a FPGA prototype with observation points	116
5.6	Local clock signals together with reference	117
5.7	Local clock signals together with reference	118
5.8	Local clock signals around Node 10 together with reference and integer sum of the errors	118
5.9	Local clock signals together with reference	119
5.10	Local clock signals around Node 10 together with reference and integer sum of the errors	119
5.11	Local clock signals together with reference	120
5.12	Local clock signals around Node 10 together with reference and integer sum of the errors	120
5.13	Preliminary floorplan of the test chip	123
5.14	Power supply scheme of the test chip	123
5.15	Simulation of the extracted from layout supply distribution network	124
5.16	The connection sequence of the programmable blocks of the network	125
5.17	Block diagram of the node in a ASIC implementation	126
5.18	Multichannel multiplexer with serial interface for the PFD output commutation	126
5.19	Layout of the test chip of the clock network	128
5.20	Microphotograph of the fabricated clocking network test circuit	129
5.21	Measured initial frequencies of the DCOs	131
5.22	Measured supply voltage sensivity of the DCOs	132
5.23	Measured current consumption of the network	132
5.24	Histogram of the reference clock	133
5.25	Zones of the network for the measurement	133
5.26	Synchronous clocks in the bidirectional configuration	134
5.27	Outputs of the PFDs from the Zone 9 in bidirectional configuration	134
5.28	Outputs of the PFDs from the Zone 9 in unidirectional configuration	135
5.29	Timing error between corner nodes	136
5.30	Configuration of the network for mode-locking study	137
5.31	Misaligned clocks in the mode-locking state	137
5.32	Outputs of the PFDs in the mode-locking state	138
5.33	Transient process in the Node 11 depending on variation of the coefficient K_p and K_i	139
B.1	Sign detection in BB circuit	165
D.1	Schematic of the developed PCB	177
D.2	Photo of the developed testing platform	178

E.1	Functional diagram of Cyclone II DSP Development Board	183
E.2	Top view of Cyclone II DSP Development Board	183
E.3	FPGA prototyping platform	184
F.1	Complete block diagram of the clock network and DFT blocks	186
G.1	Schematic of the developed PCB for network chip testing	189
G.2	Photo of the developed clocking network testing platform	190

List of Tables

2.1	Clocking network implementation specification	24
2.2	ADPLL simulation parameters and conditions summary	32
2.3	Clocking network simulation parameters and conditions summary	34
3.1	Dimensions of the transistors in the DCO cells	57
3.2	Layers functionality attribution	63
3.3	DCO chip performance summary	79
4.1	C-element truth table	90
4.2	Simulation parameters and conditions	102
5.1	Parameters of the FPGA and VLSI implementations of the DCO	116
5.2	Clock network test chip measurement conditions	130
5.3	Network test chip measurements summary	140

Chapter 1

Introduction

Contents

1.1	Area of focus	1
1.2	State of the art: synchronous clocking in modern SoC	6
1.3	Thesis outline and contribution	14

1.1 Area of focus

Nowadays, numerous scientific, industrial and multimedia tasks are performed by electronic devices based on Systems-On-Chip (SoC). Over the past decade, SoCs evolved from systems with thousands transistors performing simple information processing toward high-performance multiprocessor systems able to perform billions of operations per second. Till recently, the growth of the SoC performance has been obtained through an *intensive* strategy: reducing the transistor size, increasing the number of gates per chip (Fig. 1.1), increasing the clock frequency. Whereas the chip density increases following the Moore law, the clock frequency evolution has stalled since several years at the level of 500-4000 MHz (Fig. 1.2). Further increase of the processing power of chips is achieved through association of several parallel processing units on chips, and by optimizing their interaction.

The saturation of the clock frequency growth is explained by the difficulties of a global clock distribution over a large chip designed in deeply submicron CMOS technologies, and by a high energy cost of such distribution. Indeed, for gigahertz clock frequencies, a linear increase of the clock frequency* implies an exponential increase of the consumed power. This goes against the requirements of low-power autonomous systems which are the dominant applications on the nowadays market [50]. For those reasons, the SoC designers limit the clock frequencies to few hundred megahertz, and look toward SoC architectures which do not require a global clock generation: e.g. GALS(Globally Asynchronous Locally Synchronous). However, although clockless asynchronous approaches are gaining ground [25],

*Usually in combination with increase of supply voltage

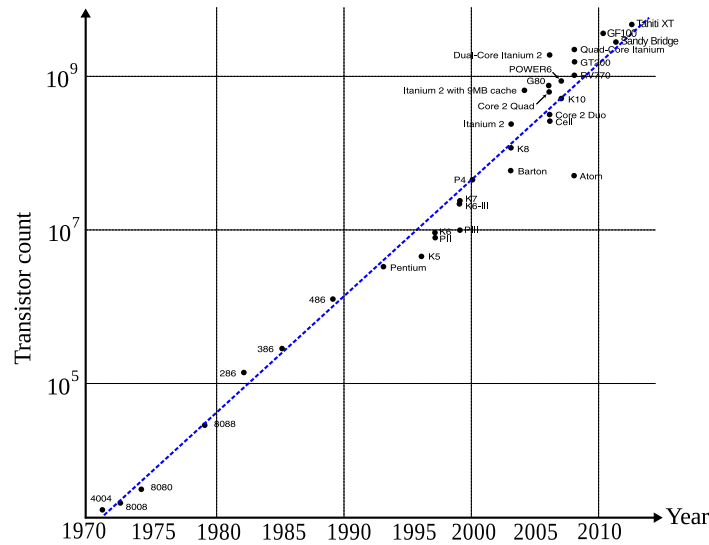


Figure 1.1: Amount of transistors in microprocessors and SoC of various companies over past 30 years

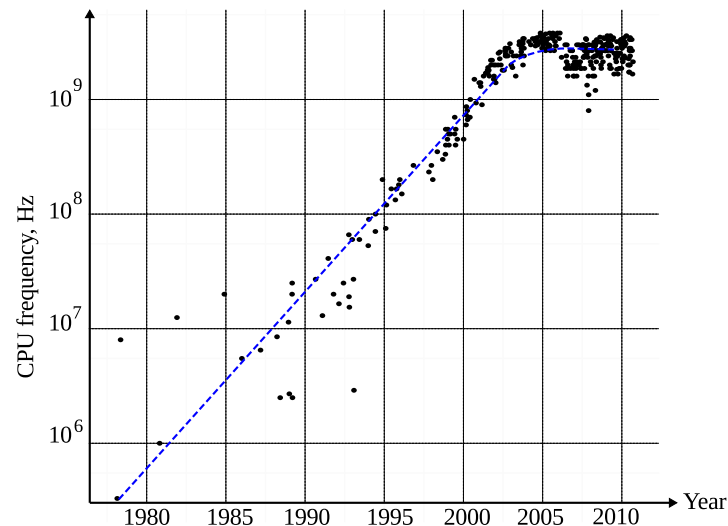


Figure 1.2: Frequency of the Intel microprocessors over past 30 years

synchronous approach remains the preferred on-chip communication technique, mainly because of the better mastering and prediction of synchronous circuits behavior [21].

This study addresses the problem of global clock generation inside complex and large SoC: e.g. MPSoC (Multi Processor SoC). It is motivated by deficiencies of conventional clock generation approaches such as a clock grid or a tree in the context of deeply sub-micron CMOS technologies. The developed clock generation technique is based on a coupled oscillators network distributed over the chip.

1.1.1 Clocking in large digital circuits

Nowadays, large digital chip are partitioned into local clock areas (domains) [79, 52, 33, 51]. These domains are also known as isochronous zones [5] or Synchronous Clocking

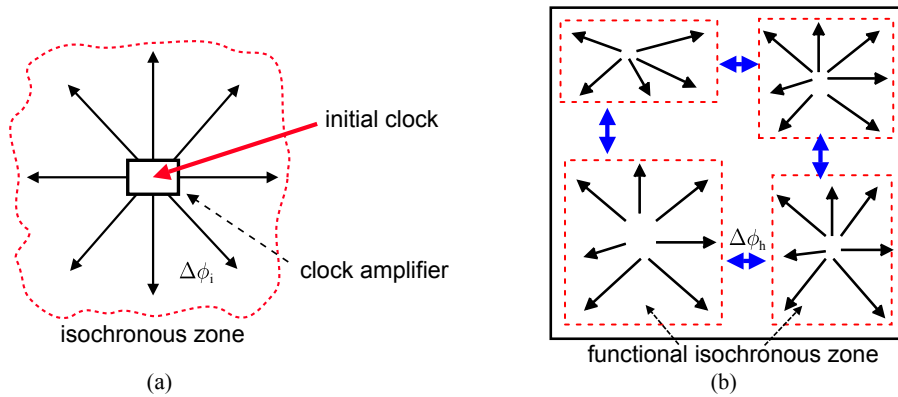


Figure 1.3: **Clock domains in a SoC:** (a) isochronous zone and (b) their placement in SoC

Areas (SCA) [30]. Each zone has its own local clock; these zones are small, so the clock distribution inside the zones can be achieved without difficulties by conventional techniques. The size of the zone is determined by the maximal delay that can experience the clock signal while being routed through the zone (cf. Fig. 1.3): this delay should be small enough for the constraint-less clock routing. The dimensions of the zone depends on the technology, and the number of gates/latches is approximately 200-300 thousands.

The communication inside the SCAs is synchronous; the communication between the blocks situated in different SCAs can be synchronous or asynchronous. In the first case the chip is GSLs (Globally Synchronous Locally Synchronous), in the latter case the chip is GALS.

In a GSLs system, the local clock oscillators are synchronized in phase. In this way, synchronous communications are possible through the borders of the neighboring SCAs, and the whole SoC can be designed with the conventional methodology of synchronous digital design. The advantages of the synchronous circuits are their reliability, deterministic behavior and high communication rate.

However, the synchronization of local clocks of the SCAs in a large chip is a very difficult task. The global clock signal must be routed over the whole chip surface, whereas the delay matching should be guaranteed with a picosecond resolution: this goal is very challenging, and its power and area cost are prohibitive. To date, mainly conventional clocking techniques like clock tree and grid are used in the commercial circuits. These clock generation and distribution architectures are sometimes reinforced by resynchronization blocks, so to compensate the timing errors due to mismatch, temperature and supply drift.

Because of these difficulties, the SoC engineers orient their choice toward GALS architecture which does not require the global clock. In a GALS system, the local SCA clocks are independent, and the communication between the zones is asynchronous. The bisynchronous interfaces guaranteeing the signal integrity during such communication are complex blocks which also reduce the communication speed. Moreover, asynchronous circuits may experience metastability whose probability can be reduced to very small values, but never being eliminated. Finally, asynchronous circuits do not have a deterministic behavior: their reliability is more difficult to guarantee at the design stage than for the synchronous circuits.

This work is addressed to the designers developing GSLS circuits: the proposed solution is an alternative to traditional global clock distribution techniques.

1.1.2 Clock error issues

In any synchronous circuit design, the engineers mainly develop the combinatorial networks performing the required functions. These networks are supplied with the cascading registers which synchronize the flow of information among the data propagation paths.

Fig. 1.4 summarizes the principle of the synchronous communication. The data is set by register R_i synchronously with an event of the clock CLK_i . After a delay due to interconnection lines and to the combinatorial logic processing the data, the signal arrives at the input of the register R_f synchronized by the clock CLK_f . Ideally, the two clocks must be identical (synchronous). The processed data should arrive at the input of the register R_f before the arriving of the next clock event of CLK_f . In this way, the whole delay of the data path, including the delay and setup/hold time of R_f must be less than the clock period.

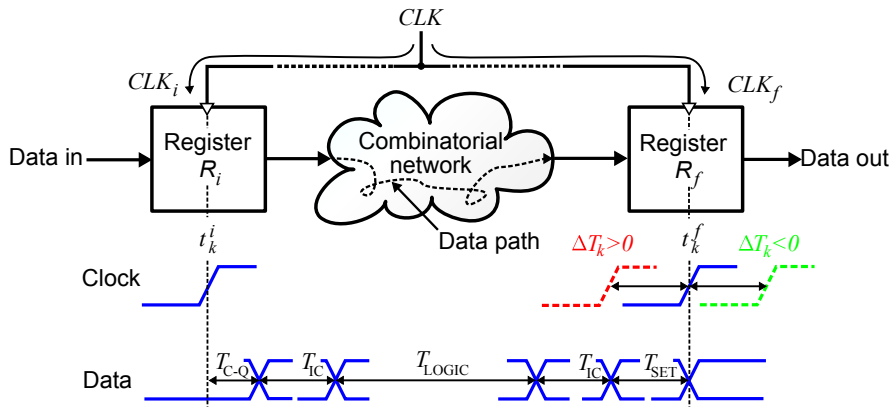


Figure 1.4: **Flow of the data in clocked system and its timing diagram**

In practice, the clocks CLK_i and CLK_f experience a synchronization error : the later can be seen as difference in the instants of the must-to-be synchronous events of the two clocks:

$$\Delta T_k = t_k^i - t_k^f, \quad (1.1)$$

where t_k^i and t_k^f are the instants of arriving of k^{th} clock events at the input of each registers. ΔT_k is a discrete random process usually considered as ergodic, and characterized by the average value called *skew* and the residual zero-centered random process called *jitter* (Fig. 1.5). In practice jitter is considered as being determined and limited, and the process ΔT_k is characterized by the maximal and minimal error values (measured during a reasonable observation time).

It can be seen that if the ΔT is negative, the time available for the data processing and propagation is reduced and equal to $T + \Delta T$. To cope with that, the following most commonly used solutions are possible: pipelining, logic delay reduction and clock frequency reduction. All these solutions decrease the performance of the data processing/transmission. Hence, the

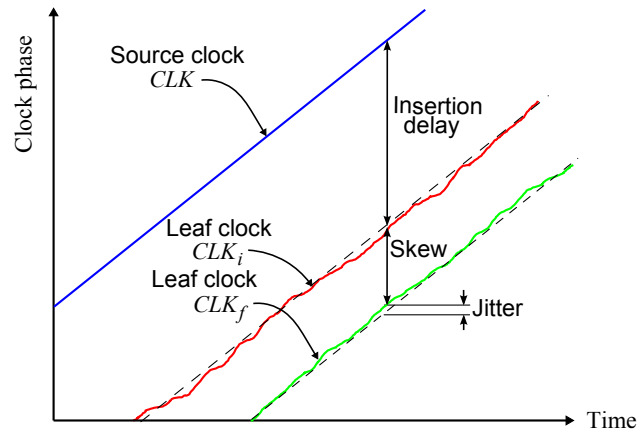


Figure 1.5: Definition of the skew and jitter

clock error must be mastered and kept under some limits defined by the chip specification, and usually not exceeding few percents of the clock period.

The major sources of clock error in clock distribution circuits are the following:

- mismatch in length of paths from a clock source to the registers;
- delays of buffers of the clock distribution network;
- mismatch of parameters of the distribution lines: e.g. resistance, dielectric constant, thickness;
- mismatch of parameters of the active electronic devices participating in the clock distribution (e.g. threshold voltage of MOS devices, local supply voltage);
- intrinsic internal noise of the active electronic devices participating in the clock distribution;

In the next section we propose a review of conventional techniques of the clock generation and distribution.

1.2 State of the art: synchronous clocking in modern SoC

Nowadays, many variants of clocking systems are used in ultra and large scale integration circuits. This is explained by a natural desire of engineers to find the most appropriate solution for each specific case. In particular, high-performance circuits like graphic processors set strong constraints on the clocking scheme: multiple clock domains, very precise frequency/phase definition, integration of the clock generator with global power management mechanisms like power saving, overclocking, etc. This section introduces few conventional and emerging clocking schemes.

1.2.1 Conventional clock trees

The diagrams of a commonly used tree structures in clock distribution networks are shown in Fig. 1.6. If the delays associated with each of the branches are equal, all receivers experience the same clock signal. The matching between different clock paths is the key point of the network design. In large area chips designed in advanced CMOS technology such a global matching is difficult to achieve.

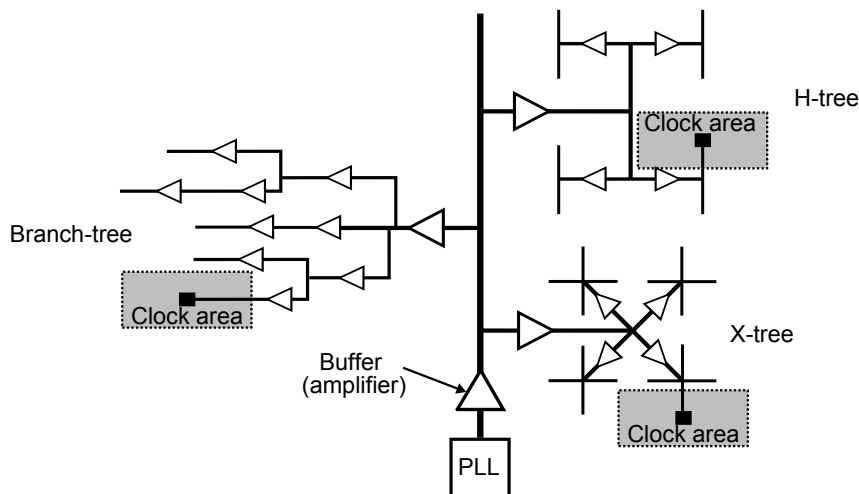


Figure 1.6: **Examples of conventional clock distribution tree structures:** Branch-tree, H-tree and X-tree

Even if the positions of the receivers are symmetric with regard to the source point of the clock, mismatches between the buffers and the lines introduce uncontrollable skew. In practice, the local clock areas have different sizes and position, making difficult a perfect clock tree equilibrium. To solve the problem of the mismatch, the size of the lines and buffers must be increased so to become less sensitive to fabrication errors. However, this solution has a large power consumption penalty [68, 7].

Despite the presented drawback, such a type of clock distribution techniques is widespread, in particular in middle-size low and average frequency chips (few hundreds of MHz), in some commercial processors [10, 7], for local clocking of the different blocks of SoC, such as cache, ALU, etc.

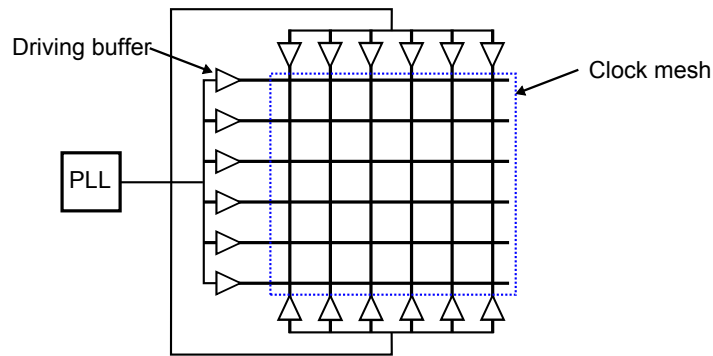


Figure 1.7: **Typical mesh clock network**

Clock grid

A simple clock network with grid architecture is shown in Fig. 1.7. The grid has the advantage of being relatively insensitive to the size and position of the receiver clock areas. The clock accuracy is mainly defined by the density and regularity of the grid geometry [1].

The dense grids demonstrate the best timing accuracy performance. Even if some grid lines are not perfectly positioned, the impact on the accuracy of the clock is negligible. However, meshes can dissipate a lot of energy because they contain a large number of redundant lines whose capacity is high and by consequence, the associated switching power is substantial.

Skew compensation in clock trees

Improvements of the original tree distribution system consist in providing the clock generator with a skew compensation mechanism. It means that after the fabrication, mismatches and imperfections caused by fabrication errors are compensated and the final skew is diminished.

Many skew compensation strategies have been proposed in the literature [58, 72, 13, 32, 18, 73, 77, 27, 60], but only few of them have been used in practice.

In a centralized skew compensation technique, the skew monitoring and control are performed by a dedicated controller called skew compensator gathering information about the skew in all characteristic points of the chip. Its role is to ensure that the clock ticks sent to all clock domains arrive at the same time.

Fig. 1.8 presents the solution proposed in [27]. Each local synchronous clocking area has three connections with the compensator: a link to the direct path (forward path) and two links for the return (feedback path). The first link connects the clock source with the roots of each local tree. Two return signals are provided: one from the root of the local tree, and one from one of the leaves. The first return link is used to measure the round-trip time of the clock pulse, the second one is used to measure delay produced by the local tree. The compensator is capable to correct the phase error by increasing or decreasing the propagation time of the forward path. This approach has two principal drawbacks: the resolution of the measuring circuit is relatively low and the accuracy of the phase correction applied to direct path signals is limited. It is difficult to implement this strategy because of the requirements imposed on the

interconnections of the various regions. In addition, this strategy does not solve the problem of high power consumption associated with buffering of high frequency clock signals.

The decentralized approach of skew compensation technique is represented by implementation described in [60]. This microprocessor uses a clock distribution network comprising two trunks. Each trunk is governed by its own clock tree (cf. Fig. 1.9). It is possible to precisely adjust the clock in each trunk distributed through an variable delay line. A phase comparator (PC) located between the two trunks compares the arrival time of the clock ticks. In case of imbalance of clock ticks between two trunks, the delay of the distribution chain is adjusted via variable delay lines (VDL). This approach is particularly effective for compensation of skew associated with interconnections, and process, supply voltage, temperature (PVT) variations. Despite these advantages, this strategy requires global distribution of the high frequency clock, which should be buffered. Hence it requires a lot of power to keep the amplitude of the clock on appropriate level.

We note that neither centralized nor decentralized skew compensation technique solve the problem of the short-term jitter.

1.2.2 Optical distribution technique

The optical clock distribution is a promising alternative to the electrical one. Compared to the wires, the optical lines are not affected by cross talk with other on-chip signals. Moreover, they are immune to noise sources of SoC, have higher bandwidth, timing accuracy and potentially consume less power than electrical distribution circuits [14]. The examples of the research and analysis in the direction of optical distribution can be found in [9, 55, 64, 20, 41, 69]. The major electrical losses caused by the metal wires and buffers are avoided. Consequently, the interconnect delay is negligible and power consumption is trimmed. The example of structure of such a network is presented in Fig. 1.10.

The main disadvantage of this approach is the problem of integration of optical circuits on a CMOS chip. Particularly, the study [11] highlighted several problems related to the performance of existing optical modulators, receivers and splitters in the CMOS silicon environment. High performance photonic devices require special fabrication options, which

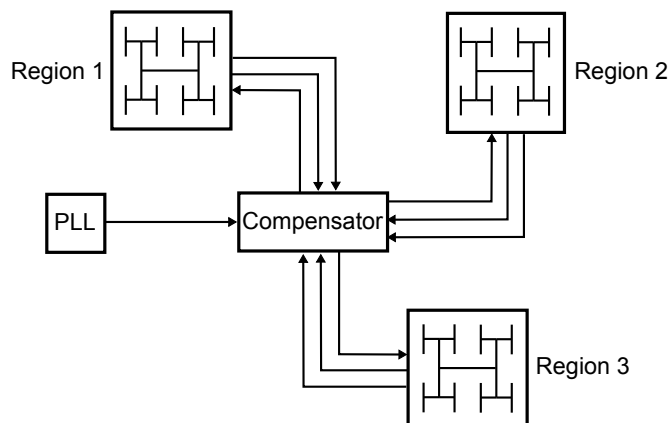


Figure 1.8: Centralized skew compensation technique from [27]

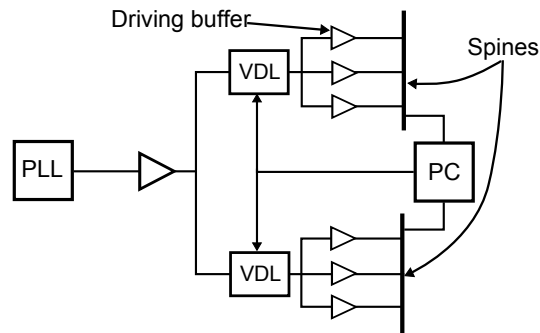


Figure 1.9: **Example of decentralized skew compensation technique proposed in [60]**

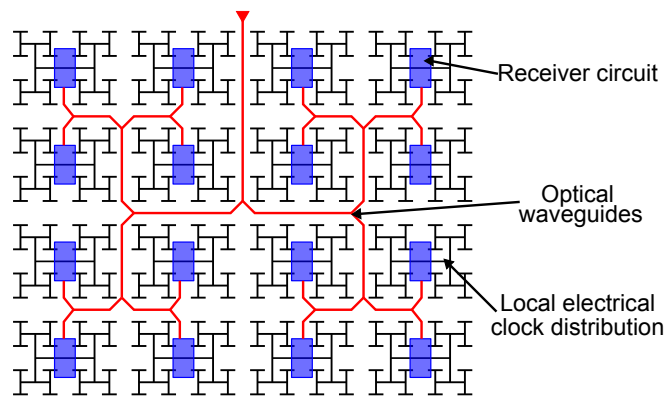


Figure 1.10: **An optical clock distribution network from [55]**

increase the cost of the chip. The design flow remains fully custom since the EDA tools has not been developed yet. The studies [11, 14] show that for the effective realization of this approach, a significant efforts in study of materials and devices should be done. More recent research [65], confirmed a keen interest to this approach. The technological evolution reduced the number of integration issues, so that the test circuit has been implemented and tested. However, the question of formal and automated design flow remains open.

1.2.3 Multioscillator architectures

Multioscillator architectures implement multiple clock sources (oscillators) distributed over the chip area. Each oscillator generates the local clock for one SCA. The oscillators are synchronized with their immediate neighbors. When properly designed, the entire network is synchronized in frequency and in phase. Such an architecture has two advantages. First, the clock signal is regenerated locally, so the perturbations induced by long routed lines in tree architectures are avoided. Secondly, only neighboring oscillators are linked, so, there is no transmission of timing information on long distances.

There are two mechanisms of oscillator linking, i.e. coupling: in the energy domain (voltage/current) and the phase domain.

Injection locked oscillator networks

Injection coupling consists in injecting a part of the energy of one oscillator into the other, and vice versa. The injection coupling is a very old concept first discovered by Christiaan Huygens in far 1657. We present here two examples of clocking systems based on this mechanism.

Muzino et al. [42] present a comprehensive study of a distribution network based on a network of voltage controlled oscillators (VCO) coupled in the voltage domain (Fig. 1.11). In this network the oscillators are coupled by equal length l transmission lines inserted between their outputs. The oscillation frequency and phase are fixed by an external reference through a use of an analog PLL. The VCOs drive the local clock distribution network. This technique is analog, with typical weakness associated with analog circuits: a sensitivity to the perturbations, noises, fabrication mismatches, stability issues.

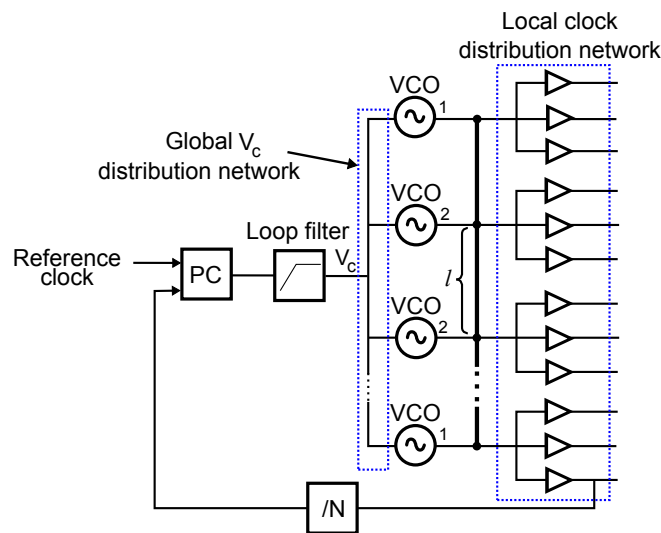


Figure 1.11: **Network of oscillators coupled by voltage presented in [42]:** the number near each output of oscillator indicates the number of nearest neighbors

The research articles [46, 56, 57] propose to use a coupled oscillators network emulating a continuum in which can exist a standing wave. If the linear size of the chip is equal to the half wavelength, all points of the chip oscillate with the same phase, according to the physics of standing waves (Fig. 1.12). The clock areas distributed over the chip use the local values of the standing wave as a local clock source. Again, the principal drawback of this approach is its analog nature: sensitivity to noises, to the geometrical disposition of the receivers, etc. Furthermore, the oscillation amplitude varies from one node to another: it is larger in the middle of the chip (at the anti-node of the wave) and smaller near the edges (at the nodes). This requires special receivers for the local clock recovery.

Networks of oscillators coupled in phase

The target quantity of the clock generators is the phase of the clock events: the energy of the oscillations, the waveform, etc. are not so important for the clock generation. Therefore, the

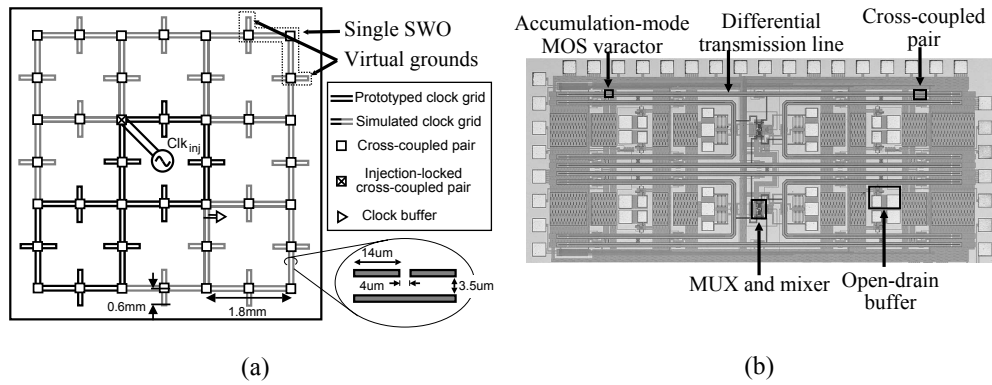


Figure 1.12: **Standing-wave clocking principle (from [46])**: schematic (a), chip microphotograph (b)

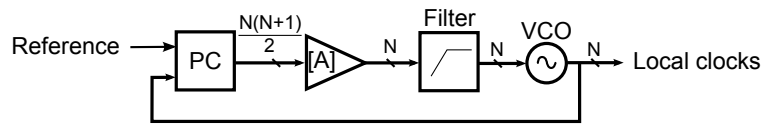


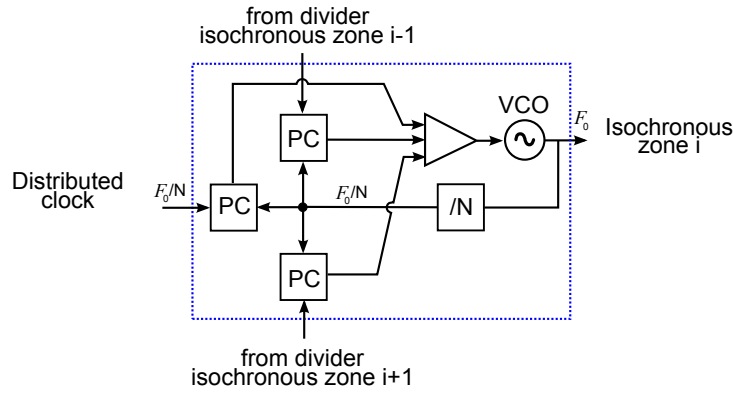
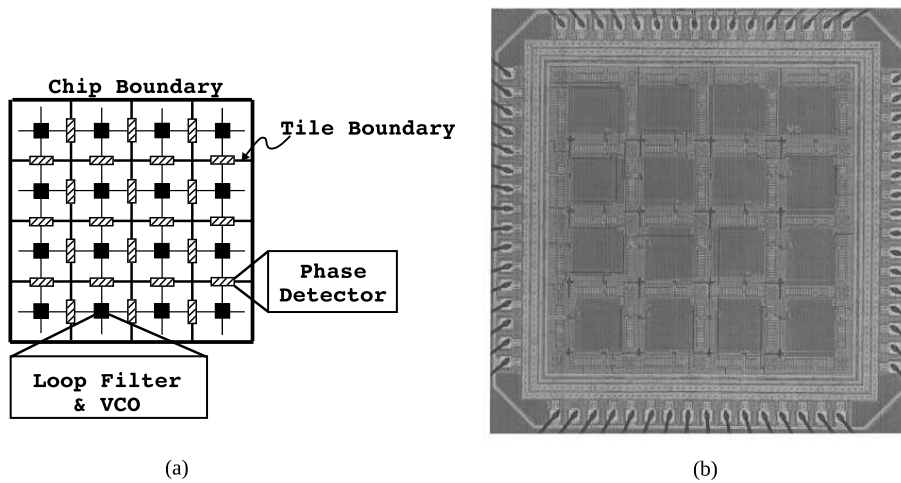
Figure 1.13: **Distributed PLL**

injection coupled network focus on what is not the most important for the clocking application. By contrast, the phase domain coupling considers only the information about the phase of the oscillators. The phase of an oscillator is the argument of the periodic function describing the oscillator trajectory in the space of the state variable; the phase is hence an analog quantity. However, the phase can't be measured directly; in general, the value of the phase is detected when the oscillation trajectory crosses a given plane, typically, the zero level of voltage/current. Hence, the phase is sampled. In addition, the instant of the phase detection can be considered as an event. Such a discrete event representation of the phase makes it less sensitive to the analog issues like perturbations, noises, etc. Moreover, a processing by digital circuit is possible. A network of oscillator coupled in phase is also called *network of coupled phase locked loops* (PLL), or a *vector PLL*. The idea of a clock generator based on a network of phase coupled oscillators has been proposed and developed in several studies [49, 5, 28, 54, 53, 23].

A general architecture of a distributed PLL is presented in Fig. 1.13. It contains N oscillators, N loop filters and up to $N(N+1)/2$ phase comparators for comparison of the phases between any clock signals (local clocks and reference). In clocking applications, the network is a Cartesian mesh where the phase error detection is performed between the nearest neighbors. The system parameters must be chosen in a way so to synchronize the phase of the outputs of the oscillators with the reference signal.

A node of a PLL network includes a loop filter and a VCO (Fig. 1.14). The phase comparator compares the phase error between each couple of neighboring nodes. The use of such a system for clock generation was studied in [49], and a practical implementation was presented in [23].

An example of implementation is presented in Fig. 1.15. The oscillators generate a clock

Figure 1.14: **Network node**Figure 1.15: **16-node distributed clocking network**: (a) structure and (b) die microphotograph

signals at each node of the network. Phase comparators are situated at the boundary between neighboring local clock area (named *tile* in the original publication). They produce the error signals which are then summed and used to adjust the oscillation frequency of the node, as in Fig. 1.14. It is shown in [49] that for the synchronization of all local clocks, it is necessary and sufficient to synchronize the clock of each node with the clocks of the neighboring nodes.

Similarly to other mentioned above multioscillator systems, the analog implementation of such a solution is characterized by a low compatibility with modern SoC environment. In spite of the advantages of this approach, there is no industrial use of such a clock network, to our best knowledge.

1.2.4 Synchronous clocking architectures: conclusion

In this section we have studied few clocking approaches in modern VLSI systems aiming a global synchronization on GALS chips. With the growth of the chip dimensions, the complexity of implementation of conventional and centralized clocking systems rises exponentially. Distributed decentralized clocking architectures seem more appropriate for large SoC,

since they do not require chip-length wires for the global clock distribution. However, the principal drawback of the existing distributed clock generators is their analog nature, which makes them weakly compatible with the digital environment of the SoCs and with the standard EDA tools existing for digital systems design.

The work presented in this PhD report focuses on multioscillator clocking architecture in which the oscillators are coupled in the phase domain. The principal contribution of this work consists in the study of fully digital implementation of the analog PLL networks originally proposed by Pratt, Nguyen, Chandrakasan and Gutnik [49, 23].

1.3 Thesis outline and contribution

This PhD project studies an alternative clocking solution for large digital systems on chip: a network of all-digital PLLs is proposed for synchronization of local clocks of all synchronous clocking areas of the chip. The main advantage of this solution is the use of the local control loops for the global network synchronization. Moreover, the proposed solution is based on digital circuits, and is compatible with the environment of the SoC using the generated clock.

This document is organized as follows.

Chapter 2 presents the studied system. It describes the topology of the ADPLLs network and the architecture of the nodes, and discusses the issues related to the system-level design of the ADPLL network. In particular, the question of the selection of the desirable synchronization mode is addressed. The chapter ends with a summary of the specifications for each block of the network whose design is presented in Chapters 3-5.

Chapter 3 focuses on the development of a highly linear, compact, portable and reconfigurable 1-2.5 GHz 10 bit DCO based on a ring oscillator architecture. The whole design process is presented, starting from the system-level design down to the layout design. The chip has been fabricated in CMOS 65 nm technology and the measurement results are presented.

Chapter 4 presents the design of the blocks measuring and processing the phase error: the digital phase comparator and the digital proportional-integral filter. The phase comparator has two particularities. Firstly, it measures not only the phase error but also the sign of the frequency error. For this reason it is called *phase frequency detector*. A formal characterization of the frequency detection ability is provided. Secondly, the information about the phase/frequency error is provided in a digital form. The design of a 5 bit PFD with 32 ps resolution is presented.

To provide a working prototype demonstrating the proposed clock generation technique, a 4x4 ADPLL network in 65 nm technology has been designed. In order to validate the design before the chip fabrication, the network architecture has been tested on an FPGA platform. The two implementations are presented in Chapter 5. The chapter is finalized with the results of measurements of the fabricated CMOS prototype.

The report is finished by conclusions and perspectives for the development and research associated with proposed clocking approach. The academic contribution consists in the articles published and presented on international conferences.

Chapter 2

Network of distributed ADPLLs

Contents

2.1	Introduction	15
2.2	Proposed clocking architecture	16
2.3	Multiplicity of synchronisation modes	25
2.4	Stability of the PLL networks	29
2.5	Modeling of the clocking network	30
2.6	Conclusion	38

2.1 Introduction

In this chapter, we present an architecture of the multioscillator clock generator based on a network of distributed all-digital PLLs developed in this PhD project. The functionality of each block of the architecture is specified. This information is the input for the design of the prototypes presented in Chapter 3 and Chapter 5.

In the first Section 2.2 we describe the topology of a PLL network for clock generation and we explain the principles of its operation. Then, we present the original idea of the PhD project: a digital implementation of the coupled PLL network for clock generation. Then, each block of the architecture is described, and their specifications are derived.

Large-signal nonlinear phenomena and stability issues are discussed in Section 2.3 and Section 2.4: these problems are related to the synchronization of the network, and are of first importance for the clock generation.

Section 2.5 describes the system-level behavioral model of the network and demonstrates the results of its system level simulation.

2.2 Proposed clocking architecture

2.2.1 Network of coupled PLLs

In 1995 Gill Pratt and John Nguyen proposed a distributed clock generator based on network of coupled analog PLLs [49]. Our work is based on this architecture. For this reason, this subsection provides essential information about it.

The proposed clock generator belongs to the family of multioscillator architectures based on a network of coupled oscillators. In such a clocking scheme, Fig. 2.1, a chip is partitioned into local clock areas, each of them having its own clock generator (oscillator) which must be synchronized with its neighbors in the phase domain. The goal of the distributed PLL network is to synchronize each oscillator in phase and in frequency*. In a steady state, such a network is a source of fully synchronous distributed local clocks.

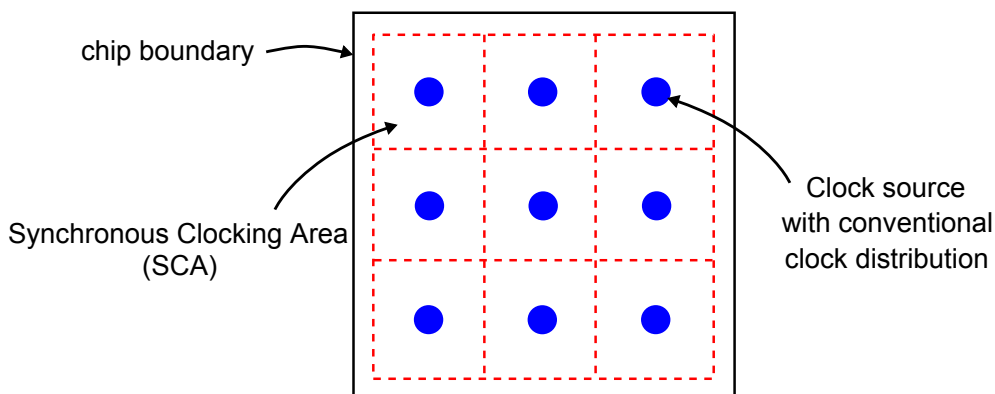


Figure 2.1: **Basic idea of multioscillator clocking approach**

The architecture proposed by Pratt and Nguyen is a 2 dimensional Cartesian mesh network, where the nodes are the local clock generators and the arcs represent the coupling links between the local generators. Each local generator is linked only with its immediate Cartesian neighbors. Such a topology requires the shortest information transmission paths – which is a main advantage of such an architecture comparing with centralized clock generation approaches.

The coupling between the oscillators is implemented in the phase domain via *phase comparators*, Fig. 2.2(a). Each phase comparator provides a measure of the error between the phases of two oscillators. This measure is then used by the control electronics associated with each oscillator in order to provide a control signal forcing the oscillators to synchronize with its neighbors. The control signal impacts directly the frequency of the oscillators – which is a derivative of the oscillator phase. For each oscillator a , the phase error $e_{a,b}$ is defined as the difference between its own phase and the phase of its neighbor b . The phase ϕ and the phase error are defined modulo 2π : the most common definition of the phase error is [49]:

$$e_{a,b} = (\pi + \phi_a - \phi_b) \bmod 2\pi - \pi. \quad (2.1)$$

*A synchronization in phase implies a synchronization in frequency

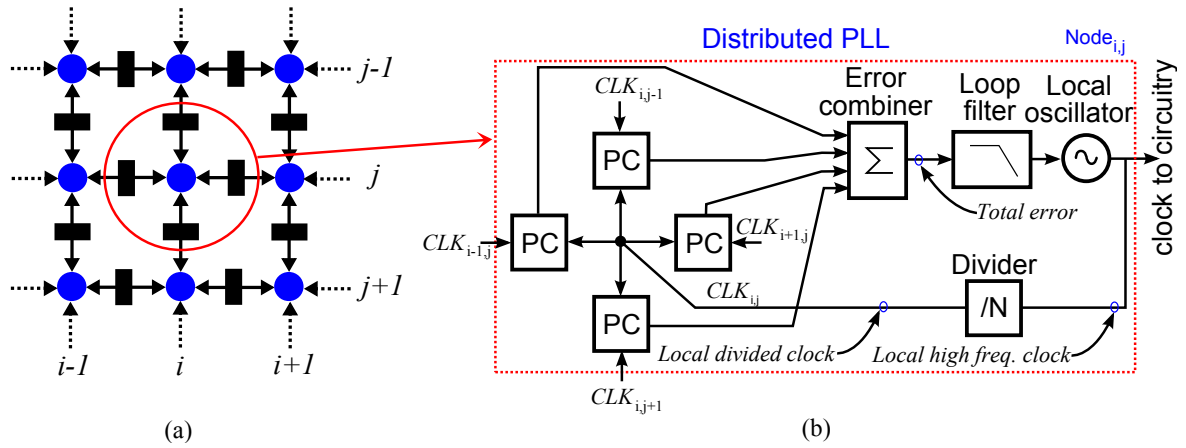


Figure 2.2: **Topology of the proposed clock network and architecture of the network node**

According to this definition, $e_{a,b}$ can have values in the interval $[-\pi, \pi]$. The phase error "seen" by the oscillator b between b and a is $e_{b,a} = -e_{a,b}$. For this reason, each phase comparator is associated with two oscillators and generates two phase error signals e and $-e$.

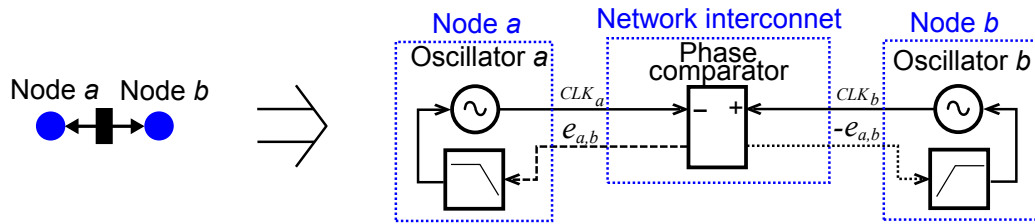


Figure 2.3: **Phase coupling between two oscillators**

Fig. 2.3 presents an example of an autonomous (without input reference signal) network composed of two oscillators. In a more complex network, each oscillator i receives n_i errors with its n_i neighbors (Fig. 2.2). These errors are processed by the control block including an error combiner and a loop filter. The error combiner can be in the simplest case a weighted adder. The filter processes the combined error signal called *Total error* so to generate a control signal on the oscillator input. The objective of the control is to keep at the signal *Total error* close to zero.

It has been proved that such a system has a stable operation mode in which all oscillators have the same phase. The existence of this mode is conditioned by a right choice of the network parameters, in particular, the parameters of the control blocks of the nodes. It has been shown that apart the mode in which all oscillators have the same phase, there are several modes in which a fixed phase shifts between the oscillators exists. Pratt and Nguyen studied this phenomenon and indicated several solutions for the selection of the desirable stable synchronous mode. This issue is discussed in Section 2.3.

The architecture proposed by Pratt and Nguyen has been successfully implemented by Gutnik and Chandrakasan [24]. The implemented chip contains 4x4 VCOs synchronized by

a PLL network.

The analog nature of this system is its main drawback for clock generation. The clock generator is usually integrated with the digital blocks which use the clock signal. The performance of an analog PLL in a digital environment may drastically degrade because of perturbations due to switching in the digital circuits. Moreover, using analog PLL network for clock generation makes more difficult a technology migration and reduces the design portability of the overall SoC.

The study of digital implementation of this architecture is the main focus of our work. This study aims to improve the portability, scalability and compatibility with the digital environment of the PLL network based clock generator. In the next subsection we present the principles of digital phase synthesis and its advantages for the distributed clock generation.

2.2.2 Digital phase synthesis

The principle of digital phase synthesis can be illustrated by the example of a single All-Digital Phase-Locked Loop (ADPLL). Known since a long time but actively used since one decade, the digital PLLs has recently gained ground on the analog PLLs [63]. The ADPLL operates following the same principle as their analog counterparts and functionally has the same structure (Fig. 2.4).

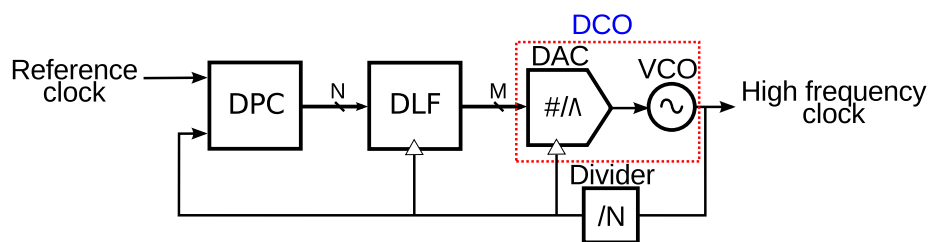


Figure 2.4: **Block diagram of the ADPLL**

An analog phase comparator generating a signal proportional to the phase error is replaced by a digital phase comparator (DPC), which generates a digital code proportional to the error. This code is processed by a digital loop filter (DLF). Thereafter, the signal from the filter output is used to control the VCO through a Digital-to-Analog Converter (DAC). The functions of the VCO and DAC are often integrated into a single block called Digitally-Controlled Oscillator (DCO). The divider defines the ratio between the reference (input) and output frequencies of the ADPLL, i.e. the frequency multiplication factor.

A digital PLL is a system processing mixed analog/digital signals: the target control quantity is analog (the phase), and the DCO and the digital phase comparators are DAC and ADC respectively. However, as it will be shown in the next two chapters, these two blocks can be implemented with digital cells. In addition, the phase is represented by the events of threshold crossing by the DCO output signal. Such a circuitry is weakly sensitive to the perturbations generated by digital circuit environment. Hence, the drawbacks usually associated with analog circuits are attenuated.

A digital PLL has the following particular property. The phase error (the quantity to be regulated) is sampled by the divided DCO output signal; the same signal is used for the digital processing block clocking. However, the DCO output signal depends on what outputs the digital processing block. By consequence, an ADPLL is a self-sampled system. This fact significantly complicates the system analysis ; the properties of the ADPLL network related to the self-sampled operation were studied in the frame of the PhD project of Jean-Michel Akre [3]. Comparing with analog frequency synthesis, the digital phase synthesis has numerous advantages:

- **Use of digital design techniques.** Digital design costs less time and resources than analog design, thanks to the use of computer-aided design (CAD): automated synthesis, place and route flow.
- **Reconfigurability and programmability.** Digital implementation of the phase error processing block allows a use of complex processing algorithms not limited by simple PI or PID filtering as it is the case of the analog PLLs. An intelligence can be added to the clocking system: the control of the synchronisation mode, more effective filtering of perturbations, design-for-test (DFT) features, etc.
- **Immunity to perturbations.** The signals within the ADPLL network blocks are digital: hence, they are less sensitive to the noise.

While a substantial expertise in the design of single ADPLLs has been accumulated during past decade, ADPLL network is a insufficiently studied system, particularly in what concerns its implementation in submicronic CMOS technologies. The next subsection present the architecture of the ADPLL network blocks.

2.2.3 Blocks of the ADPLL network

In this subsection we describe the architecture and discuss the basic parameters of the ADPLL blocks in the context of clocking network. We start from the oscillator, since it is the most critical block of the ADPLL system.

Digitally controlled oscillator

The oscillator in an ADPLL generates the output clock signal with the frequency defined by the input digital M -bit digital control word. As it is explained in Section 2.2.2, the DCO is generally a DAC whose output analog quantity is the frequency of oscillations at the output. In general the DCO DAC is supposed to be linear:

$$F_{osc} = F_0 + \Delta F K, \quad (2.2)$$

where F_{osc} is the output frequency, ΔF is the DAC resolution (the frequency step), K is the value of the input code, F_0 is the DCO initial frequency. A DCO is characterized by the following parameters:

- **Frequency tuning range.** It specifies the upper and lower limits of the output signal frequency. In the context of clocking network it defines the tuning range of the clock frequency.
- **Central frequency.** This is the middle frequency of the frequency tuning range. In this project, the central frequency of the DCO defines the nominal frequency of the generated clock.
- **Frequency tuning step.** This parameter is the resolution of the DCO (cf. Eq. (2.2)). It defines the overall accuracy of the system (cf. the next subsection).
- **Power consumption and area.** These parameters are obviously critical for the clocking system. They are at the base of a figure of merit allowing a comparison of different clocking approaches.
- **Input word width.** This is the number of bits composing the input word. Together with the frequency tuning step this parameter defines the tuning range. This parameter has a direct impact on the complexity of the error processing blocks.
- **Linearity/monotonicity.** The transfer function of the oscillator relates the output frequency with the input code. This function is usually sought to be linear (cf. Eq. (2.2)). For the PLL applications, a default in the linearity is not critical, since in the steady state mode the amplitude of an input DCO code is small. However, the error on linearity has the same impact as the error on the DCO gain. If the error is large, the performances of the PLL can be strongly affected.

The monotonicity of the code-frequency DCO characteristic is critical for the PLL application: a local inversion of the DCO characteristic slope sign is equivalent to an inversion of the PLL feedback sign. That immediately leads to an unstable behavior. Therefore, design techniques guaranteeing the monotonicity should be employed.

- **Phase noise.** This parameter defines the jitter at the output of the DCO. Although the DCO jitter directly impacts the quality of the generated clock, it is much less critical than in the RF applications. This allows a use of simple and "economic" oscillator architectures such as ring CMOS (comparing with LC resonators required in RF architectures).

Since the goal of our project is a proof of feasibility of the ADPLL network based clock generation, we do not focus on a design of DCO with state-of-the-art performance. The most important characteristics of the DCO for this project are those critical for functional implementation of the network: in particular the parameters related to the frequency range, frequency step and monotonicity. The specification related issues will be discussed in Section 2.2.4.

Loop Filter

The loop filter implements the processing of the phase error and the generation of the M -bit control digital word defining the frequency of DCO. Digital and analog PLLs employ proportional-integral (PI) or proportional-integral-derivative (PID) filters [22, 78, 16]. The

study of Pratt and Nguyen demonstrated that a simple PI filter is a sufficient solution for synchronization of a PLL analog network. The studies of Akre et al. and Korniienko et al. demonstrated that this stands as well for digital PLL network [3, 34, 35, 36].

This digital filter is a synchronous SISO (Single Input Single Output) digital signal processing block. Its operation is described by the equation:

$$H(z) = \alpha + \beta \frac{1}{1 - z^{-1}} = \frac{(\alpha + \beta) - \alpha z^{-1}}{1 - z^{-1}} \quad (2.3)$$

where α and β are the gain coefficients of the proportional and the integral paths respectively. The topology of the loop filter is shown in a Fig. 2.5.

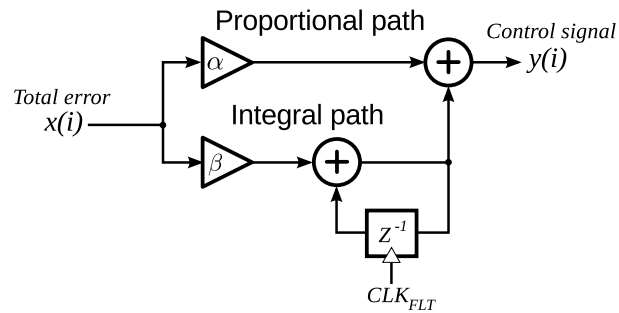


Figure 2.5: **Block diagram of the PI filter**

In a single ADPLL, the loop filter receives at the input the phase error between the input signal (the reference) and the DCO output. In the context of a network of ADPLLs, one node receives the errors between the local DCO and the neighboring nodes. The loop filter processes the combined value called *Total error* (cf. Fig. 2.2). In the simplest case the *Total error* is the sum of the errors with all neighbors, but more complex preprocessing is possible (weighted sum, shaping, selection of the link directivity, etc.). This preprocessing can satisfy specific requirements of the application, can be dynamically programmable. Its implementation is particularly easy in the digital version of PLL network.

Phase comparator

The phase comparator compares the phases of two clock signals and produces a digital N -bit code proportional to the phase error. The phase comparator is an ADC: its main characteristics are the dynamic range and the resolution. The analog phase comparator has a linear transfer function in a range between $-\pi$ and π ; an ideal digital phase comparator has the same quantized transfer function within the same range (Fig. 2.6(a)). However, the PLL and the PLL network are designed so to have small errors in the normal operation mode: hence, a large dynamic range is not useful. Together with high resolution, it results in a high power and area cost. For this reason, the digital phase comparator may have an input range limited by certain value $\Delta\phi_r$ (Fig. 2.6(b)). The phase comparator quantizes the phase error within this range with resolution Δ_{PFD} .

When designing an ADC, the choice must be made in what concerns the interpretation of the input values inferior to the ADC resolution: are they considered as zero or one? In the

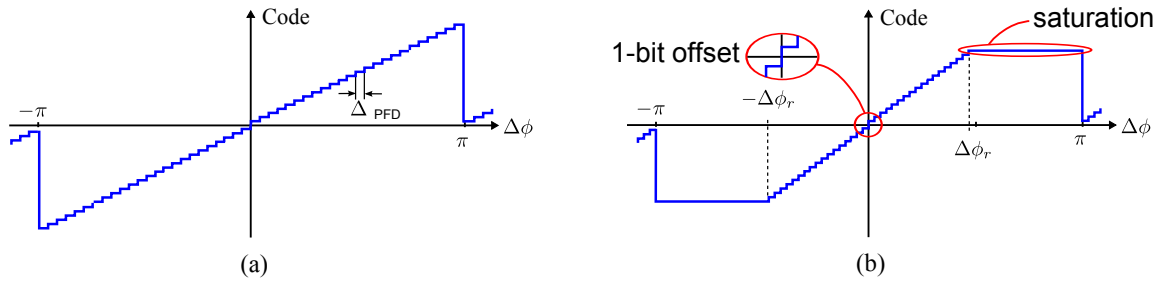


Figure 2.6: **Transfer function of the digital phase comparator:** (a) equivalent of analog linear phase comparator and (b) linear phase comparator with limited range

first case, the ADC has a two resolution steps *dead zone* around the origin. In the second case, the sign of the error is correctly detected even for very small errors. For the digital phase comparator the second option is chosen, since in this case the phase comparator provides more information in the small error range (cf. Fig. 2.6(b))

2.2.4 Specification of clocking network parameters

In this subsection we discuss the system-level specifications defining the constraints for the CMOS design of the ADPLL network. For the choice of these specifications we considered the typical parameters of the modern clocking systems. The discussed specifications are focused on the functional properties of the system such as the output frequency range and the clock error. We did not consider the constraints related to the size and with the power consumption of the blocks, seeking only a proof of the functional and physical feasibility of the architecture.

Frequency step and error processing rate

The most important specification having an impact on the system design is the clock frequency tuning step of the DCO and the error processing rate (the frequency of the internal clock of the control block). These two parameters define the low limit of the output phase error (the tracking error of the regulation). The existence of a low limit of the tracking error is related to the discrete grid of the DCO output frequencies.

This phenomenon is illustrated in Fig. 2.7, on the example of a single ADPLL. Let the input reference signal is periodic, with a linear phase evolution (dotted line). The phase of the DCO output signal must approximate as close as possible this trajectory. However, the output phase can only follow the lines with discrete slopes defined by the DCO frequency grid. The slope (the frequency) can change in function of the input DCO command word arriving with cadence $1/T_s$. If the input reference frequency is not exactly one of the DCO frequency grid values, an ideal control loop switches the DCO output frequency between two neighboring values (solid and dash-dot lines). There is a non-zero tracking error which can be minimized but never zeroed. It can be seen that the "minimal" error is maximal when the reference frequency is exactly in the middle between two neighboring grid values. In this situation, the DCO frequency changes with each next input word, and the output phase error

$\Delta\phi_{max}$ is equal to:

$$\Delta\phi_{ideal} = \frac{\Delta\omega\pi}{2\omega_s} \quad (2.4)$$

where $\Delta\omega$ is the output grid frequency step, ω_s is the DCO input word cadence.

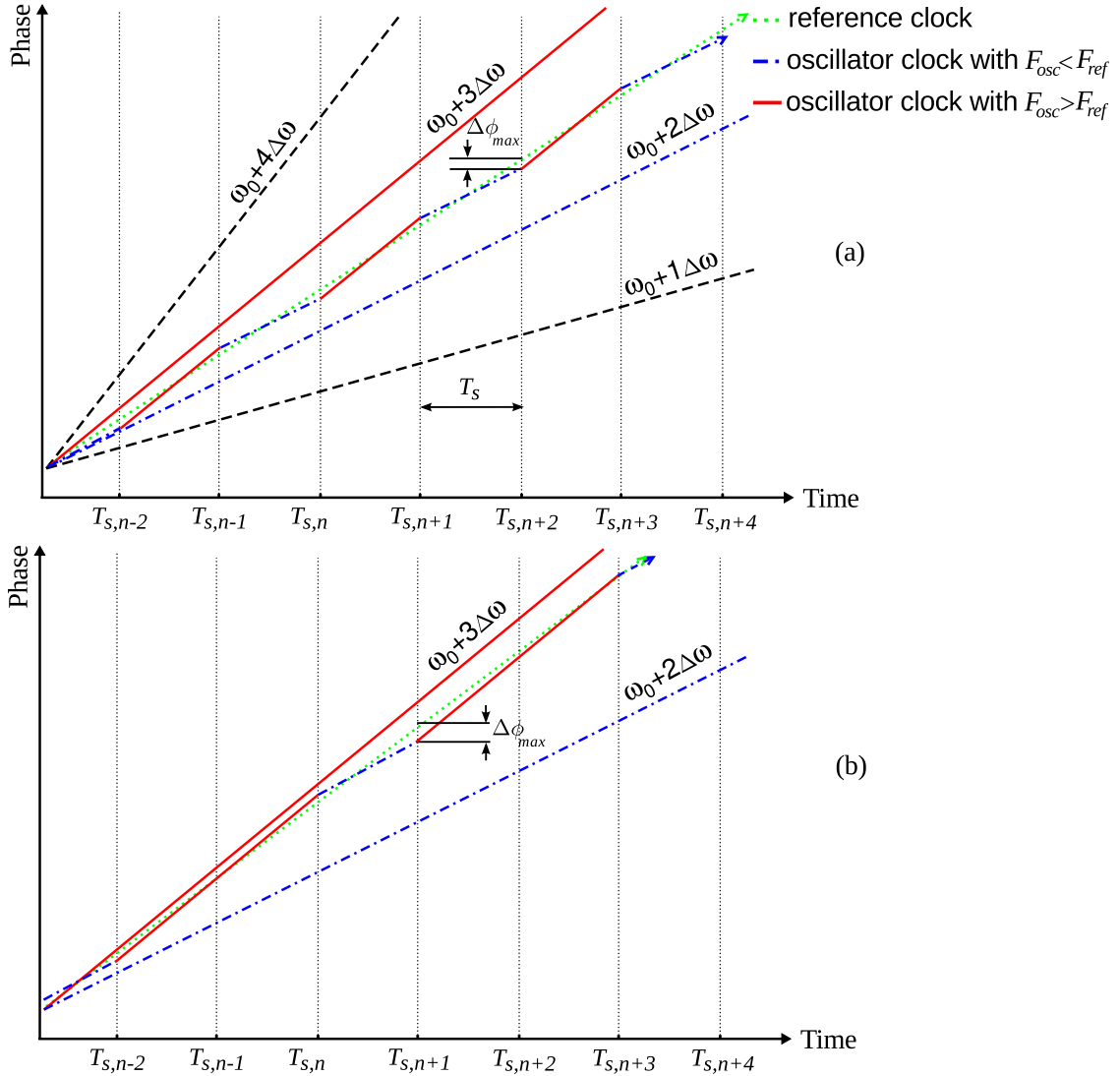


Figure 2.7: **Definition of two main design constraints for the clocking network:** (a) the case when reference frequency exactly in the middle between two neighboring grid values and (b) the case when reference frequency close to the grid value

The choice of the frequency tuning step and of the error processing rate is a trade-off between the hardware complexity and the desired precision; it is strongly application dependent. A low frequency step of the DCO implies a large width of the DCO command word if the tuning range is large. High rate at the DCO input implies a high phase error processing rate, hence, a large consumed power and a large size of the digital processing block.

The main specification is the acceptable clock error: we fixed it to 1 % of 2π . The nominal output frequency is chosen to be 1 GHz: this figure corresponds to the typical need of modern multiprocessor SOCs. The error processing rate is obtained by division of the output DCO frequency by a power of 2 (for simplicity) ; we had a choice between 125,

Table 2.1: **Clocking network implementation specification**

<i>Parameter</i>	<i>Value</i>
Local high frequency F_{HCLK}	1 GHz
Filter clock frequency F_{ref}	250 MHz
Frequency tuning step ΔF	≤ 1 MHz
Frequency tuning range	$\pm 40\%$
Clock period error	$\leq 10\%$ or 100 ps
SCA size	$\approx 300k$ gates

250 and 500 MHz. Given the typical processing rate allowed by the used 65 nm CMOS technology, the error processing rate of 250 MHz was chosen.

Frequency tuning limits

The frequency tuning limit concerns the specifications of the DCO. There are two reasons to choose a wide frequency tuning range. The first one is a need to have a flexible clocking system able to modulate the clock depending on the application requirements. Such a modulation is a part of the SoC power management strategy employed for all modern digital integrated systems [8].

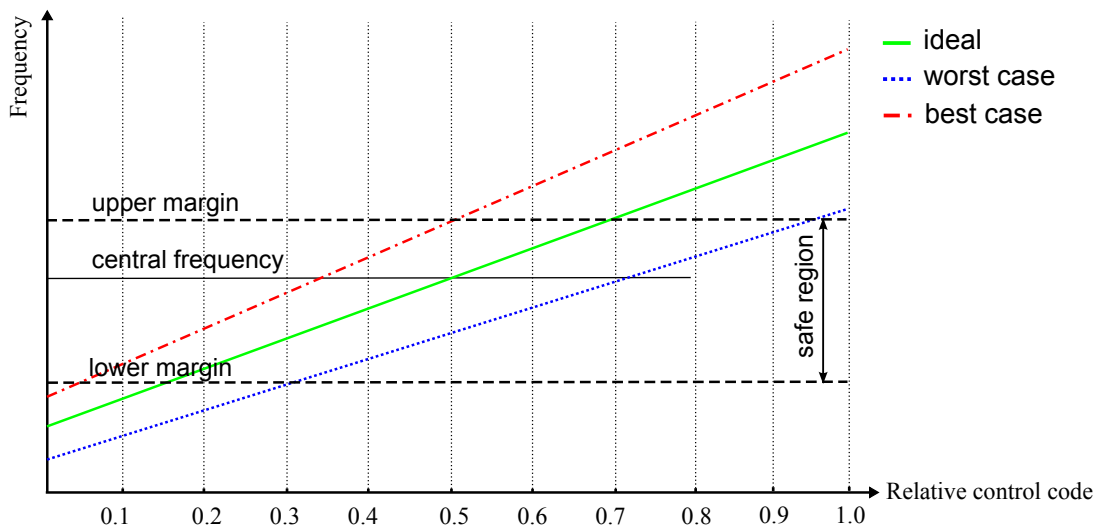


Figure 2.8: **Frequency tuning limits consideration:** variation of tuning curves caused by PVT variations

The second reason is the need of compensation of the DCO frequency mismatch due to PVT variations. The oscillator in ADPLL is a mixed signal block which can have significant sensitivity to these variations. For example, after fabrication, oscillators in different parts of SoC may have up to $\pm 20\%$ distribution over the nominal value, thus each individual DCOs must be able to compensate this mismatch (Fig. 2.8).

For these reasons we defined the tuning range of the oscillators so to be $\pm 40\%$. The summary of clocking network specification is given in a Tab. 2.1.

2.3 Multiplicity of synchronisation modes

The work of Pratt and Nguyen in [49] highlighted a fundamental problem specifically related to the PLL networks. A PLL network may have several modes in which the local oscillators are synchronized in frequency and in phase, but with fixed phase errors between the oscillators. The residual errors may be zero or not. For the clocking applications, only the mode in which the phase errors are zero is suitable for the clocking application. However, when several synchronization modes exist, the actual mode depends on the initial conditions of the system – which cannot be controlled in practice. This section presents this phenomenon in details and provides a review of the solutions to this problem.

2.3.1 Definition of the problem

The multiplicity of synchronization modes has been called *mode-locking* in the literature. It is caused by the cyclic (modular) nature of the phase: the transfer function of the phase comparator is defined modulo 2π (Fig. 2.9).

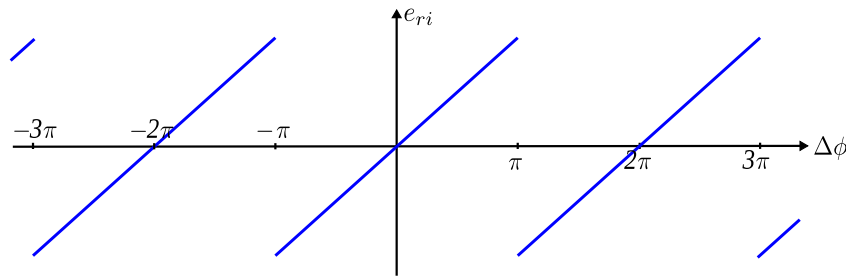


Figure 2.9: Cyclic nature of the conventional analog linear phase comparator

Such a nonlinear transfer function of phase comparator and a large number of degrees of freedom of a PLL network define multiple synchronization modes. Depending on the initial conditions, the system can converge either to desired state (all oscillators has the same phase) or to undesirable state (oscillator phases misaligned).

The mode-lock phenomenon is illustrated on the example of a 4 nodes network Fig. 2.10, in which the error processing block receives the sum of the errors with two neighbors. As it was explained in Section 2.2.1, the error processing block operates so to keep the *Total error* equal to zero. The *Total error* can be zero if :

- 1) the phases ϕ_1 , ϕ_2 , ϕ_3 and ϕ_4 of the oscillators are equal
- 2) the phases have the values given in Fig. 2.10.

In the first case, all phase errors of the network are obviously zero, their sum is zero as well. For the second case, the phase errors are non-zero, but the *Total error* value is zero. For example, for the Node 1: the phase difference with the Node 2 is $\phi_1 - \phi_2 = +\pi/2$, which is compensated by phase difference with the Node 4 $\phi_4 - \phi_1 = -\pi/2$. In this way, the sum for all nodes of this network is zero despite the unequal phases of all oscillators. It can be proven that such a state is stable; once acquiring this operation mode, the network remains in it, since the control objective (zeroing the *Total error* quantity) is fulfilled.

For a network with more complex topology (more nodes), several undesirable states can exist, with fixed phase errors of any values. Hence, in a more complex network, the probability of appearance of undesired synchronization modes increases. Therefore, it is important to take into account this phenomenon during the design of the network. In the following subsection we present the solutions we have studied to design a stable network, spontaneously converging to a state where all oscillators are synchronized in phase and frequency.

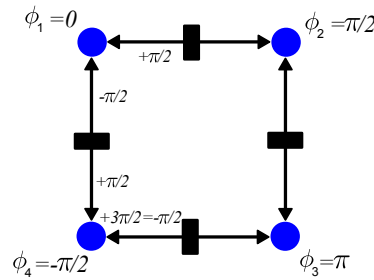


Figure 2.10: **Illustration of the mode-locking phenomenon in a 2×2 mesh network**

2.3.2 Synchronization mode selection

Solution proposed by Pratt and Nguyen

Pratt and Nguyen proposed a technique allowing to make the undesirable modes unstable, while keeping stable the desirable synchronisation mode. This is achieved through a use of a phase comparator with a negative slope transfer characteristic for large phase errors Fig. 2.11(a). Pratt demonstrated that for the networks with Cartesian topology, the transfer function slope must be negative for errors greater than $\pi/2$.

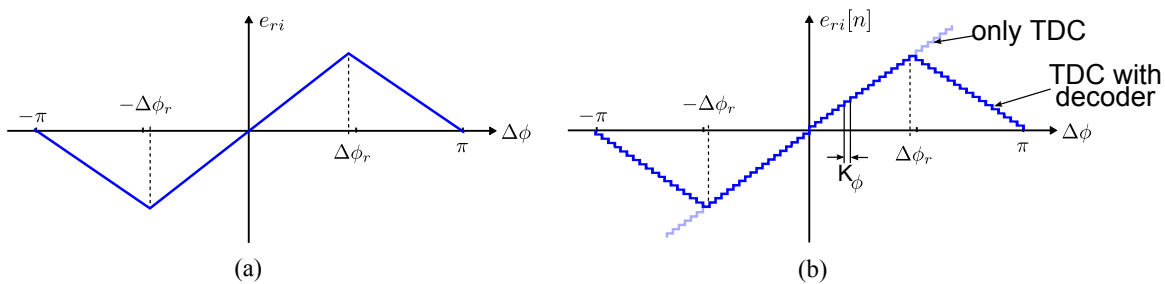


Figure 2.11: **Transfer function of the phase comparator to avoid undesirable states:** (a) proposed in [49] analog PC and (b) its equivalent for digital PLL; $\Delta\phi_r$ must be less than $\pi/2$

This method can be directly applied to the ADPLL network. For this, it is enough to pass the output of the digital phase comparator through an arithmetic block shaping the characteristic (Fig. 2.9) so to obtain the characteristic given in Fig. 2.11(b). Such a solution requires a digital phase comparator with dynamic range $[-\pi, +\pi]$. As we said in Subsection 2.2.3, this may result in a large number of bits of the word coding the phase error – which in turn increases the complexity and the time/energy costs of the phase comparator characteristic shaping. If a high resolution of the phase comparator is desired, the number of the time

quantization steps can be large. For example, if 50 ps resolution step for a signal with 4 ns nominal period is desired, the phase comparator range is composed on 80 steps resulting in a 7 bit output word (together with sign bit).

Saint-Laurent solution

Several studies were done previously on the problem of selection of the desired synchronization mode [54, 53]. In these research articles, to prevent the undesired stable states, authors propose an unidirectional network of PLLs (Fig. 2.12). The nodes of unidirectional network are connected in a master-slave configuration. Such a network topology excludes cycles of propagation of phase information, hence eliminates the possibility of undesired non-zero phase error synchronization modes [49]. However, this solution suffers from the cumulative errors which propagate through the network and accumulate with the increase of the distance from the root node. The phase synchronization in this case may be difficult to obtain if the size of the network is large [43]. Our modeling and experimentation with FPGA prototype of the network confirmed this fact [30], cf. Section 2.5.5.

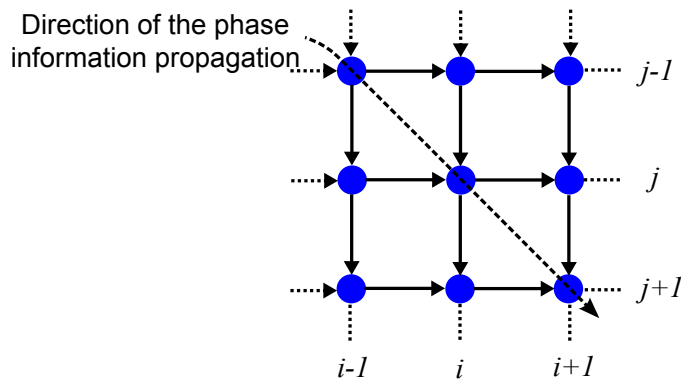


Figure 2.12: **Propagation of the phase information in unidirectional network proposed in [54, 53]**

Dynamic network reconfiguration

This technique is based on the fact that each stable mode of a dynamical system has a basin of attraction: if the system starts from an initial condition corresponding to a point in this basin, the system settled up to the corresponding stable mode. The idea of the method [49] suggested by Pratt and Nguyen is to bring the system to a state close to the desired synchronization mode by configuring the network to be unidirectional as in Fig. 2.12 [54, 53]. When the system is synchronized, the links between the oscillators are reconfigured so to set the network in the bidirectional configuration (Fig. 2.13). Since in this case the bidirectional network starts in a state close to the desired synchronization, it remains in this mode.

This method is particularly suitable for digital implementation since it is very easy to reconfigure the network and to distribute the global signal ordering the reconfiguration. We

note that in the original publication of Pratt and Nguyen this technique was judged as being inappropriate for the analog implementation.

In this technique it is not necessary for the digital phase comparator to have a large dynamic range. It can be shown that even if the phase comparators have only one bit of resolution (if they detect only the sign of the phase), the unidirectional network still converges to a state with the phase errors close to zero. However, when the network switches to the bidirectional configuration, the phase comparator characteristic should be linear with positive slope at the phase error values less than $\Delta\phi_r$. Hence, the phase comparator characteristic may have a limited linear range as in Fig. 2.6(b), as far as the residual error obtained in unidirectional mode is within this range.

This technique is selected for the desired synchronization mode selection in our project. The validation of this method has been done by simulation and on a FPGA platform. Details of this implementation can be found in Section 5.2.

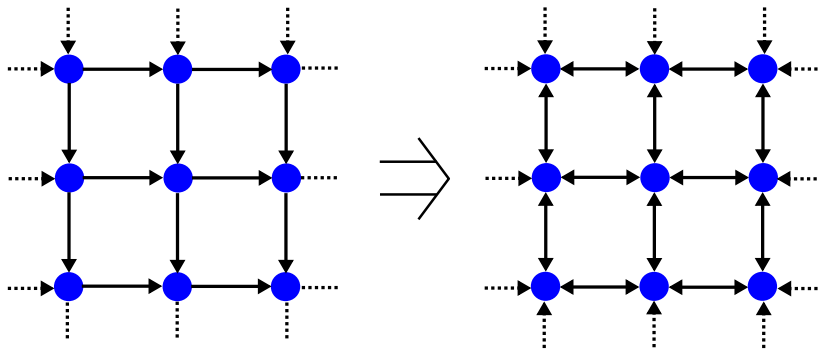


Figure 2.13: **Dynamic reconfiguration of the network from uni- to bidirectional**

2.4 Stability of the PLL networks

The stability issues of stand-alone PLLs have been well studied in literature. However, interconnecting several PLLs in a network increases the order of the system, making necessary a robust design procedure guaranteeing the global stability. In the context of the *HODISS* project, our colleagues from the CEA-LETI and AMPÈRE laboratories developed a mathematical tool based on the control theory, allowing to synthesize the node loop filter guaranteeing that (cf. [4, 34, 35, 36, 3]).

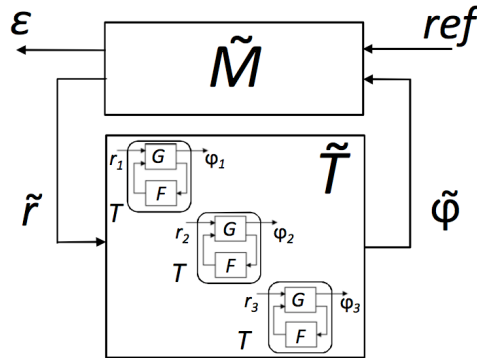


Figure 2.14: **Representation of the PLL network for stability study in [59]**

The design procedure is based on the decentralized H_∞ control technique making use of the dissipative properties of the system [59]. The system is first represented as a loop including a sub-system of a unidirectional chain of the network nodes \tilde{T} , and a sub-system \tilde{M} representing interconnections in the network (Fig. 2.14). *ref* represents the input phase of the reference, ε represents the phase errors between neighbors which should converge to zero. In this way, the procedure of filter synthesis is done in two steps: firstly, the local stability (that of the matrix \tilde{T} is ensured), then the global stability is guaranteed. The procedure provides to the implementation engineers the range of stable PI filter coefficient values.

2.5 Modeling of the clocking network

The ADPLL network is a complex non-linear high order system. By consequence, its modeling is of paramount importance at all stages of design. The system-level design requires a system-level modeling where each block is represented by its behavioral macromodel. As the design of the individual blocks progresses, more detailed description of the blocks must be used (e.g., gate or transistor level). Hence, the model must be built on an open platform able to integrate different levels of description. In our study, the AdvanceMS tool of Mentor Graphics was used for the modeling of the system at all levels: this tool allows a simultaneous use of VHDL, VHDL-AMS, Verilog and transistor-level Spice descriptions in a common simulation. As shown in [82], the VHDL description is particularly appropriate for the system-level description of the ADPLL network, outperforming Matlab Simulink-based approach in what concerns the precision and simulation time.

This section presents the methodology of behavioral modeling of a single ADPLL and of a network of ADPLLs. The presented model has been used for a validation of the theoretical studies carried out in the frame of the HODISS research project [4, 34, 35, 36, 3] and for quick virtual prototyping of the designed ADPLL network.

The studied ADPLL network is composed of three basic blocks: a phase comparator, a loop filter and a DCO. The VHDL behavioural models of these blocks are described in the three following subsections. Subsection 2.5.5 presents the simulation results of the ADPLL network.

2.5.1 The model of the phase comparator

The architecture of the modeled digital phase comparator is shown in Fig. 2.15. The sign detector determines the sign of the phase error between ref (reference clock) and div (feedback divided clock) periodic sequences of events. The events are represented by the rising edges of the signals ref and div . If ref events are leading it means that the phase error is positive, the output is '0'. If div events are leading it means that the phase error is negative, the output is '1'. A quantizer is used to measure the absolute phase error range and converts it into a non-signed code representing its value $|e_{ri}|$. The quantizer can be seen as a n -bit ADC. The arithmetic block combines the sign with the absolute value generating a 2-complement output word e_{ri} on $n + 1$ bits. The implemented transfer function is given in Fig. 2.6(b).

The behavior of the two blocks of the digital phase comparator is described in details in Section 4.2.

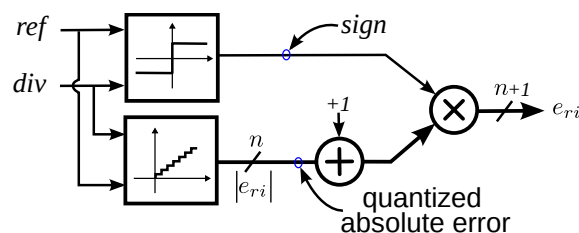


Figure 2.15: Block diagram of the modeled phase comparator

2.5.2 Loop filter

The VHDL model of the proportional-integral filter is implemented as a register transfer level (RTL) block whose structure is given in Eq. (2.3). The real-values signals and coefficients are defined using the fixed-point arithmetic.

2.5.3 DCO

The digitally-controlled oscillator is modeled as a digital period-controlled oscillator (Fig. 2.16). The oscillator generates already divided clock used for feedback div (Fig. 2.2(b), signal $CLK_{i,j}$). It has a linear code-period characteristic defined by:

$$T_{DCO} = \Delta T_{DCO}^{vhdl} 2^{m+1} - K, \quad (2.5)$$

where ΔT_{DCO}^{vhdl} is elementary tuning step which is chosen to be 3 ps, m is the width of the input binary code K . For better precision, This is not a fair model of a DCO with a linear code-frequency characteristic. However, as we mentioned in Subsection 2.2.3, the linearity of the DCO is not a key property for an ADPLL. A non-linear code-frequency characteristic means that the DCO gain is frequency-dependent. For the validation of an ADPLL architecture originally using a linear code-frequency DCO with a linear code-period model, it is enough to chose for the model an appropriate gain at the frequency of operation. If necessary, the implemented DCO model Eq. (2.5) can have a linear code-frequency characteristic if the offset of the DCO period is code-dependent.

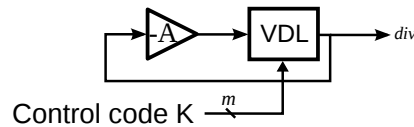


Figure 2.16: **Principle of the VHDL implementation of digital period-controlled oscillator**

2.5.4 Simulation of ADPLL

The three designed models were assembled so to obtain a model of a single ADPLL. Fig. 2.17 presents the evolution of the PFD output and the evolution of the period of the DCO oscillations in transient process. The conditions and parameters of the ADPLL blocks used for this simulation are summarized in Tab. 2.2.

At the starting time, the ref signal frequency is higher than the div . This can be seen from the phase comparator output: the output code is at its maximal (saturation) value equal to 15. Immediately after the start of the process, the frequency acquisition process takes place: the frequency difference between the output DCO signal and the input ref signal decreases, while the output of the phase comparator keeps a high positive value. When the frequencies of the signals are close, the phase acquisition starts: the output of the phase comparator

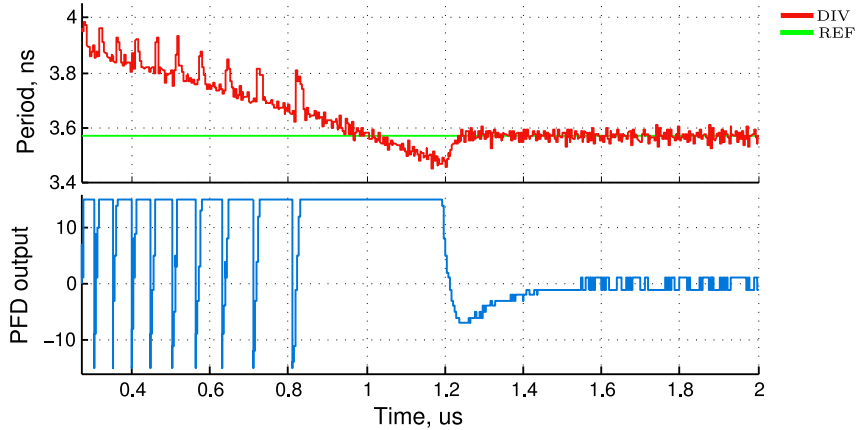


Figure 2.17: **Simulation of the developed VHDL model of ADPLL**

Table 2.2: **ADPLL simulation parameters and conditions summary**

<i>Parameter</i>	<i>Value</i>
Reference period T_{ref}	3.57 ns
Oscillator initial period T_{div}	4 ns
$\Delta T_{DCO,VHDL}$	3 ps
TDC resolution	32 ps
DCO input width M	10 bit
PFD output width N	5 bit
Filter coefficient α	1
Filter coefficient β	0.03

varies in a large range highlighting a transient process. At the end of the transient, the output of the phase comparator is ± 1 – what indicates that the real absolute phase error values are under the resolution of the absolute phase quantizer, i.e. 32 ps.

2.5.5 Simulation of network

In order to simulate the clocking network, we assembled the developed VHDL blocks in a 4×4 network with the topology depicted in Fig. 2.18 and the node structure shown in Fig. 2.2. The reference signal is coupled with the Node 1 (Fig. 2.18). The simulations were performed for different initial conditions. The filter coefficients used for this simulation are obtained from the theoretical study of [34, 35, 36]. They are summarized in Tab. 2.3 together with other parameters of the network model.

Three modeling experiments were done.

Verification of robustness to perturbation

The goal of the first experiment was to check the robustness of the steady-state synchronised mode to a perturbation. All DCOs are identical, and they all have identical starting frequencies. The starting frequencies of the DCOs are physically determined by the output of the

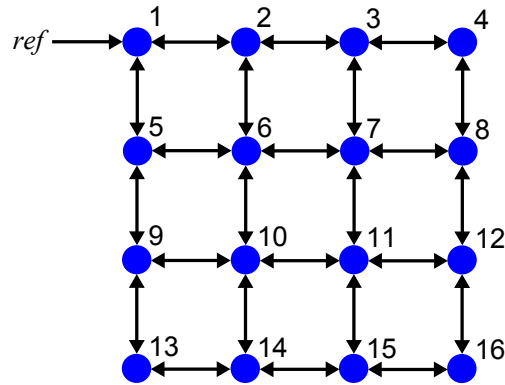


Figure 2.18: **Simulated 16-node clocking network**

control block at $t = 0$ and by the code-frequency characteristic of the DCOs. The starting frequencies of all DCOs are given by the period value $T_{div,1-16}=3.97$ ns. They are lower than the reference signal frequency given by $T_{ref}=3.76$ ns. It can be seen on Fig. 2.19 that the network converges to a synchronization after subsequently frequency and phase acquisition processes.

A perturbation has been introduced at the Node 11 at time $40 \mu\text{s}$. The local clock of the Node 11 has been forced to change its frequency (the new period is $T_{div,11}= 3.98$ ns). After 50 ns the Node 11 has been released and network started the compensation of this perturbation. From the figure it can be seen that this perturbation has negligible impact on frequencies of neighboring nodes ($<1\%$ of reference frequency). The nodes of the network which are not connected directly with this node have not been affected. The output errors of the PFDs associated with the Node 11 are depicted in Fig. 2.20.

Start-up with initial frequency dispersion

The goal of the second experiment is to study the situation in which the network DCOs start from different initial frequencies – for example, because of a dispersion of the DCO parameters due to the fabrication. In this experiment the DCO initial frequencies differ by up to $\pm 20\%$. Fig. 2.21 presents the results of the experiment. It can be seen that after a frequency acquisition phase, the nodes converge to the reference frequency so to come to a synchronous mode. In the locked state, the phase errors between the nodes are minimal and do not exceed ± 2 steps of the phase comparator. The observed behavior is in a good agreement with what was expected by the theoretical investigations and with the results of the Matlab simulations performed by our colleagues and published in [36].

Verification of dynamic reconfiguration

The goal of the third experiment is to test the technique of dynamic on-fly selection of the desired synchronization mode. As in the second experiment, the initial frequency of the DCOs are randomly distributed over $\pm 5\%$ range, and the initial conditions of the network are

Table 2.3: Clocking network simulation parameters and conditions summary

<i>Parameter</i>	<i>Value</i>
Reference period T_{ref}	3.76 ns
Oscillator initial period T_{div}^*	3.97 ns
$\Delta T_{DCO,VHDL}$	3 ps
TDC resolution	32 ps
DCO input width M	10 bit
PFD output width N	5 bit
Filter coefficient α	1
Filter coefficient β	0.00243

*For first experiment

chosen so to obtain undesirable synchronization mode if the network is in the bidirectional configuration.

Upper 4 plots of Fig. 2.22 present the behavior of the network which starts in unidirectional mode, then switches to the bidirectional mode: all phase errors converge to values in range ± 2 .

Comparison of robustness to perturbation

The goal of the fourth experiment is to demonstrate that in the bidirectional mode the network has a better immunity toward perturbations than when the network is in the unidirectional mode. For this, a perturbation is injected in the Node 5, by changing the initial frequency of its DCO by 3 % at $t=6.3 \mu s$. The phase error perturbation observable on PFD 5–6 output plot reduces as the distance from the perturbed node increases, and is nearly non-observable between Node 11 and Node 15 (PFD 11–15 output). The lower plot presents the error PFD 11–15 output for a unidirectional system with the same perturbation: the perturbation is well observable. This prove that the bidirectional mode of coupling ensure better immunity to noise and perturbations.

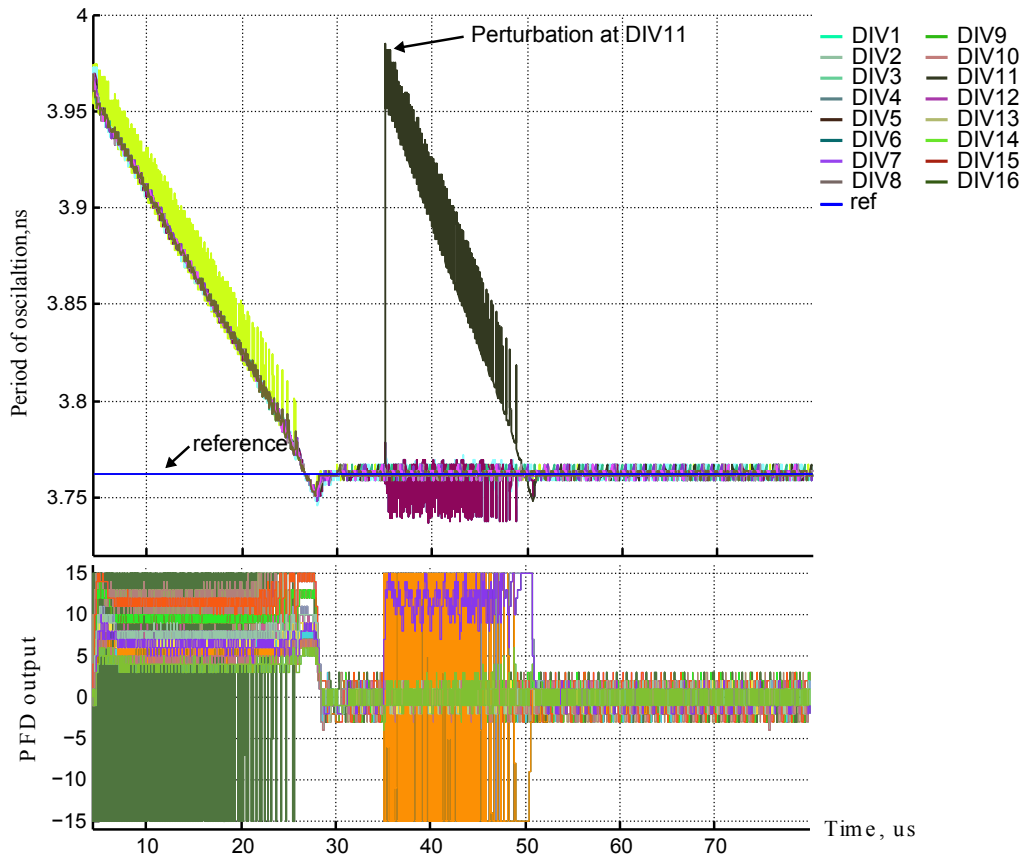


Figure 2.19: **Simulation of the developed VHDL model of 16-node network:** showing network locking process with equal local initial frequencies; upper graph is period of clock signals and lower is output of PFDs

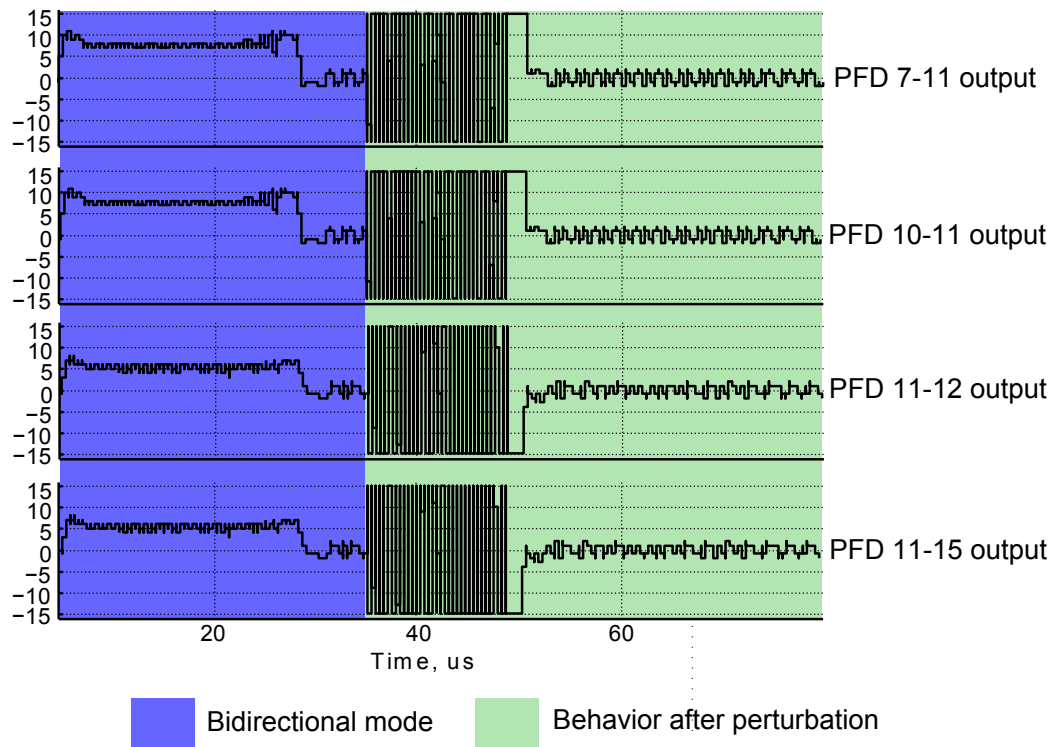


Figure 2.20: **Errors of the PFDs connected to the node 11 in first experiment**

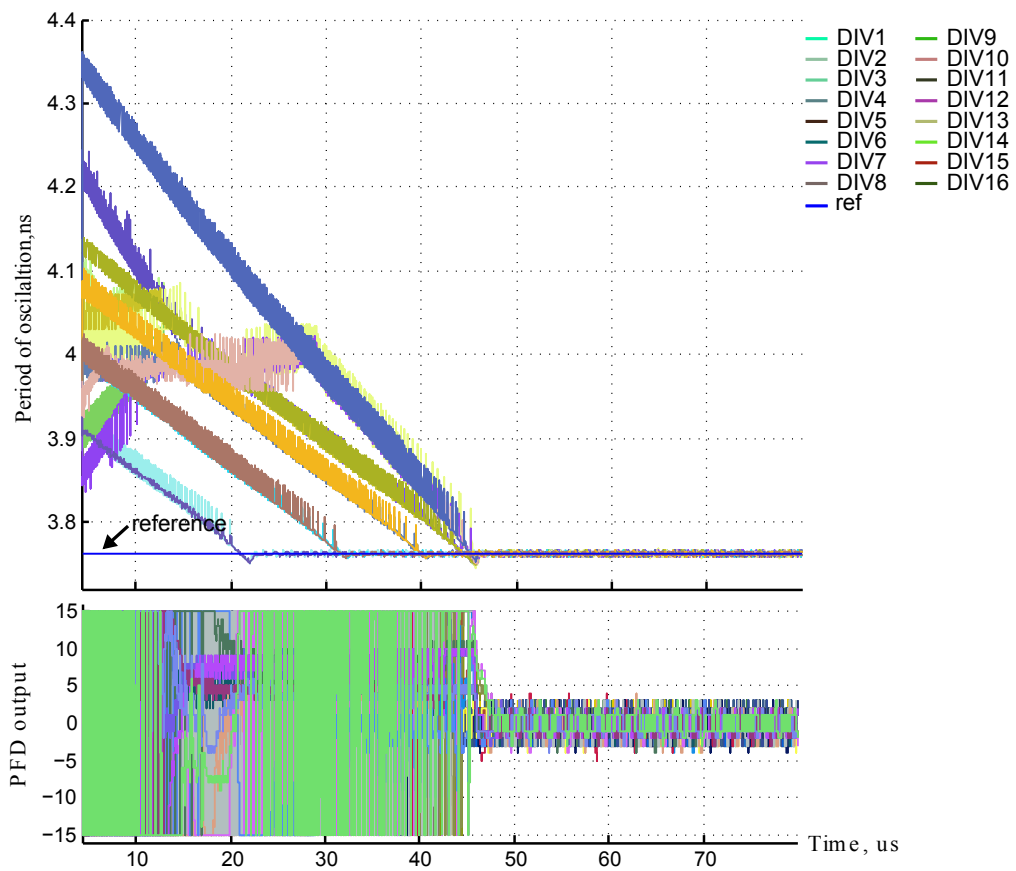


Figure 2.21: **Simulation of the developed VHDL model of 16-node network:** network locking process with different local initial frequencies (second experiment); upper graph is period of clock signals and lower is output of PFDs

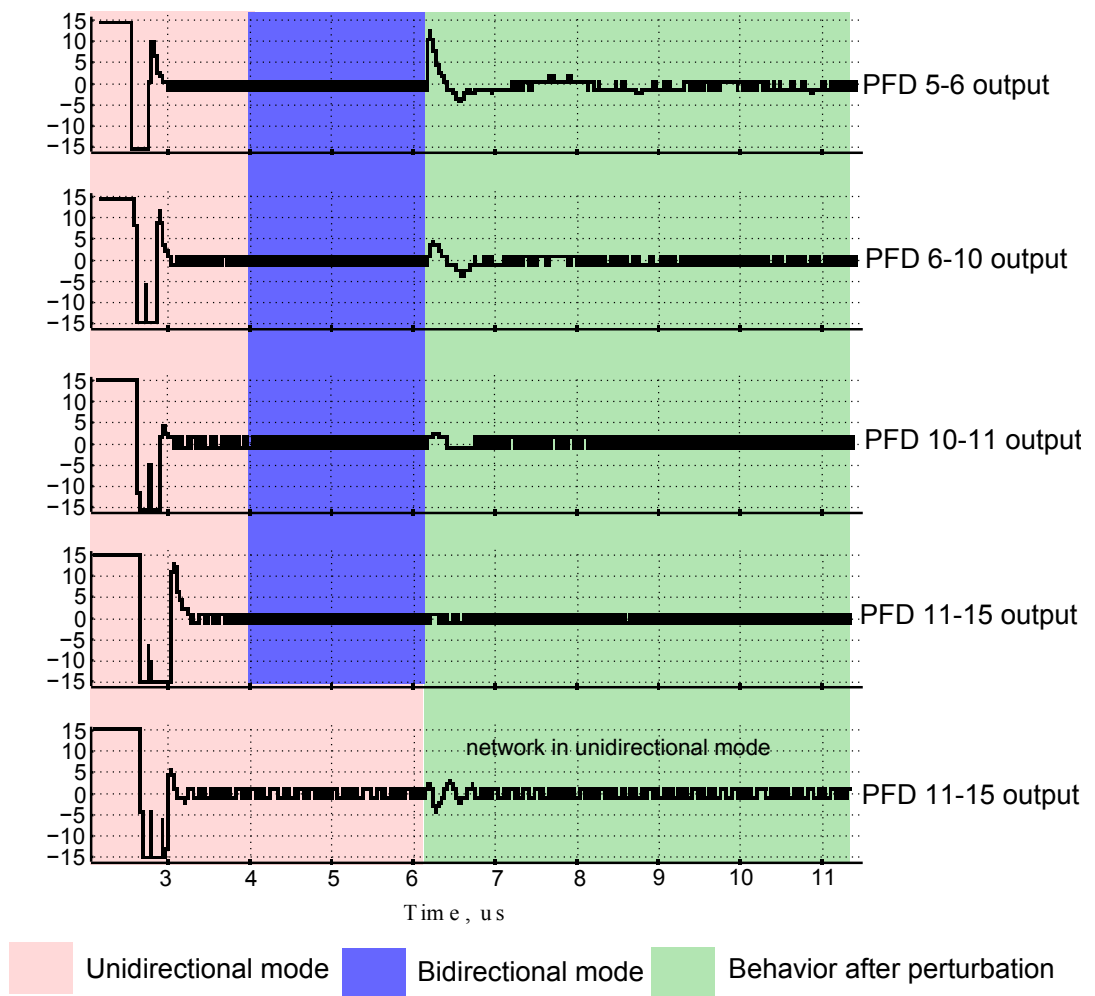


Figure 2.22: **Simulation results of dynamically reconfigurable network.**

2.6 Conclusion

In this chapter we have introduced the principle of the proposed clocking approach based on multioscillator architecture. We have demonstrated the basics of such a clocking principle and highlighted its major advantages. As a resume, the strong features of such a clocking system are:

- The local clock source allows designers to avoid the chip-length global high frequency clock distribution system;
- Less buffers and amplifiers, which practically are sources of skew, accumulative jitter and high power consumption;
- In a context of jitter, PLL behaves like a natural low-pass filter. Hence, it can attenuate the jitter and locally regenerates the clock from potentially noisy reference;
- In a steady state, proposed network is considered as a single source of clock.

The previous work [24] has justified the feasibility of this approach. Inspired by this work, we have proposed another implementation, which is based on a fully digital frequency synthesis technique using an ADPLL. This technique simplifies the integration of the system into SoC environment and adds extra degrees of freedom, such as reconfigurability and portability.

We have introduced and explained the nature of the mode-locking phenomenon which is the most significant problem of such a clocking networks. To overcome it, we have introduced and discussed three methods of the desirable synchronous mode selection.

The clocking network has been modeled at a system-level through VHDL simulations. They implied behavioral models providing efficient instrument for the verification of the theoretical studies, test of the proposed mode-locking elimination techniques and estimation of the parameters for the future implementation. The results of this work were partially published and presented on the international conferences ISCAS2011 and NEWCAS2010 [82, 30].

Finally, we have derived specifications for the implementation of clocking network, which are used as input data for design of each block. They are introduced and detailed in a following chapters.

Chapter 3

Digitally controlled oscillator design

Contents

3.1	Introduction	39
3.2	Digital frequency tuning in ring oscillators	42
3.3	DCO architecture	52
3.4	Sizing of the DCO core and tuning cells	57
3.5	VHDL modeling of the oscillator	58
3.6	DCO layout desing I: the DCO floorplan	61
3.7	DCO layout design II: cell design	65
3.8	DCO chip test	76
3.9	Conclusion	80

3.1 Introduction

This chapter presents the design of the DCO for the distributed clock generator. From the point of view of the power consumption and silicon area, the DCO is, by far, the most critical block of the system. Moreover, its non-idealities have direct negative impact on the performances of the overall ADPLL network. The custom design approach used for the DCO layout design is motivated by these considerations. It also explains the fact that the DCO study and design took more than a half of the time available for the PhD project.

Oscillating circuits have a very important place in microelectronics: they generate time-varying signals required in virtually all electronic systems. There is a huge amount of literature about them: their theory, design, implementation for different applications and contexts. Therefore, it is natural, that the first task is to chose the class and the architecture of the DCO.

The PLLs described in the literature utilize three kinds of oscillator, depending on the frequency range and the phase noise requirement: LC-based oscillators for high-frequency

low noise RF applications, ring oscillator for intermediate frequency precision, and fully-digital oscillators for low frequency digital systems. For this project, the ring-based oscillator is a natural choice for the following three reasons:

1. The phase noise requirements are not as stringent in clocking as in RF systems;
2. Ring oscillators don't contain inductors which occupy large silicon area;
3. Ring oscillators have a regular structure well suitable for design automation by digital CAD tools.

A DCO can be considered as a DAC followed by a VCO (Fig. 3.1). For digital phase synthesis applications, one of the most critical properties of the DCO is the monotonicity of its code-frequency characteristic. This monotonicity is required so to ensure the negative sign of feedback in the DCO frequency range, which is a necessary condition for the ADPLL stability. The monotonicity requirement is the main criterion of choice of the DAC architecture.

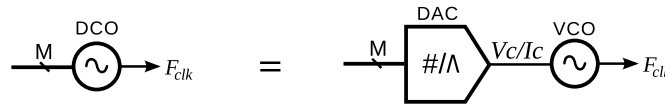


Figure 3.1: **Representation of DCO as a combination of DAC and VCO**

The oscillator presented here is a CMOS DCO with frequency range 999-2480 MHz and 1024 equal frequency steps within it, with highly linear and monotonous code-frequency characteristic. The DCO ring includes 7 inverting stages. The frequency control is achieved through a *width modulation* technique [67, 31, 45] using an array of tuning CMOS inverters. An original control scheme employing a mixed thermometer/weighted encoding provides a high frequency precision and monotonicity of the DCO. This technique is a result of compromise between the DCO precision, ensured in a best way with thermometer coding, and the DCO resolution and area economy improved by a weighted control scheme. In addition, we present here a frequency divider, which will produce all necessary clock signals for the digital filter, the local clocking zone and feedback.

The DCO layout is a regular matrix including many identical cells. Hence, the design of the oscillator cells is custom so to optimize their area and electrical characteristics. The cells are assembled by abutment of repeating layout blocks. The digital parts of the DCO necessary for the interface with the ADPLL control blocks are designed through a standard digital synthesis method.

The chapter is organized as follows. Section 3.2 presents the principles of the digital tuning in the CMOS ring oscillators. The frequency tuning in a 2D array of inverters is explained and analytic calculation on the frequency-code relation is proposed. Section 3.3 describes the structure of the developed oscillator. In Section 3.4, the key parameters of the oscillator are calculated (e.g. tuning range, frequency step) and used for the sizing procedures. Section 3.6 presents the details on implementation of the oscillator in a 65nm CMOS

technology. Finally, the results of the measurements of the DCO chip test prototype are presented in Section 3.8.

3.2 Digital frequency tuning in ring oscillators

In this section we review the ring oscillator architecture, determine analytically its main parameters and discuss the implementation of the digital frequency tuning.

A basic CMOS ring oscillator is composed of an odd number of CMOS inverters (Fig. 3.2) connected in a ring chain. It can easily be proven that such a structure produces steady-state oscillations with frequency given by Eq. (3.1):

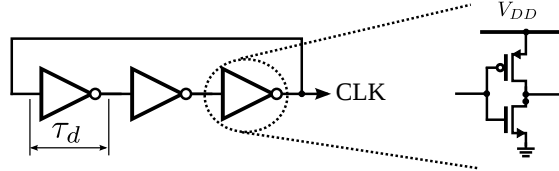


Figure 3.2: **Inverter-based ring oscillator:** three stage single-ended configuration

$$F_{osc} = \frac{1}{2 \times \sum_{i=1}^N \tau_{d_i}} \quad (3.1)$$

where τ_{d_i} is the delay of the i^{th} stage of the ring and N is the (odd) number of the inverter in the chain. τ_{d_i} is the natural delay of the each stage of the oscillator caused by a parasitic capacitances of the CMOS transistors and interconnections (C_{int}) between them (Fig. 3.3(a)). An exact expression for the CMOS inverter delay is difficult to obtain, because of complex nonlinear dynamic behavior of the MOS transistors. However, an approximation is possible by making two simplifying assumptions :

1. The transistor parasitic capacitances are constant;
2. During the transition up to the threshold voltage ($V_{dd}/2$ for simplicity), the transistor produces a constant current equal to the current in saturation mode with $V_{gs} = V_{dd}$.

Taking into account these assumptions, the equivalent circuit of the single ended stage of oscillator can be represented as a two ideal MOS transistors driving a lumped load capacitance C_l . The load capacitance C_l is the sum of the transistor parasitic capacitances and the interconnection capacitances. These MOS transistors can be represented as current sources $I_{s,p}/I_{s,n}$ (Fig. 3.4).

According to this simplified model of an inverter, we can calculate the propagation delay times of transition from low to high level τ_d^{LH} and from high to low τ_d^{HL} (defined as charge and discharge time of the load capacitor respectively from 0 to $V_{dd}/2$ and from V_{dd} to $V_{dd}/2$):

$$\begin{aligned} \tau_d^{LH} &= \frac{1}{2} \frac{C_l}{I_{s,p}} V_{DD} \\ \tau_d^{HL} &= \frac{1}{2} \frac{C_l}{I_{s,n}} V_{DD} \end{aligned} \quad (3.2)$$

The currents $I_{s,p}$ and $I_{s,n}$ define the symmetry of the output signal of oscillator, and are generally wanted to be identical, so to achieve identical delays τ_d^{LH} and τ_d^{HL} .

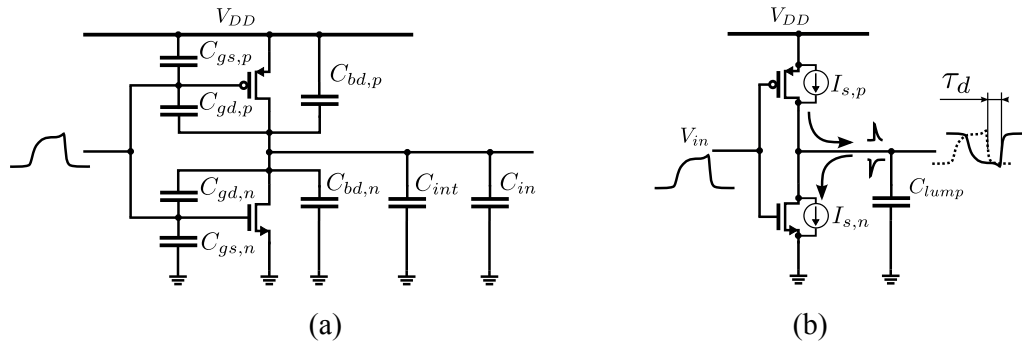


Figure 3.3: **CMOS inverter with parasitic components:** (a) complete and (b) reduced schematic

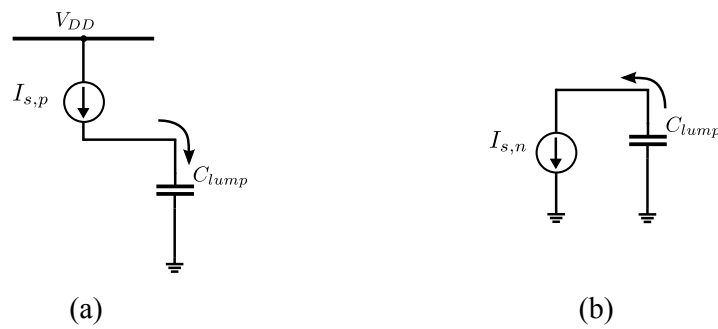


Figure 3.4: **Charge and discharge of parasitic capacitance in CMOS inverter:** (a) charge by a current source and (b) discharge

When a frequency control is needed, all constituent parts of the equations 3.2 can vary: the load capacitance, the supply current and the supply voltage. If the frequency control must be digital as in DCO, a use of DAC is necessary so to generate discrete values of these parameters.

3.2.1 Capacitive tuning

The capacitive tuning is achieved by varactors controlled by a voltage, where voltage V_c is generated by a DAC (Fig. 3.5(a)). To implement a voltage-controlled capacitor (a varactor), a reverse-biased PN junction of a MOS capacitance can be used. However, the achievable tuning range is relatively low (no more than 30 % [75, 19, 80, 38]), and the parasitic noise on the control line is critical.

A direct discrete frequency tuning may use banks of fixed capacitors whose overall capacitance is controlled by reconfigurable connection network (Fig. 3.5(b)). Such a technique is widely used in ring oscillators [31, 38] as well as in LC-tank oscillators [75, 19, 80, 76]. However, such an implementation remains extensively analog, and therefore, not suitable for our application.

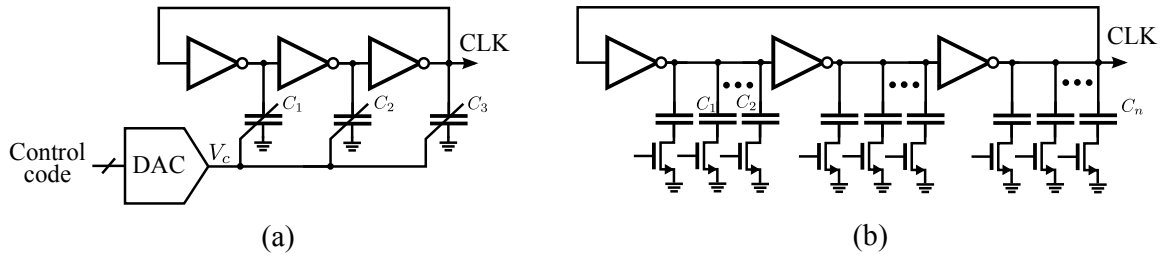


Figure 3.5: **Capacitive tuning in a ring oscillator:** (a) with DAC and (b) direct digital control

3.2.2 Current/Voltage tuning

Varying the driving current I_s or supply voltage V_{dd} and hence, the inverter delay, the frequency can be tuned in a wide range. This technique uses two blocks: a DAC with either a current or a voltage output followed by a conventional VCO [37].

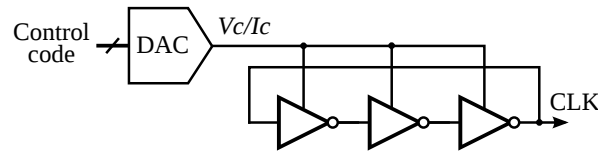


Figure 3.6: **Current tuning in a ring oscillator**

The principal weakness of this technique is the nonlinearity of the voltage/current tuning. Also, the voltage nodes are very sensitive to the ambient noise. The latter is strong in digital environment and thus, noise performance of the oscillator can degrade significantly [2].

The DAC needed to generate a discrete voltage at the VCO input may have different architectures. As we said, in the context of the digital phase synthesis, the main requirement for the DCO DAC is its monotonicity. In order to fulfill the specification of our project, the DAC should be able to process the input control code at few hundred megahertz rate with 10 bits resolution. The implementation of such a DAC is not a trivial task.

3.2.3 Current tuning with width modulation technique

Recent publications [81, 48, 45, 44] demonstrated that it is possible to integrate the digital control into the oscillator core structure.

The principle is shown in Fig. 3.7(a). The tri-state inverters are connected in parallel to a stage of the main inverters ring. These supplementary inverters can provide additional discrete portions of charge/discharge current to the load capacitance of the stage. These tri-state inverters are called tuning inverters. They controlled directly by digital binary signals. It is important to notice, that we consider the load capacitances of the stages are constant over all circuit operation time.

In order to equalize the on and off capacitances of the tri-state inverters, the latter are implemented as in Fig. 3.7(c): the switches are put near the supply rails, rather than near

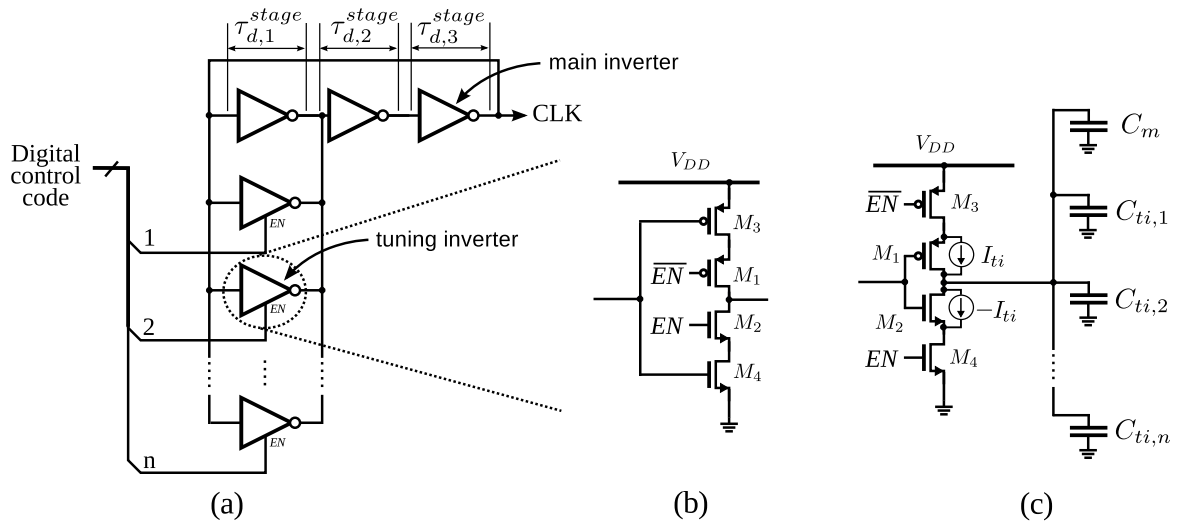


Figure 3.7: **Current tuning technique with direct digital control:** (a) circuit diagram of n step tuning stage, (b) schematic of the conventional tri-state inverter and (c) schematic of the proposed tri-state inverter

the output signal rail, as in conventional tri-state inverters (Fig. 3.7(b)). Indeed, the switching transistors in a conventional tri-state inverters not only cut the current paths but also disconnect the parasitic capacitance of the active transistors.

We propose a simple analysis allowing a calculation of the oscillation frequency of the DCO in Fig. 3.7.

As aforementioned, the load capacitance of each stage is considered constant for all input control codes. The capacitance of tuning inverter C_{ti} remains constant in all regions of operation. Consequently, a total load capacitance C_l of the stage composes of

$$C_l = C_m + \sum_{j=1}^Y C_{ti} \quad (3.3)$$

where C_m is a lumped capacitance of the main inverter and Y is a total amount of tri-state inverters in stage.

Fig. 3.8 illustrates the variation of stage delay τ_d of stage in the ring oscillator. The total current contributed by the main inverter and y active tuning inverters is defined as:

$$I_s = I_m + KI_{ti} \quad (3.4)$$

where I_m is a current contributed by main inverter; I_{ti} is a current contributed by each tuning inverter; K is the digital integer control code, which determines the number of active tuning inverters in a stage. Referring to the Eq. (3.2) and Eq. (3.4), we can calculate the dependence of stage delay from the number of the active three-state inverters K :

$$\tau_{d,1} = \eta \frac{C_l V_{DD}}{I_m + KI_{ti}} \quad (3.5)$$

Now, from Eq. (3.1) and (3.5) we can express the oscillation frequency of the oscillator composed of N stages (Fig. 3.7,(a)):

$$F_{osc} = \frac{1}{2(\sum_{i=2}^N \tau_{d,i}^{stage} + \eta \frac{C_l V_{DD}}{I_m + yI_{ti}})} \quad (3.6)$$

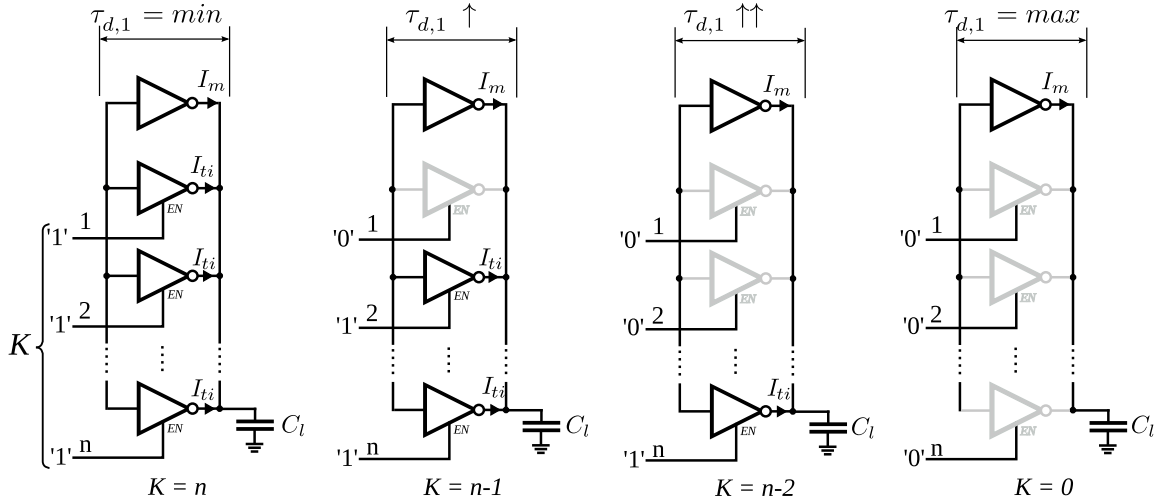


Figure 3.8: **Digital delay control principle:** control code varies from maximal to minimal value

The relation between the oscillation frequency and the number of the active tri-state inverters is nonlinear. However, it can be seen that if KI_{ti} is small compared to I_m for all K , the relation is close to be linear. This property is used to design a DCO with highly linear code-frequency characteristic, as presented in the next subsection.

3.2.4 Frequency tuning in 2D array oscillator

From practical point of view, if the number of the tuning cells in stage Y is large, they should be distributed over all stages of the oscillator. Naturally, a two dimensional array DCO is obtained like in Fig. 3.9. Such a topology is used in most similar implementations [48, 45].

In this subsection we present the calculation of the output frequency for the DCO with identical tuning cells distributed over all stages. We demonstrate a possibility of a linear output frequency control.

For the analysis we made the following assumptions, which all can be ensured in practice by appropriate design of the DCO:

- number of tuning cells in all stages is the same
- total capacitances of stages are equal and constant
- difference in number of active tuning cells between stages does not exceed 1.
- tuning current $I_{ti} \ll I_m$, so to obtain a linear frequency tuning (cf. Eq. (3.6))

Therefore, according to the Eq. (3.2) and Eq. (3.3) the total delay of the stages in oscillator is defined as

$$\tau_{osc}^{2D} = \sum_{i=1}^N \eta \frac{C_l}{I_m + yI_{ti} + x_i I_{ti}} V_{dd} \quad (3.7)$$

where N is a number of stages, y is number of active tuning cells in each stage (equal for all stages), x_i represents each cell in a partially filled row (between 0 and $N - 1$) (see Fig. 3.9).

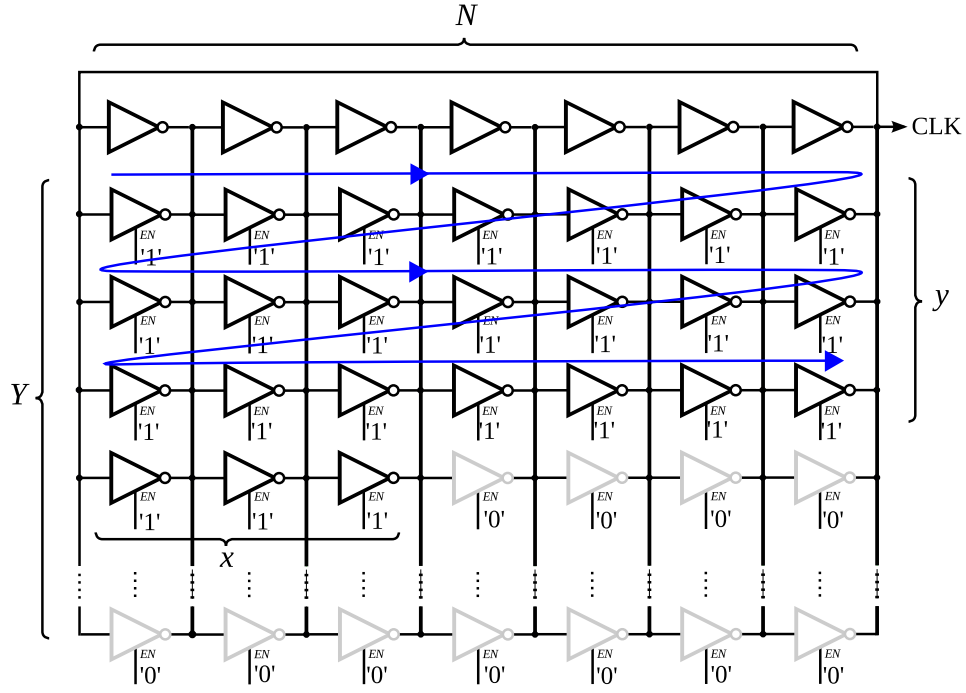


Figure 3.9: **Frequency tuning in 2D array oscillator:** 7 stage example with Y as amount of tuning cells in each stage, y is amount of active cells in a stage and x is amount of active cells in an unfilled row

Two integer variables y and x_i together are defined by the number of activated tuning cells K (i.e. integer value of the digital frequency control code). They relates as:

$$K = yN + \underbrace{\sum_{i=1}^N x_i}_x \quad (3.8)$$

Assuming that $\eta C_l V_{dd}$ is a constant parameter, we replace it with symbol P , hence, simplifying Eq. (3.7))

$$\begin{aligned} \tau_{osc}^{2D} &= \sum_{i=1}^N \frac{P}{I_m + yI_{ti} + x_i \cdot I_{ti}} \\ &= \frac{P}{I_m + yI_{ti}} \sum_{i=1}^N \frac{1}{1 + \frac{x_i I_{ti}}{I_m + yI_{ti}}} \end{aligned}$$

since $\frac{1}{1-\alpha} \approx (1-\alpha)$ when $\alpha \ll 1$ and since $\frac{x_i I_{ti}}{I_m + yI_{ti}} \ll 1$, than period of oscillation will be

$$\begin{aligned} T_{osc}^{2D} &= \frac{2P}{I_m + yI_{ti}} \sum_{i=1}^N \left[1 - \frac{x_i I_{ti}}{I_m + yI_{ti}} \right] \\ &= \frac{2P}{I_m + yI_{ti}} \left[N - \frac{x I_{ti}}{I_m + yI_{ti}} \right] \end{aligned} \quad (3.9)$$

Now we can calculate the output frequency according to the control code K :

$$\begin{aligned}
F_{osc}^{2D} &= \frac{1}{2\tau_{osc}^{2D}} = \frac{I_m + yNI_{ti}}{2NP} \left(1 + \frac{xI_{ti}}{I_m + yNI_{ti}} \frac{1}{N} \right) = \\
&= \frac{1}{2NP} \left(I_m + yI_{ti} + xI_{ti} \frac{1}{N} \right) = \\
&= \frac{1}{2NP} \left(I_m + I_{ti} \left[y + x \frac{1}{N} \right] \right) = \\
&= \frac{1}{2NP} \left(I_m + \frac{I_{ti}}{N} \underbrace{\left[yN + x \right]}_K \right) = \\
&= \frac{1}{2NP} \left(I_m + \frac{KI_{ti}}{N} \right) = \\
&= \underbrace{\frac{I_m}{2NP}}_{F_0} + \underbrace{\frac{KI_{ti}}{2N^2P}}_{\Delta F} \tag{3.10}
\end{aligned}$$

From this equation it is clearly shown that frequency of the 2D arrayed oscillator F_{osc}^{2D} is defined by two parameters: F_0 which is the minimal oscillation frequency of oscillator and the tunable increment ΔF defined by the number of activated cells K .

3.2.5 The choice of the coding

The DCO oscillation frequency is defined through a digital code K applied to its input (Fig. 3.1). In most cases, the DCO input code is a binary unsigned word. The encoding mechanism allows to define, for each value of the input code, a unique value of output frequency, so to obtain the desirable code-frequency characteristic. This definition is done through the choice of the size of the tuning cells and through a definition of the relation between each input code K and the set of the active tuning cells corresponding to this code.

Thermometer coding

The width-modulated DCO architectures using identically sized tuning cells (Fig. 3.9) are naturally controlled by a thermometer code. The thermometer code is composed of all control bits of all tuning cells; the first K bits of this word are one. An unique relation between K and the parameters y and $x_i|_{\{x=1\dots N\}}$ is easily defined.

The thermometer coding is a redundant coding technique based on a sum of equal elementary quantities weighted by 0 or 1:

$$I_{s,therm} = I_{ti}t_0 + I_{ti}t_1 + \dots + I_{ti}t_n = I_{ti} \sum_{i=0}^n t_i, \tag{3.11}$$

where $I_{s,therm}$ is the output value, I_{ti} is the value of an elementary quantity, t_i are the weight of the i^{th} term of the sum, $t_i \in \{0, 1\}$. An example of a thermometer coding is shown in the Fig. 3.10.

The thermometer coding guarantees a monotonicity of the code-frequency characteristic. The drawback of the thermometer code for a DAC is the complexity and the size of the

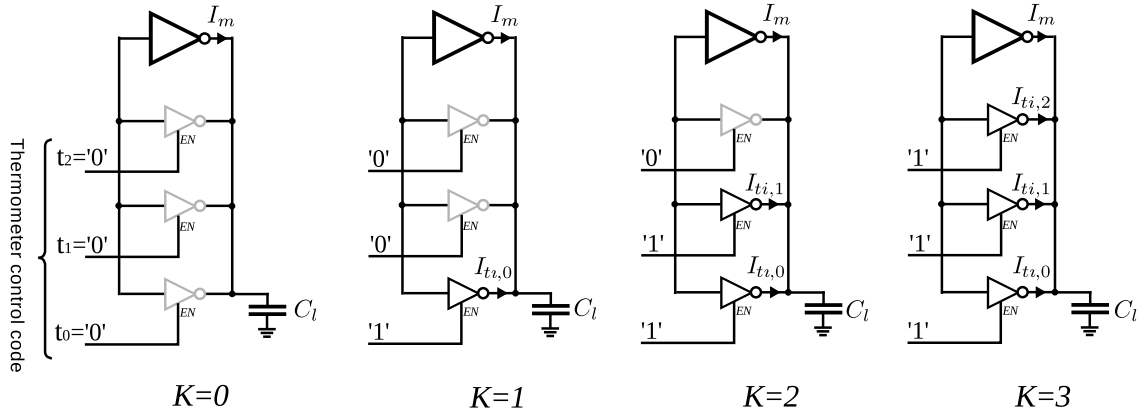


Figure 3.10: **Thermometer current coding:** example with four frequency steps

architecture: the number of the tuning cells is equal to the number of the frequency steps. If the latter is large, the size of the DCO may be prohibitive. Also, a binary-to-thermometer encoder is necessary since the input control word K is binary coded in most cases. Its complexity increases when the number of tuning cells increases.

Binary coding

In the binary coding scheme the currents provided by each of $m + 1$ tuning inverter are in binary weighted relation :

$$I_{ti}^l = I_{ti} 2^l, \quad (3.12)$$

where $l \in 0, 1, \dots, m - 1$ is the index of the stage, I_{ti} is the current generated by each tuning stage, I_{ti} is the current generated by the smallest tuning stage. This is obtained by appropriate sizing of the active transistors in tuning inverters.

The overall tuning current is a sum of all weighted components

$$I_{s,bin} = I_{ti} b_0 2^0 + I_{ti} b_1 2^1 + \dots + I_{ti} b_{m-1} 2^{m-1} = \sum_{i=0}^{m-1} I_{ti} b_i 2^i \quad (3.13)$$

where $b_i \in \{0, 1\}$ is the value of the control bit of the i^{th} tuning cell. The schematic representation of such a coding technique is shown in Fig. 3.11.

The advantage of the binary coding is the simplicity of implementation: a large number of tuning steps can be obtained with small number of the tuning elements. However, for a large number of bits the fabrication errors become a critical issue. For instance, if in a 10 bits DCO based on binary coding the input word decreases from "10 0000 0000" to "01 1111 1111", the frequency may increase because of the fabrication mismatches in tuning cells, and the monotonicity of the code-frequency characteristic may be locally violated.

Weighted-thermometer coding

The mixed weighted-thermometer encoding scheme proposed in this work takes benefits from the binary and thermometer encoding techniques.

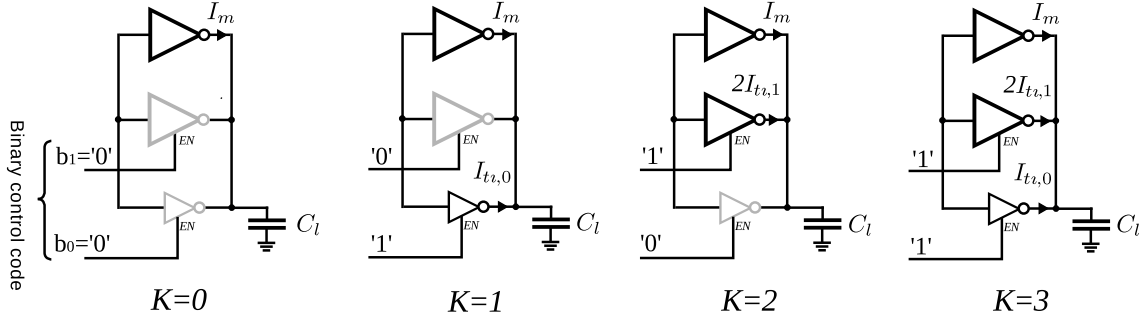


Figure 3.11: **Binary current coding:** example with four frequency steps

The binary coding based DAC is reliable when the number of bits is under 6-7. The loss of precision is related to the large difference between the sizes of low weight and high weight elements (in our case, the size of the tuning cells). A combination of thermometer and binary coding is frequently used. In particular, the coarse tuning is implemented with thermometer coding, and the fine tuning is implemented with binary coding. In that way, the synthesized analog quantity X is given by:

$$X = X_0 + x_{min} \left(\underbrace{2^{L+1} \sum_{i=1}^M t_i}_{\text{Thermometer}} + \underbrace{\sum_{i=0}^{L-1} b_i 2^i}_{\text{Binary}} \right), \quad (3.14)$$

where X_0 is the bias (initial) value of the synthesized quantity, X_{min} is the minimal incremental step, t_i are the bits of the thermometer coded fraction of the control word (M bits), b_i are the bits of the binary coded fraction (L bits). Note that the thermometer coded part of the control word has M bits and encodes $M + 1$ values from 0 to M . The bits are numbered starting from 1. The bits of the binary encoded part of the control word are encoded starting from 0.

The coding we used for the DCO implementation is inspired from this mixed techniques. The coarse tuning is achieved through a M bits thermometer code encoding $M + 1$ frequency values, as shown in Fig. 3.10. For each coarse step, L tuning steps are allowed thanks to $L - 1$ identical fine tuning cells whose size relates to the coarse tuning cells size as $1/L$. The $L - 1$ fine tuning cells are connected in parallel with the stages of the ring oscillator, are distributed as regularly as possible between them. The fine tuning cells are activated with a thermometer code. The overall DAC operates as shows the equation:

$$X = X_0 + x_{min} \left(\underbrace{L \sum_{i=1}^M a_i^c}_{\text{Coarse tuning}} + \underbrace{\sum_{i=1}^{L-1} a_i^f}_{\text{Fine tuning}} \right), \quad (3.15)$$

where a_i^f and a_i^c are coarse and fine tuning cells control code bits respectively.

Such a DAC provides $(M + 1)L$ discrete values, with only $M + L - 1$ tuning cells.

This principle is different from the mixed thermometer-binary coding, since the fine tuning cells are not sized with a binary ratio. We call this coding *weighted-thermometer coding*.

Such a DAC requires a conversion of the input binary word into the coarse and tuning thermometer coded word. Practical implementation of this approach is presented in the next sections.

3.3 DCO architecture

3.3.1 Oscillator topology

The proposed architecture is based on a 7-stage ring oscillator with parallel connection of tuning inverters to each stage of oscillator. The code-frequency characteristic is defined using the weighted thermometer coding presented in the previous section.

In this structure, each stage $ST_0 - ST_6$ of the oscillator contains one main inverter $MI_0 - MI_6$ which is always active (Fig. 3.12. $MI_0 - MI_6$). The tuning inverters are distributed over the 7 stages of oscillator on 39 rows and are organized in three arrays.

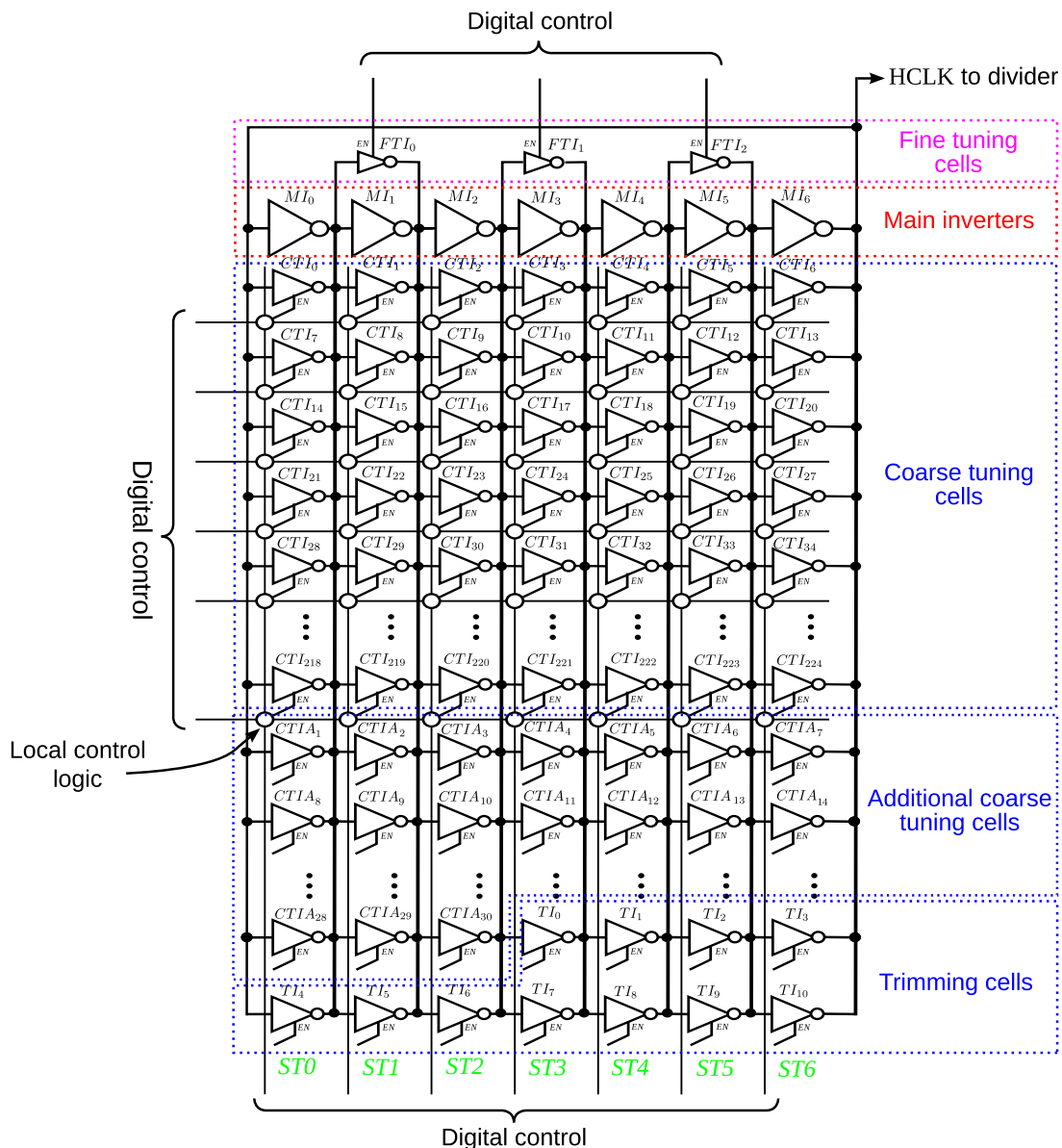


Figure 3.12: Core of the proposed oscillator: MI - main inverters, CTI - coarse-tuning inverters, FTI - fine tuning inverters

The first array consists of 224 (7 stages \times 32 rows) Coarse-Tuning Inverters (CTI). These cells (Fig. 3.12, $CTI_0 - CTI_{223}$) provide a coarse tuning of the output frequency.

The second array consists of one bank of 42 identical tuning cells (7 stages \times 6 rows). They have the same dimensions as the CTIs and provide the same tuning step. 31 of them are used as Additional Coarse-Tuning Inverters (CTIA) which implement the virtual extension of number of ring stages (cf. Subsection 3.3.2 for details). The 11 remaining cells are used for the frequency trimming. Totaling 266 identical elements, CTI, CTIA and trimming cells provide $2^8 - 1$ equal frequency steps (2^8 frequency values) and 11 additional frequency steps for the frequency trimming. Both, CTI and CTIA cells are controlled by their own logical block locally and by a grid of horizontal and vertical control lines globally.

The third array is implemented with three identical fine-tuning inverters (FTI) (Fig. 3.12, $FTI_0 - FTI_2$), which are connected to stages $ST1$, $ST3$ and $ST5$ of oscillator. These cells provide 3 fine frequency tuning steps for each coarse tuning step of the DCO. Together with coarse-tuning inverters, these fine-tuning inverters allow 1024 DCO frequency values (1023 tuning steps). The obtained thermometer weighted coding described by Eq. (3.15) have the following parameter values: $M = 2^8$, $L = 2^2$.

The size of transistors in FTI cells is defined by the dimensions of transistors in CTI/CTIA divided by the number of the fine tuning steps corresponding to each coarse tuning step (size ratio 1:4). The number of fine tuning steps L must be a power of two for control simplicity, and set to 4 as a compromise between the precision (monotonicity may degrade when the number of fine tuning steps increases) and the obtained gain in the DCO overall number of frequency steps. In the 10 bit input control binary code, the coarse and fine frequency tuning correspond to the 8 MSB and to the 2 LSB frequency variation respectively.

3.3.2 Control algorithm

The DCO control defines the mapping between the input 10 bit binary encoded word and the enable signals of all tuning cells. This mapping is physically implemented by the DCO control circuitry. Since the tuning cells are distributed in a two-dimensional array, it is natural to control them through row and column signals. Hence, the control map is a composition of two maps: the first, from the input 10 bit word to the column/row signals, the second from the column/row signals to the enable inputs of the tuning cells. The design of the DCO control circuitry should minimize the delay and the power associated with the control circuitry.

The most advantageous situation for the row/column signal generation is when the numbers of rows and columns in the tuning array of oscillator are equal to a power of two. In this case the row/column signals are obtained by a set of parallel binary-to-thermometer (B2T) decoders whose implementation is trivial. However, the single-ended CMOS inverter based ring oscillator architecture requires an odd number of stages: since the tuning cells are distributed over all stages, there is an odd number of columns, and the row/column signal generation requires a complex logical arithmetics.

To circumvent this difficulty related to the odd number of the inverting delays, we proposed an original control scheme based on a *virtual extension* of number of stages to a power of 2. Its principle can be observed in Fig. 3.13. In the 7-stage oscillator, the tuning cells of the virtual 8th stage are placed in the additional coarse tuning bank in which the tuning cells

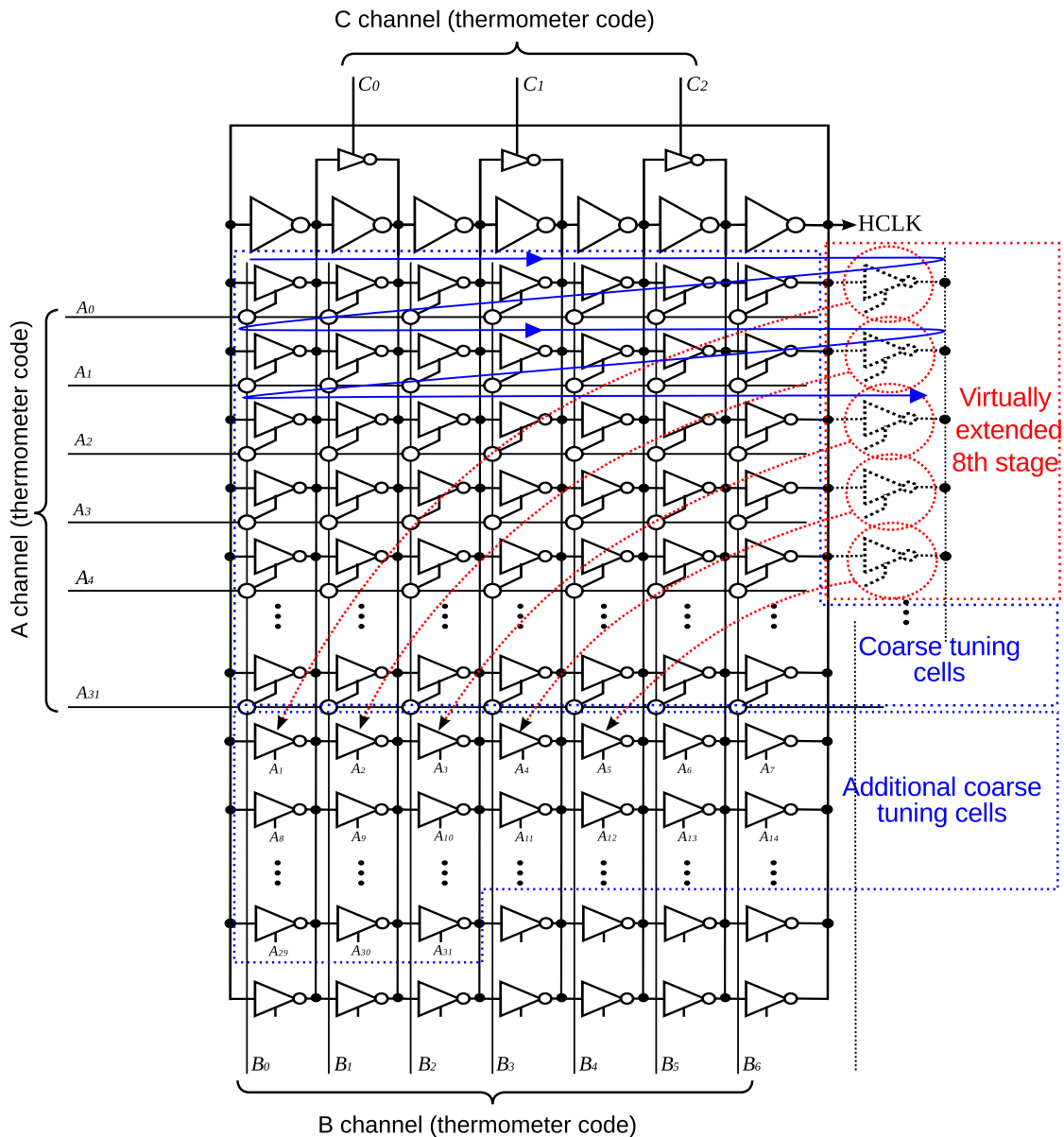


Figure 3.13: **Virtual extension of number of stages:** CTI cells from 8th virtual stage are relocated as CTIA cells in additional rows of oscillator

are distributed over the 7 physical stages. However, they are considered by the control circuitry as the 8th stage. As it will be shown below, such an extension simplifies considerably the implementation of the coarse cells control.

Coarse cells control

With virtual extension of stages the coarse tuning cells in the DCO are controlled by two thermometer coded signals A and B associated with rows and extended columns respectively. Each cell receives one bit from the thermometer coded buses $A \langle 0 : 31 \rangle$ and $B \langle 0 : 6 \rangle$ corresponding to A and B values. The indexes of A and B correspond to the indexes of the addressed rows and columns respectively (Fig. 3.13). $A \langle 0 : 31 \rangle$ and $B \langle 0 : 6 \rangle$ are used for the generation of the *enable* signal for the individual tuning cell. Fig. 3.14 demonstrates

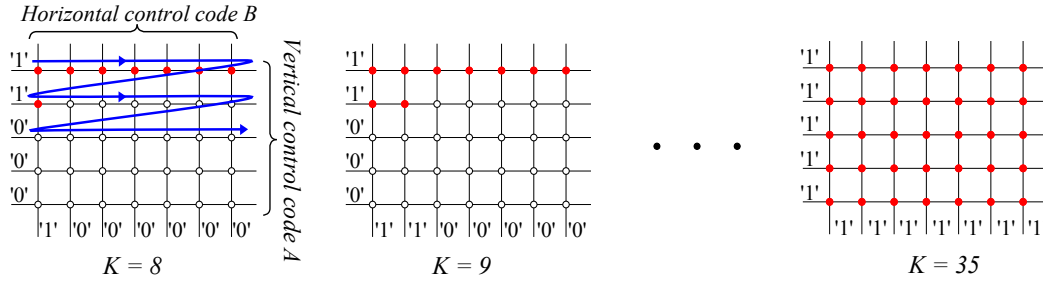


Figure 3.14: **Oscillator control grid:** addressing the CTI cells by two thermometer codes

the example of array 5×7 controlled by this algorithm for the CTI cells.

A and B are generated from the 8 MSB of binary input 10 bit code W by two B2T decoders. To obtain a *thermometer zigzag* activation order of the coarse tuning cells (Figure. 3.9), the values of the control codes A and B are calculated from the K code through the following equations:

$$\begin{aligned} A &= K \% 32 + 1 \\ B &= (K \text{ modulo } 32) \% 4 \end{aligned} \quad (3.16)$$

where $\%$ means the integer division. The operation $(K \text{ modulo } 2^p)$ on a binary coded signal K is implemented by selection of p LSB from K , the operation $K \% (2^p)$ is a right shift of K by p position. Note that according to that formula, the bit A_0 of the thermometer coded A bus is always 1.

The logical equation for individual enable signal of the coarse tuning inverters is given by:

$$EN_{cti\ i,j} = \begin{cases} A_{i+1} \vee B_j \wedge A_i, & i \leq 30; \\ B_j \wedge A_i, & i = 31; \end{cases} \quad (3.17)$$

where $i \in \{0, 1, \dots, 31\}$ is the number of the CTI row and $j \in \{0, \dots, 6\}$ are the number of the CTI row and column respectively. In this formula j defines the physical columns of the array and its range doesn't include the virtual extended row.

For the additional coarse tuning cells implementing a virtual row the logical function is given by:

$$EN_{ctia\ i} = A_{i+1}, \quad i = 0 \dots 30. \quad (3.18)$$

Here i is the index of additional cell equal to the index of the CTI row completed by this cell. Note that the last CTI row with index 31 doesn't have the extension. That is because the coarse tuning cells provide 2^8 frequency values, and for that $2^8 - 1$ coarse tuning cells is required.

The reader can verify that the equations 3.16-3.18 implements a thermometer zigzag activation of the coarse tuning cells when the 8 MSB fraction of the input code K increments (Fig. 3.13).

Fine cells control

The FTI cells are controlled by a separate thermometer code C coded on 3 bit bus $C < 0:3 >$. This code is generated with 2 LSB fraction of the input code K . It may take the value in range from zero (all bits are zero) to 3 (all bits are 1). The code C is given by

$$C = K \text{ modulo } 4, \quad (3.19)$$

and the corresponding individual *enable* signals by

$$EN_{fti\ i} = C_i \quad (3.20)$$

where $i = 0..2$, i is the index of the FTI cell.

3.4 Sizing of the DCO core and tuning cells

The Eq. (3.10) suggests that the ratio between the frequency step ΔF and the initial frequency F_0 is given by the ratio between the currents I_m and I_{cti} generated by the core cells and the tuning cells respectively. Although this equation is derived for the architecture including only coarse tuning cells (Fig. 3.9), it is also valid for fine tuning cells.

It can be roughly considered that the current generated by an inverter is proportional to the W/L ratio of the N transistor. Hence, we come to the following relation:

$$\left(\frac{W}{L}\right)_{main} : \left(\frac{W}{L}\right)_{cti} : \left(\frac{W}{L}\right)_{fti} = F_0 : \Delta F_{coarse} : \Delta F_{fine} \quad (3.21)$$

The parameters F_0 and ΔF_{fine} are given by the specifications (cf. Section 2.2.4). From the architecture of the designed DCO, $\Delta F_{coarse}/\Delta F_{fine} = 4$.

This double relation includes three unknowns (the W/L ratios of the three cell types). The third missing equation is given by the requirement about the absolute value of F_0 or F_{max} . The W/L should be such that when all tuning cells are activated, the ring oscillates at the maximal desired frequency corresponding to the maximal input code value. Unfortunately, the relations proposed in Section 3.2 are based on an approximate simplified transistor model and do not provide exact values of the inverter delays. Hence, after obtaining approximate values of W/L using the formula given there, a simulation-based adjusting/tuning is needed to obtain the desired absolute value of F_{max} .

After fixing the W/L value, the values of L and W must be chosen. An immediate approach consists in minimizing the size of the smallest inverter of the architecture. However, in order to reduce the impact of the fabrication errors on the DCO code-frequency characteristic, the use of tuning cells with minimal sizes allowed by the technology should be avoided. We chose the L of the tuning cells equal to twice the minimal size allowed in the used 65 nm CMOS technology.

The dimensions of the PMOS and NMOS devices in inverters obtained from mentioned considerations are given in Tab. 3.1.

Table 3.1: **Dimensions of the transistors in DCO cells**

Cell	MI	CTI	CTIA	FTI
NMOS	18.72/0.06 μm	1.8/0.12 μm	1.8/0.12 μm	0.4/0.12 μm
PMOS	26.4/0.06 μm	2.56/0.12 μm	2.56/0.12 μm	0.56/0.12 μm

3.5 VHDL modeling of the oscillator

In order to design and study the ADPLL network, a realistic DCO macro-model is needed. This model should be detailed and extended from a generic model presented in Chapter 2.

The DCO macromodeling is required for two reasons. Firstly, an abstraction from the transistor-level DCO schematic is required for a quick simulation of the system. Indeed, the designed ADPLL network includes many DCOs; each DCO includes thousands of transistors. A transistor-level model is practically useless for the simulations required by the system-level design, mostly because of huge necessary computation time.

The main challenge of the realistic DCO macromodeling is the difference existing between the real frequency grid of the DCO and an ideal equal step grid. This difference has been observed through the transistor-level simulation of the DCO and through a measurement of the fabricated DCO prototype. A precise DCO macro-model has been developed. It synthesizes the frequency grid basing on a real data obtained from precise simulation and measurement.

3.5.1 Precise modeling of the real code-frequency characteristic

The main difference of the precise DCO macro-model from the generic model proposed earlier in the project is the possibility to define the output frequency grid through a look-up table. The look-up table (LUT) contains the values of the output frequency corresponding to each 10^{th} input code (0, 10, 20, ...). The frequency values corresponding to the intermediate codes are obtained by first-order linear interpolation Eq. (3.15) with the formula

$$F_{DCO}(i) = F(10(i\%10)) + \frac{10(F(10(i\%10 + 10)) - F(10(i\%10)))}{i \text{ modulo } 10} \quad (3.22)$$

where $F(10(i\%10))$ is a value of the frequency from the LUT, i is the number of the frequency grid point, $i = 0..1023$.

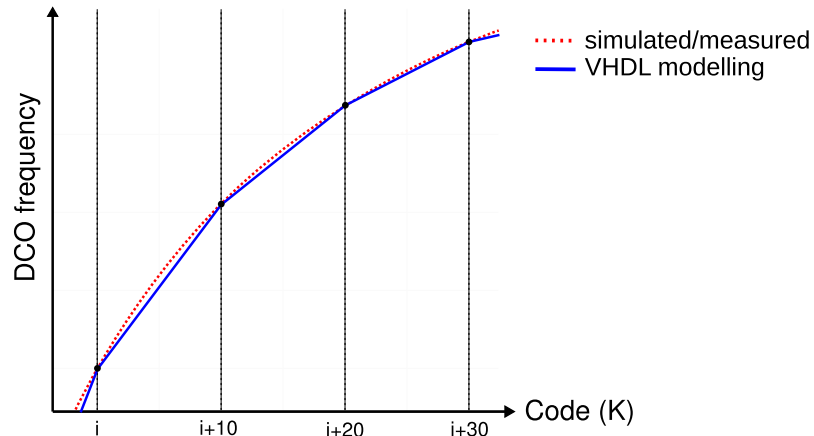


Figure 3.15: Approximation of the code-frequency characteristic in VHDL model of DCO.

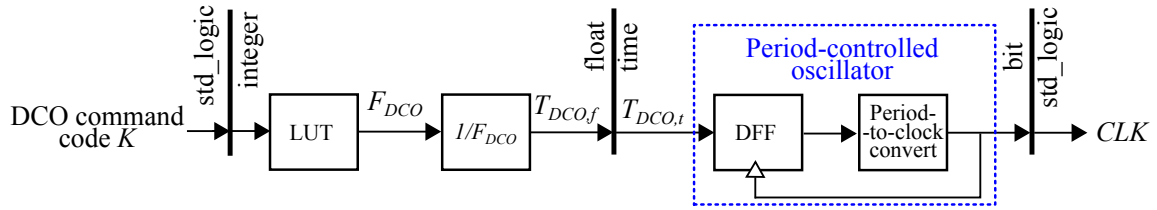
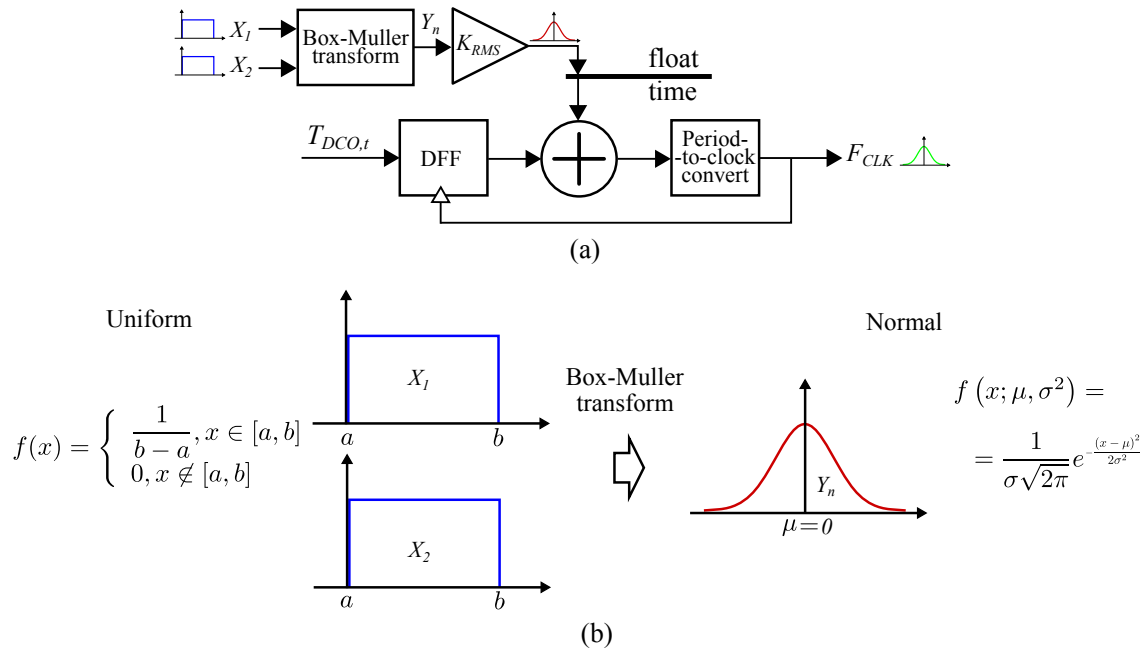


Figure 3.16: Principle of operation of VHDL model

Figure 3.17: **Period-controlled oscillator with jitter injection:** (a) block diagram and (b) generation of a random variable with normal distribution using Box-Muller transform.

3.5.2 Synthesis of the DCO output frequency in the precise VHDL model

The implemented VHDL model is a period-controlled oscillator, whose period depends on the input code through the formula Eq. (3.22). In the VHDL models, the time resolution is 1 fs. Fig. 3.16 demonstrates the structure of the model.

The input binary control code is used for selection of the appropriate frequency value from a LUT. The LUT outputs a float-point frequency value F_{DCO} , which is then converted to a period T_{DCO} in time format. The latter is then used for the clock generation VHDL instruction $CLK \Rightarrow not CLK \text{ after } T_{DCO}$.

The implemented model must account for the DCO jitter. This is necessary for realistic modeling of the ADPLL behavior and performance. In the DCO macromodel, the jitter is modeled by adding a random Gaussian variable Y_n to the value T_{DCO} at each period of the output signal. This variable represents the additive random fluctuation at each ideal time-stamp of clock signal [71]. VHDL itself does not provide a dedicated command for the generation of random variables with normal distribution. VHDL provides a function generating pseudo-random numbers only with uniform distribution. To generate a Gaussian variable, the Box-Muller transform is used. The Box-Muller transform generates a normally distributed

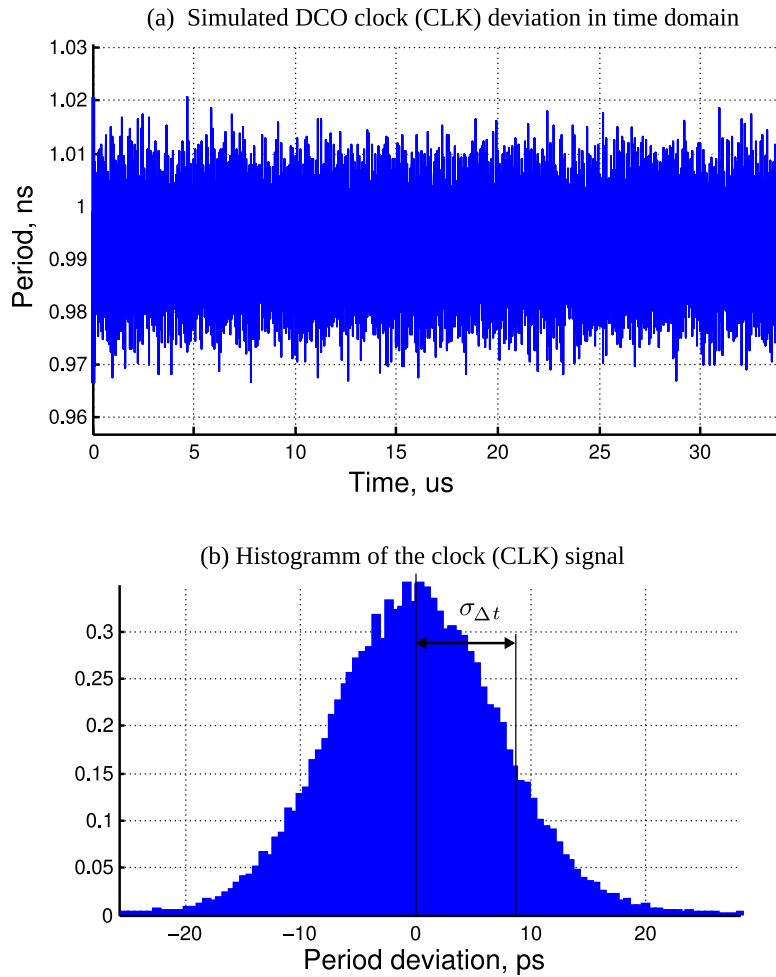


Figure 3.18: **Simulation of the DCO behavioral model:** (a) deviation of the oscillation period in time domain and (b) distribution of the generated period deviations.

random variable from the two uniformly distributed random variables. This transform is given by

$$Y_n = \sqrt{-2\ln X_1} \cos(2\pi X_2) \quad (3.23)$$

where Y_n is a normally distributed variable with expected value 0 and variance 1; X_1 and X_2 are independent random variables uniformly distributed in the interval $[0, 1]$.

The complete listing of the precise VHDL macro-model is given in Appendix A.

3.6 DCO layout desing I: the DCO floorplan

The proposed DCO has a regular structure and is composed of 5 blocks: main inverters, 255 CTI, 11 CTIA, 3 FTI and frequency dividers. The main challenge of the DCO layout implementation is the management of the 266 CTI placement and the routing of the control signals. The implementation strategy must optimize the performances of the DCO and the circuit area. The following techniques have been used for the DCO layout design:

- Cell based design. The DCO layout is an assembly of identical elementary blocks implementing CTI, CTIA, FTI, MI and interconnections;
- Connection by abutment. The cell layout is designed so that they are connected by geometrical abutment;
- A flow oriented location of the cells. The choice of the placement of the cells accounts the signal propagation direction;
- Cell oriented supply design. As in conventional geometries of digital circuit layouts, the supply is routed through multi-finger structures (Fig. 3.19). That allows each cell to share its ground and supply pads with the neighbors.

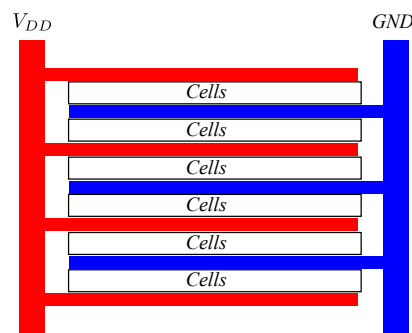


Figure 3.19: **Interdigitated multi-finger power routing geometry**

3.6.1 Cell based design

In the cell oriented design, the functional blocks are implemented as cells having common geometrical parameters. For compatibility with the standard library cells, the custom DCO cells must have the same geometrical constraints as the standard design kit cells:

- The height of the cells;
- The topology of the supply/ground wires;
- The location and maximal dimensions of substrate and N-well polygons;

Fig. 3.20 demonstrates a template of the cell with prerouted supply and reserved space for active transistors and routing wires. The prerouted supply and ground are done in metal layer Metal1. The width of the power strips is $0.56\ \mu\text{m}$. They are shared with the neighboring cells from bottom and top sides.

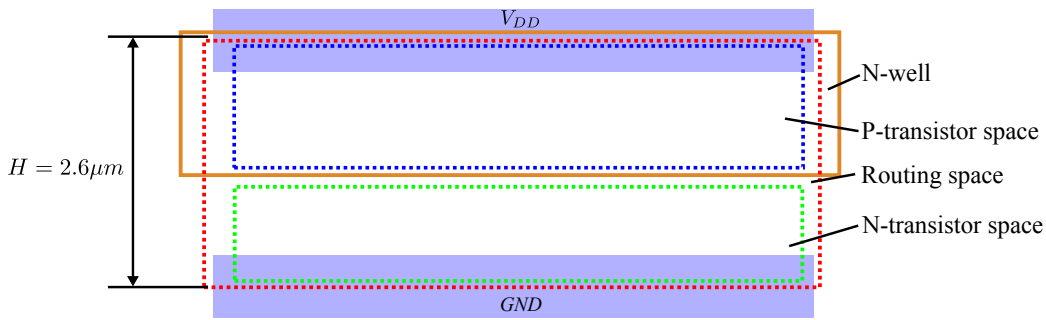


Figure 3.20: **Cell layout template:** routing space includes the whole cell area, the N-well defines the location of the P-transistors. The height of the cell ($2.6 \mu\text{m}$) and the supply/ground polygons size are the same as those of the standard library cells.

The routing of signals is free within the cell area at the levels of metal Metal2-Metal7. The routing space is only limited by the supply strips at metal layer Metal1.

The active zones of transistors should be placed inside the area specified by the red dotted rectangle. The N-well zone is specified by the blue rectangle area: it is reserved for P transistors. The green rectangle specifies the P-well zone for the N transistors.

3.6.2 Power planning

The power distribution network design is critical for analogue and mixed signal circuits. A smart power planning can reduce the coupling between blocks and subcircuits and can improve the power supply noise rejection.








The power distribution network is implemented with interdigitated fingers routing V_{DD} and GND lines (Fig. 3.19). In this structure the cells are placed in the space between the supply and the adjacent ground fingers forming a row of cells. Each V_{DD} and GND finger supplies two rows of the cells. The substrate connections are placed under the supply strips, so saving the silicon area.

Two vertical wide rails in upper metal layer Metal6 distribute the supply/ground through the oscillator core layout. The horizontal rails in Metal2-Metal1 layers (the fingers) distribute the supply/ground to the rows of the oscillator cells. The use of double layer Metal2-Metal1 and of more than 200 vias between them allows to low down the IR drop voltage and decrease the effect of coupling between the stages of oscillator. However this feature limits the dimensions of active areas and complicates the routing of the local signals in the lower metal layers.

A three dimensional view of the layout highlighting the power distribution network structure is given in Fig. 3.27 in Section 3.7.2.

The width of the fingers and power rails is calculated so to provide the minimal resistance and to meet the electromigration rules for the given technology. The width of the fingers in Metal1 and Metal2 layers $w_{M1,2}$ is $0.56 \mu\text{m}$ and for the Metal6 rails is $6 \mu\text{m}$. According to the design kit maximum RMS current for the metal layer we calculate the maximal current that can provide these elements of power distribution.

Table 3.2: Layers functionality attribution

Level	Technology layer	Function
0	Metal1 	local supply and connections
1	Metal2 	local supply and connections
2	Metal3 	vertical signals
3	Metal4 	horizontal signals
4	Metal5 	global supply
5	Metal6 	global supply
6	Metal7 	global supply

The IR voltage drop due to the resistance of the horizontal power fingers (Metal1-Metal2) is less than 3 mV, and for the vertical Metal6 layer rails on a full length of wire is less than 6 mV.

In order to improve the power supply rejection, the decoupling capacitors are placed regularly within the structure of the DCO. They are implemented as polysilicon capacitors formed by gates of PMOS transistors [6, 40], Fig. 3.21.

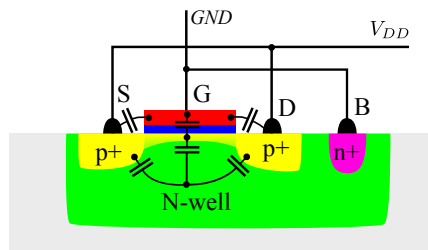


Figure 3.21: Supply decoupling polysilicon CMOS capacitor.

3.6.3 Signal flow oriented layout

In the signal flow oriented layout design the cells are placed according to the topology of the signal propagation paths. In particular, the cell pins are placed in a way to allow the cell connection by abutment. The Tab. 3.2 defines the functional attribution of the available metal layers. The high frequency clocks and control signals are routed at different levels, so reducing the capacitive coupling. Fig. 3.22(a) shows the structure of the DCO layout on an example of an array 4×7 of CTI cells. Since the V_{DD} and GND stripes alternate, each pair of neighboring rows must have a vertical symmetry (the neighboring rows are mirrored). To implement it properly, a single layout cell contains two mirrored CTIs cells ranged in a column Fig. 3.22(b). As it can be seen, the layout cells include the bypassing row and column control lines and the input and output signals. The cells are connected by abutment, which simplifies the assembly of the DCO.

The CTIA cells are implemented in a similar way. The 31 CTIAs placed on 5 rows are controlled by the bits 1...31 of the A bus which correspond to the CTI row control. This fact

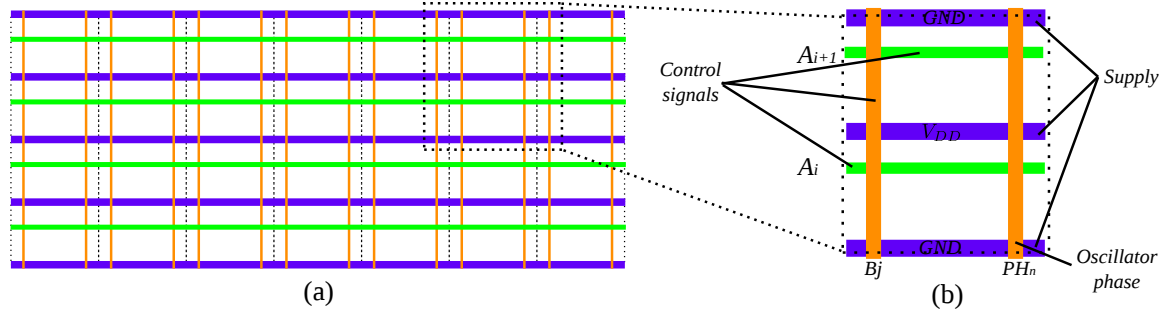


Figure 3.22: **Signal flow driven placement of the tuning cells in DCO:** (a) array 4×7 of CTI cells with propagated supply, control signals, stage output signals; (b) single prerouted CTI cell.

leads to a need of a non-regular topology of the control signal routing. For this reason, the latter is individual for each cell and has been done manually.

Three fine tuning inverters are placed at the bottom row and has been routed manually.

3.6.4 Guard rings

To decrease the effect of substrate coupling with surrounding digital circuits, the oscillator is isolated from the aggressors by guard rings routed around it. The guard ring is a continuous path of the substrate connection to the supply or to the ground. It collects the parasitic currents flowing through the substrate, so ensuring an isolation of the surrounded block from the substrate noise generated by the environment.

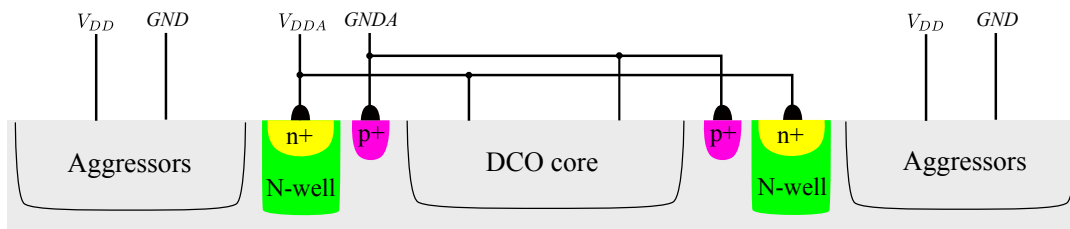


Figure 3.23: **Guard rings around oscillator core:** combination of two guard ring to collect major carriers of both polarities

Fig. 3.23 demonstrates the geometry of the used guard rings. The N-well guard ring collects the stray electrons injected by NMOS devices and flowing in substrate toward the oscillator core. The P-diffusion guard ring collects the stray holes from the substrate.

3.6.5 General DCO floorplan

The general floorplan of the oscillator includes 13 groups of blocks. They are towered in stack in the order depicted in Fig. 3.24. These groups are:

- frequency dividers;
- main inverters;

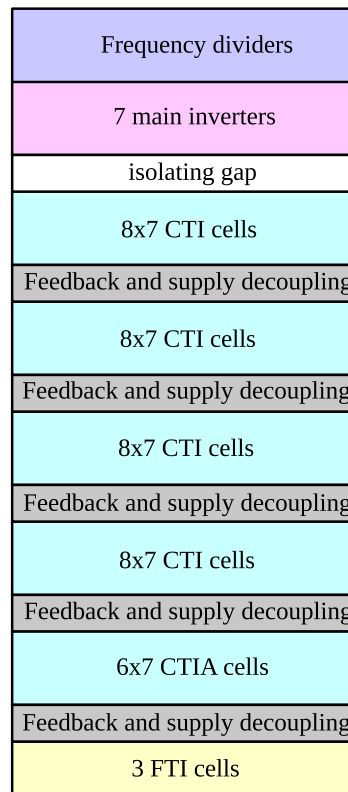


Figure 3.24: **Floorplan of the designed oscillator**

- arrays of CTI cells;
- arrays of CTIA cells;
- array of FTI cells;
- feedback line and supply decoupling cells;

The dividers are located so to be far from the sensitive tuning cells. They are aligned with the main inverter row. A free space (isolating gap) is available for the routing of signals. Four combinations of CTI cells (composed as 4×7), the decoupling capacitors and global loop feedback lines are placed under the main inverter row. The array of FTI cells is placed at the bottom of this stack.

3.7 DCO layout design II: cell design

This section sections describe the cell layout design procedure.

3.7.1 Main inverters

The role of the main inverters of the oscillator is to provide a fixed component of the stage driving current. A main inverter is implemented with a CMOS inverter provided in the standard library of the design kit. A custom-based design could be an acceptable option as well.

A large inverter $x213$ has been chosen from the standard library, so to comply its dimensions with the requirement given by Eq. (3.10). Fig. 3.25 shows the schematic of this inverter. It consists of 24 parallel inverters with transistors NMOS $M_1 - M_{24}$ and PMOS $M_{25} - M_{48}$.

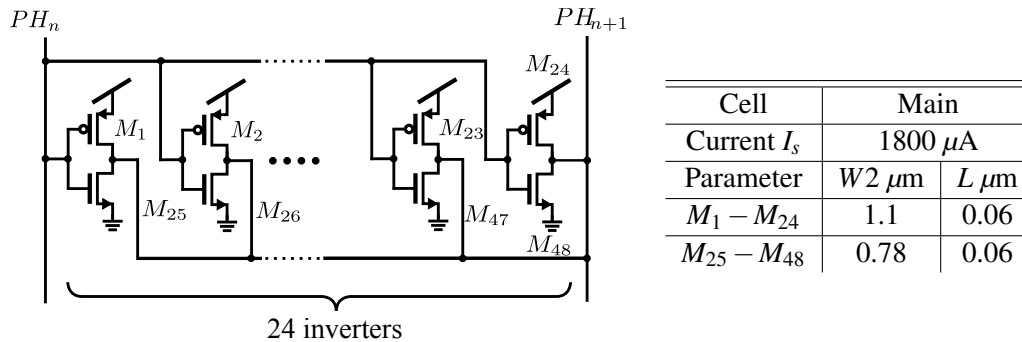


Figure 3.25: **Schematic of the main inverter of oscillator:** 24 inverters connected in parallel.

The layout of a main inverter is shown in Fig. 3.37(a). The inverters are connected by abutment. The substrate connections to the power rails so to prevent latch-up and decrease the substrate noise were done as close as possible to the inverter cells. The inverter cells have their own power and ground wires in Metal1 layer so that the row supply strips are naturally created when the inverters are connected by abutment. Additional wires of layer of Metal2 are placed above the cell supply rails in Metal1 layer, and are connected to them by hundreds of via. This is done in order to decrease the resistance of the supply lines.

The inverters are connected to the oscillator stages by vertical stripes of the Metal3 layer. The distance between these wires is fixed. This means that the main inverters can be connected by abutment to other cells of oscillator at any position on the vertical axis of the floorplan.

3.7.2 Feedback wire and decoupling capacitors

The oscillator feedback line which closes the oscillator loop is placed in a special service cell (designated as "Feedback and supply decoupling" in Fig. 3.24). This service cell contains also the supply decoupling capacitors. 5 instances of this cell are distributed over the stack of the delay cells of the oscillator. The complete layout of cell is shown on Fig. 3.26.

The feedback lines are of 0.56 μ m width and 44.6 μ m length, and are implemented in the Metal4 layer. Such a large width and high metal level has been chosen in order to reduce their resistance and capacitance and consequently the associated parasitic delay.

The DCO supply decoupling capacitors are described in Subsection 3.6.2. Each MOS capacitor has a value of 25.49 fF. Totaling 2×7 capacitors, each double row cell provides total 356.86 fF of decoupling capacitance.

The height of the feedback line/decoupling capacitance cell is twice the height of a standard cell: it occupies two rows shown in Fig. 3.19 (a double row cell). Fig. 3.27 presents a

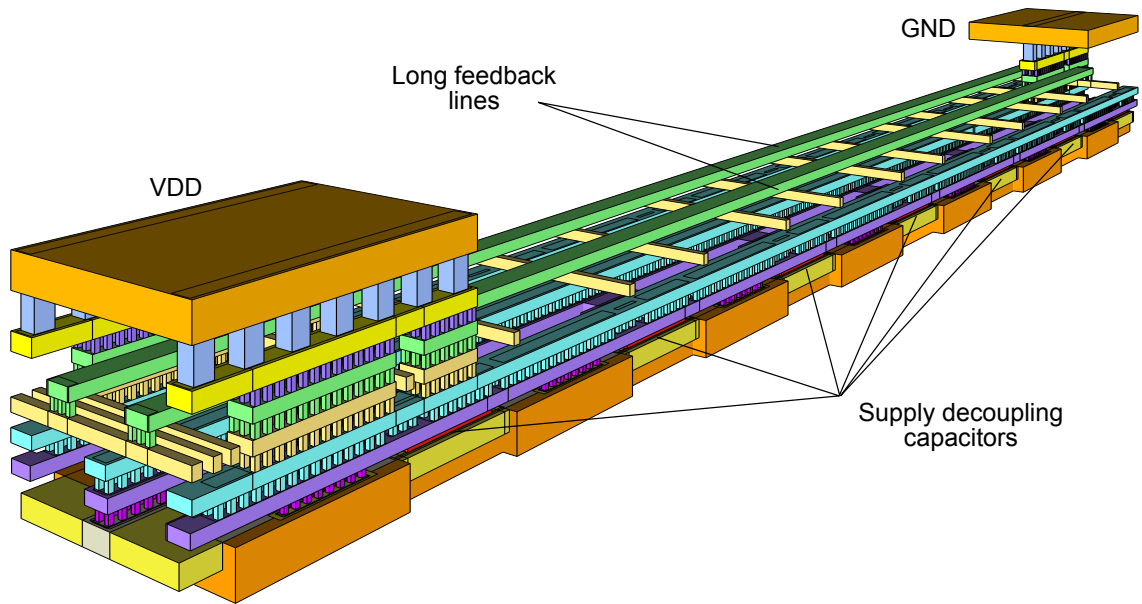


Figure 3.26: Layout of the feedback line/decoupling capacitance service cell

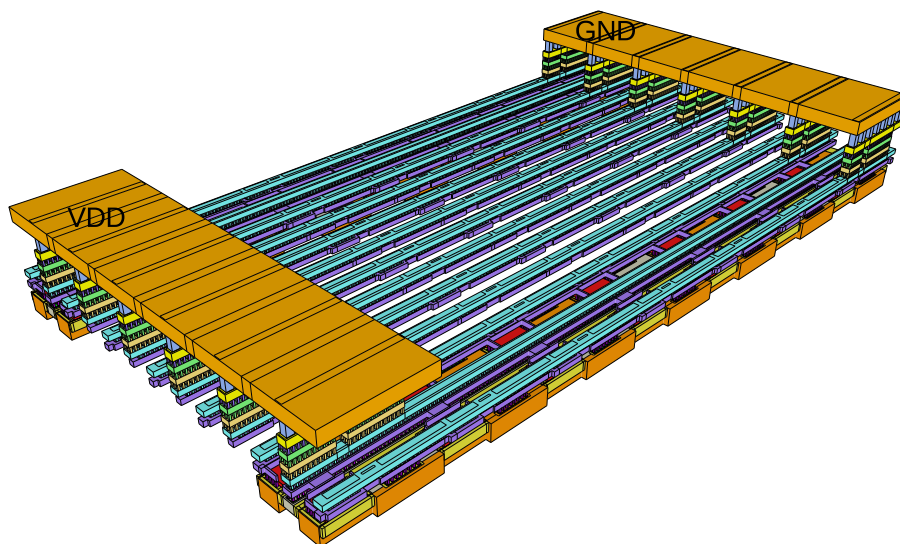


Figure 3.27: Feedback/decoupling capacitor cell in the power distribution network: example of 8×7 array block bounded by two feedback/decoupling capacitance cells.

three dimensional view of the power distribution network in which the feedback line/decoupling capacitance cells are inserted.

3.7.3 Tuning inverters

Each tuning cell of the DCO is provided with a local circuitry decoding the A, B and C signals depending on its specificity and generating the individual enable signal. The proximity of this local decoding circuit to the inverter cell allows an optimization of the layout and a minimization of the delay.

In order to optimize the size of the tuning cells, their layout is designed in a custom way (manually). The tuning cells use the general cell structure defined in Fig. 3.20. The latter

template was slightly modified in order to provide flexibility for the inverter sizing. The Fig. 3.28 demonstrates the template of the layout used for design of tuning cell layout. It contains the prerouted supply wires, the layout of three-state inverter with flexible size and reserved space for the transistors of the local control logic.

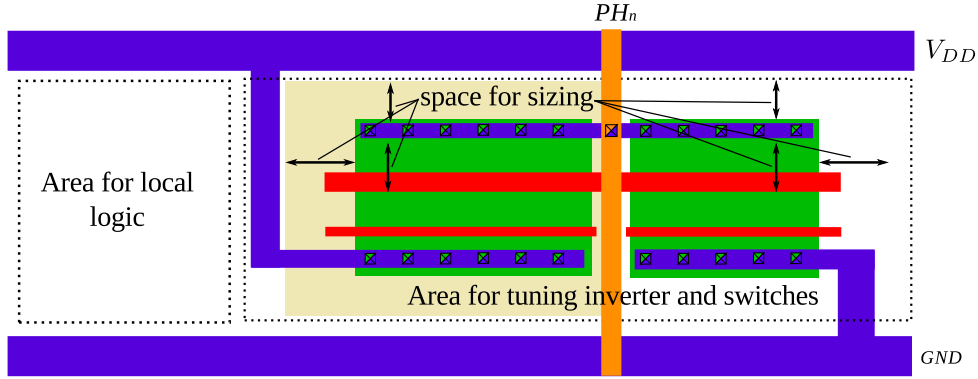


Figure 3.28: **Single tuning cell layout template**

CTI cells

The encoding logic circuit of the CTI cells in i^{th} row and j^{th} column receives three control signals: two horizontal A_i, A_{i+1} and one vertical B_j . The enable signal is obtained according to Eq. (3.17).

The schematic of the CTI cell is shown in Fig. 3.29. It consists of tuning tri-state inverter and the local control logic. The CTI is routed in the cell according to Fig. 3.28. All transistors in this cell are oriented in the same direction in order to improve the manufacturability of the circuit and minimize the mismatches between the cells. Apart the transistors forming the CMOS inverter, all transistors in local control logic have minimal dimensions.

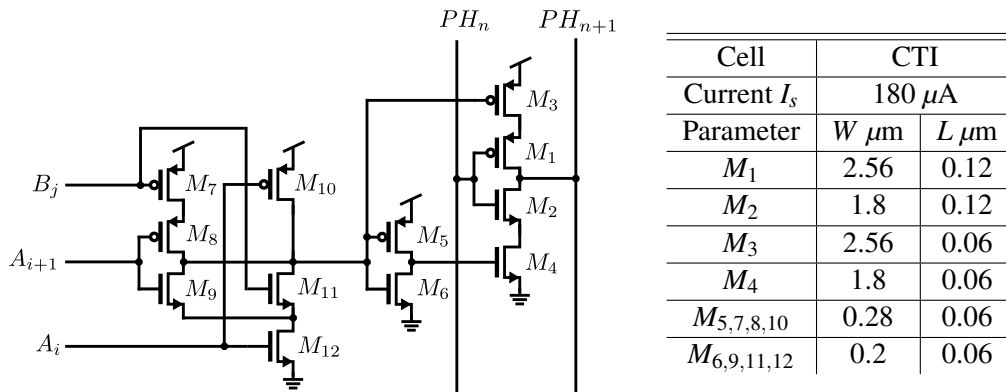


Figure 3.29: **Schematic diagram of the coarse tuning cells**

The layout of the proposed cell is shown in Fig. 3.30. The CTI layout cell is a double CTI cell (double height cell) for the reasons explained in Section 3.6.3. The DCO architecture uses an array of 4×7 double height CTI layout cells ($2 \times 4 \times 7 = 56$ CTIs).

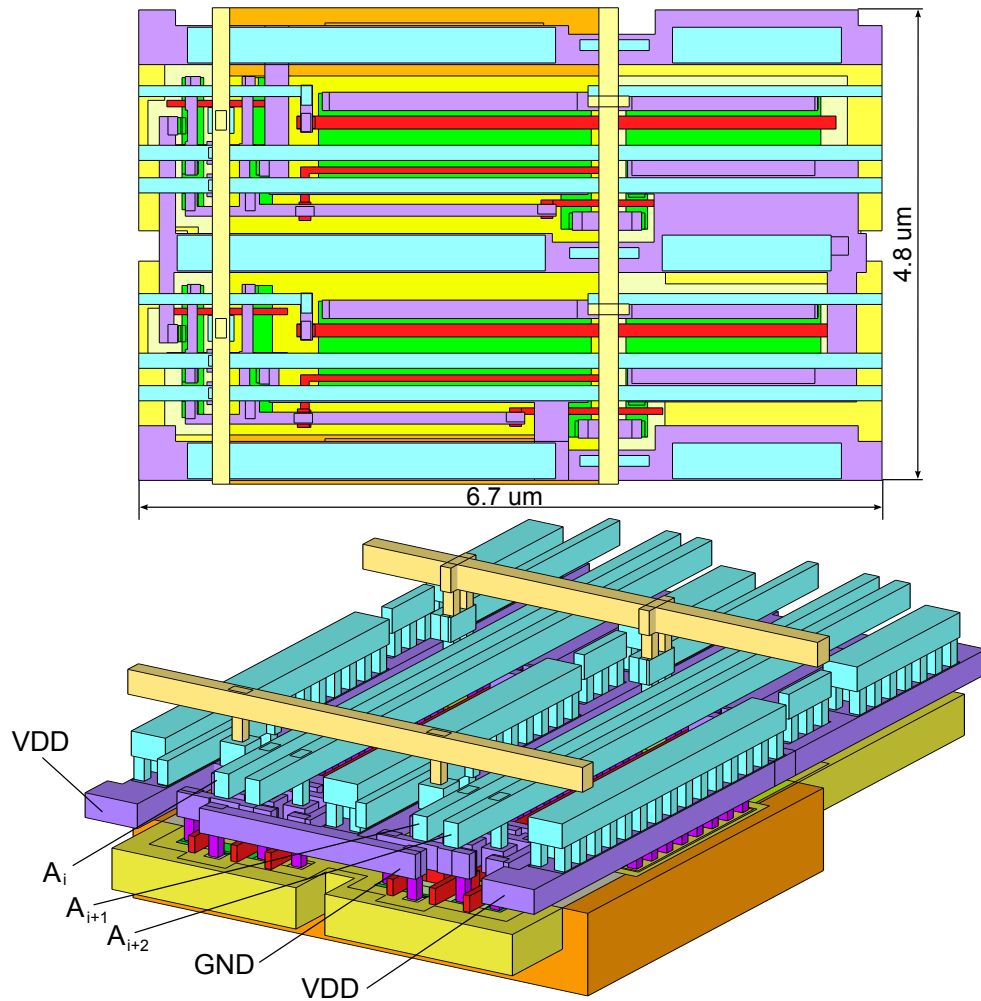
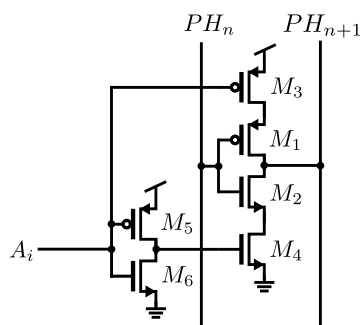


Figure 3.30: **Layout of the coarse tuning cells: two instances with local control circuitry**

CTIA cells



Cell	CTIA	
Current I_s	180 μA	
Parameter	W μm	L μm
M_1	2.56	0.12
M_2	1.8	0.12
M_3	2.56	0.06
M_4	1.8	0.06
M_5	0.28	0.06
M_6	0.2	0.06

Figure 3.31: **Schematic diagram of the additional coarse tuning cells**

The schematic of the CTIA cell and dimensions of the transistors are shown in Fig. 3.31. The CTIA layout is shown in a Fig. 3.32. As for the CTI, it consists of a double height cell. As explained in Section 3.6.3, due to the complicated routing of the A bus control signals, the local A control lines were routed manually.

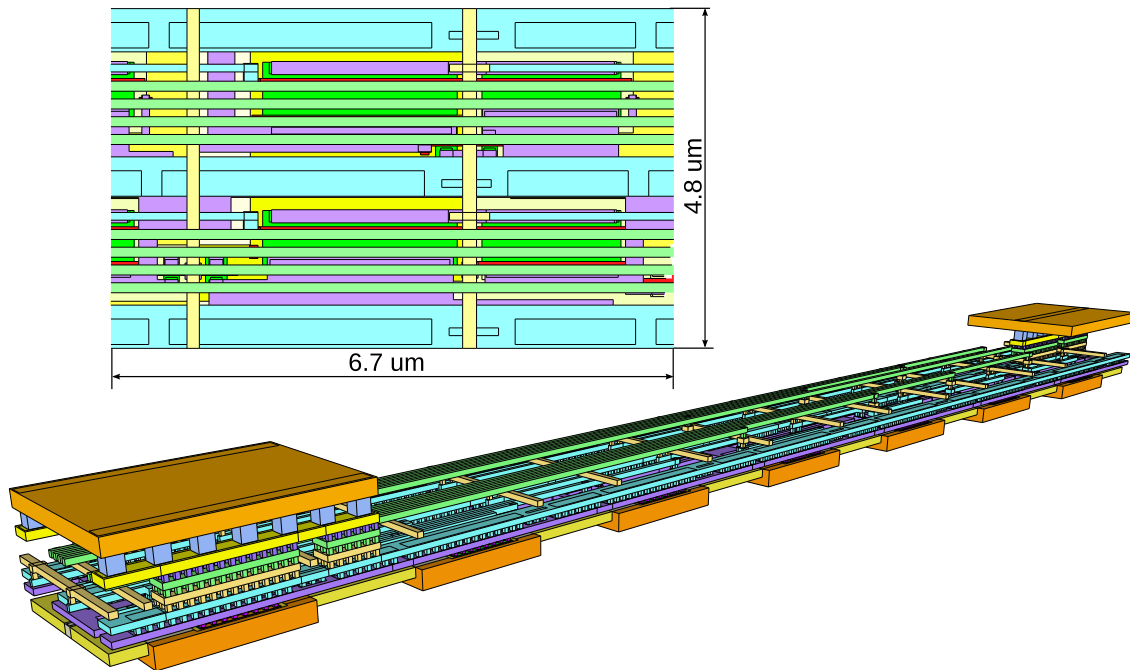
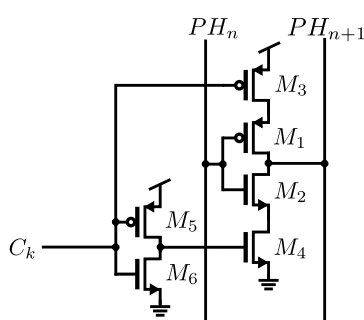


Figure 3.32: **Additional coarse tuning cells layout**: flat view of the additional coarse tuning cell and layout of the 7×2 array with local control circuitry and routed supply

FTI cells

The FTI cells use the template layout given in Fig. 3.28. They have the same schematic as CTIA cells. They are controlled by the C bus, and are implemented in a simple-height layout cells. The transistors in the tuning inverter in FTI cells have smaller width in order to provide a smaller current and a smaller frequency step (Fig. 3.33).



Cell	FTI	
Current I_s	65 μA	
Parameter	$W \mu\text{m}$	$L \mu\text{m}$
M_1	0.56	0.12
M_2	0.4	0.12
M_3	0.56	0.06
M_4	0.4	0.06
M_5	0.28	0.06
M_6	0.2	0.06

Figure 3.33: **Schematic diagram of the fine tuning cells**

The layout of the FTI cells is demonstrated in a Fig. 3.34. It is a single height cell with prerouted supply and ground stripes. Since there are only three FTI cells in oscillator topology, the layout is placed in a single row 1×7 with four dummy cells without any transistors, but with prerouted supply/ground wires, global and local control signals which only pass through them (so called "transparency" of the cell [6]).

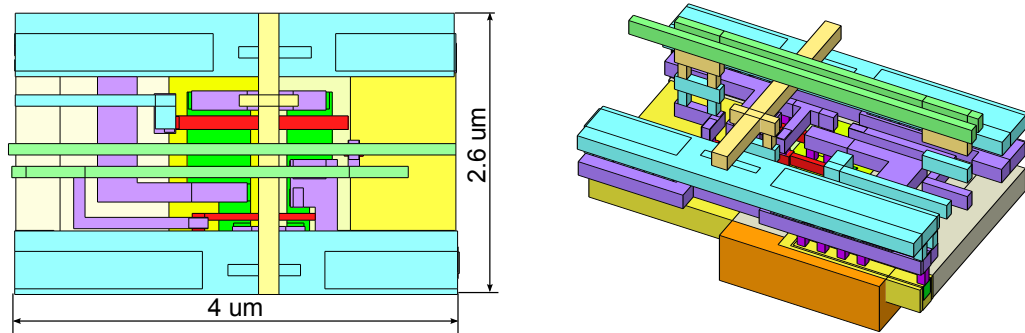


Figure 3.34: Layout of the FTI cell: flat view and 3D view

3.7.4 DCO output interface

The output interface of the DCO includes the frequency dividers and buffers.

The output from the ring oscillator is successively divided by 2, by 4 and by 2. The first division by 2 is done in order to obtain a 0.5 duty cycle on the output clock. That is required by some applications using multiphase clocking (e.g. double data rate transfer of data). In the original DCO signal the symmetry between the positive and negative phases is not guaranteed, since it is practically difficult to match the currents of the PMOS and NMOS transistors in a CMOS cell (cf. Fig. 3.35).

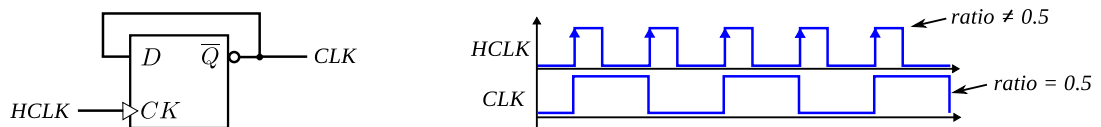


Figure 3.35: Improvement of duty cycle by divider: schematic and timing diagram

The divider block generates three clock signals: *CLK* is a divided by 2 clock for the local SCA clocking (the output of the ADPLL network clock generator), *DIV* is a divided by 8 clock for the feedback of the ADPLL and for the phase coupling with neighbors. The signal *DIV16* is a divided by 16 clock routed on the chip pads for the off-chip test purposes. Only this signal can be properly measured outside of the chip, since the communication frequency of the I/O pads available from the design kit is limited to 270 MHz.

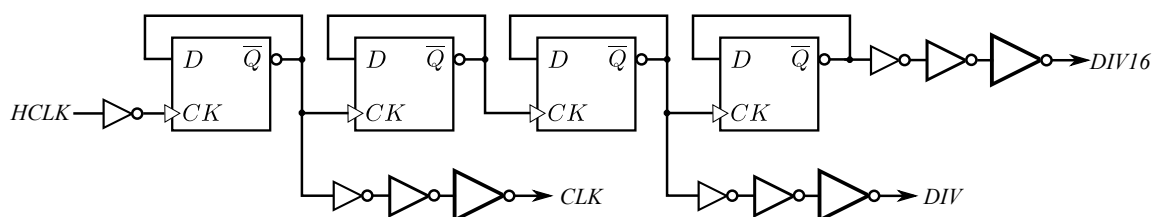


Figure 3.36: Schematic diagram of the feedback frequency divider

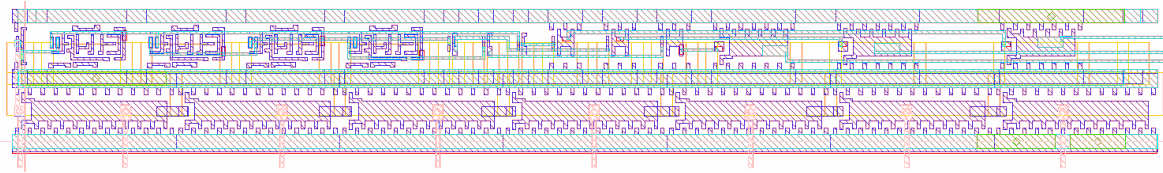


Figure 3.37: **Layout of the block containing the main inverters of oscillator, dividers and buffers**

Fig. 3.36 shows the schematic of the divider. It consists of input buffering inverter and four D flip-flops with inverted output. Input inverter is used to decouple the oscillating core of DCO from the dividers and output buffers. It has minimal dimensions and hence, minimal parasitic capacitance between input and output.

The output buffers are used to deliver enough driving current to the capacitive loads of the local clock distribution network in SCA. It is necessary to use several cascaded inverters to match the weak outputs of dividers with load. As recommended in [6], the load coefficient of the inverter (called fanout) should be equal to ≈ 4 in order to provide a minimal delay in the chain. The inverters thus were chosen according to their fanout load specified in the name of library standard cell. In our case, taking into account the fanout load of the dividers, the chains contain standard design kit cells with the following size multipliers: *divider* \Rightarrow *x9* \Rightarrow *x35* \Rightarrow *x142* \Rightarrow *output*.

The divider and buffers are placed close to the main inverter blocks of oscillator. The layout of the divider is designed following the same methodology as the oscillator core (Fig. 3.37). It shares one of the supply rails with the main inverters and the ground rail is connected to the oscillators global ground rail in metal layer Metal5.

3.7.5 *A*, *B* and *C* bus generator

The buses *A* $\langle 0:31 \rangle$, *B* $\langle 0:6 \rangle$ and *C* $\langle 0:2 \rangle$ code thermometer-coded integer values obtained from the input binary coded 10 bit signal *K* $\langle 0:9 \rangle$ according to equations 3.16 and 3.19. By convention, the thermometer code is zero when all bus wires are at zero values.

The bus *A* $\langle 0:31 \rangle$ is generated as follows: bit 0 of bus *A* is always '1' according to Eq. (3.16), and bits 1 : 31 are generated by B2T decoder from bits 5-9 of input code *K*. The bus *B* $\langle 0:6 \rangle$ is generated by B2T decoder from bits 2-4 of input code *K*. The *C* $\langle 0:2 \rangle$ signal is generated by a B2T from bits 0 and 1 of the input code *K*.

The schematic diagram of the control circuitry is given in (Fig. 3.38). The signals *A*, *B* and *C* are latched in order to remove glitches. Three decoders and the output registers are implemented as a separate block through a standard synthesis flow. Fig. 3.39 demonstrates a layout of the control block. It is composed of standard library cells and occupies $10.4 \times 100 \mu\text{m}^2$ of the chip area.

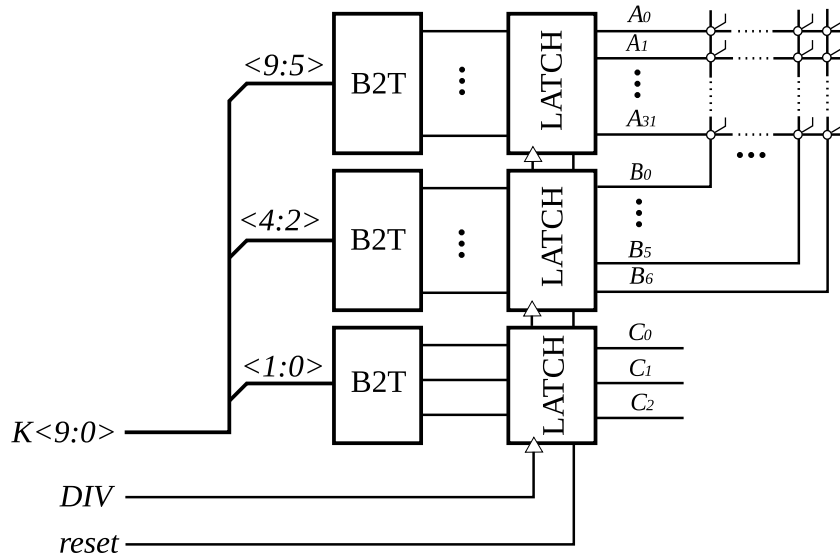


Figure 3.38: Schematic of the A , B and C signal generation circuit

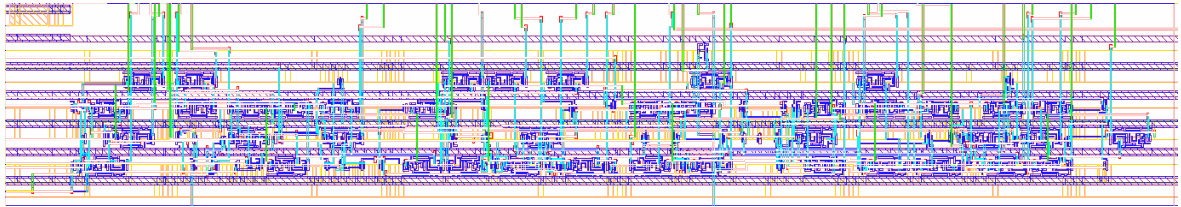


Figure 3.39: Layout of the B2T decoders

3.7.6 DCO chip

The DCO test chip has been designed and fabricated in order to validate the DCO design. The structure of the chip is shown in a Fig. 3.40. The chip includes the DCO core, the divider block and the control logic circuitry including three B2T decoders and the latches for A , B and C buses. In order to provide the slew rate adjustment of the I/O pads and its PVT compensation, the special service cell provided in the design kit was used. It generates the code controlling the pad operation. This code is distributed by the internal connections in the pad ring.

The chip I/O interface includes the 10 bit input binary code allowing the frequency selection, and two divided DCO output DIV and $DIV16$ (cf. Section 3.7.4). As we mentioned, only $DIV16$ output with maximal frequency 175 MHz (at maximal input code) can be properly transmitted through the available output pads with frequency bandwidth of 270 MHz. However the output DIV is also lead out of the chip for testing/verification purposes.

The chip requires two supply sources VDD/GND and $VDDA/GNDA$. The first one is used for the digital circuitry, and the second one is for oscillator core. These power domains in schematic and layout are kept separately, in order to reduce the cross-talk between the digital and analog parts, and also for the testing purposes. A separate supply allows a precise measurement of the power consumed by analog and digital part of the circuit. The impact of the supply variation on oscillation frequency can be measured too.

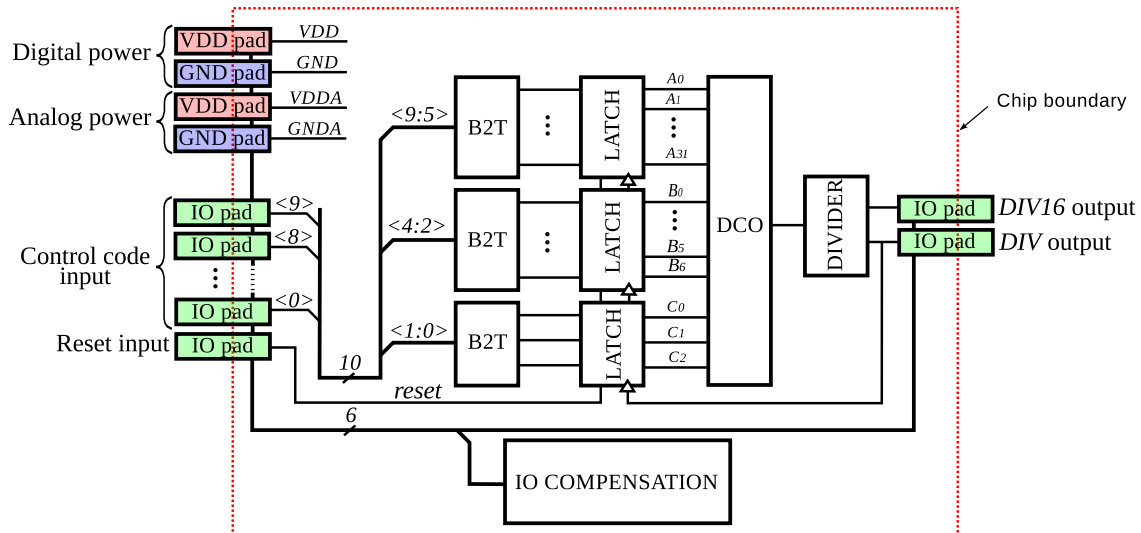


Figure 3.40: Schematic diagram of the DCO test chip

The total area of the circuit is 0.28 mm^2 . The core of oscillator together with double guard ring are placed close to the analog $VDDA/GNDA$ power pads and occupy $6324 \mu\text{m}^2$. The digital components of the circuit are placed aside the DCO core and occupy $1040 \mu\text{m}^2$. The rest of the chip area ($\approx 0.275 \text{ mm}^2$) is occupied by the I/O compensation cell, by the pad ring and by substrate connection fillers.

Fig. 3.41 is a microphotograph of the fabricated and packaged chip. Due to the metal filling procedure which is applied by default in all advanced sub-micron process, the details of the circuit layout are not visible.

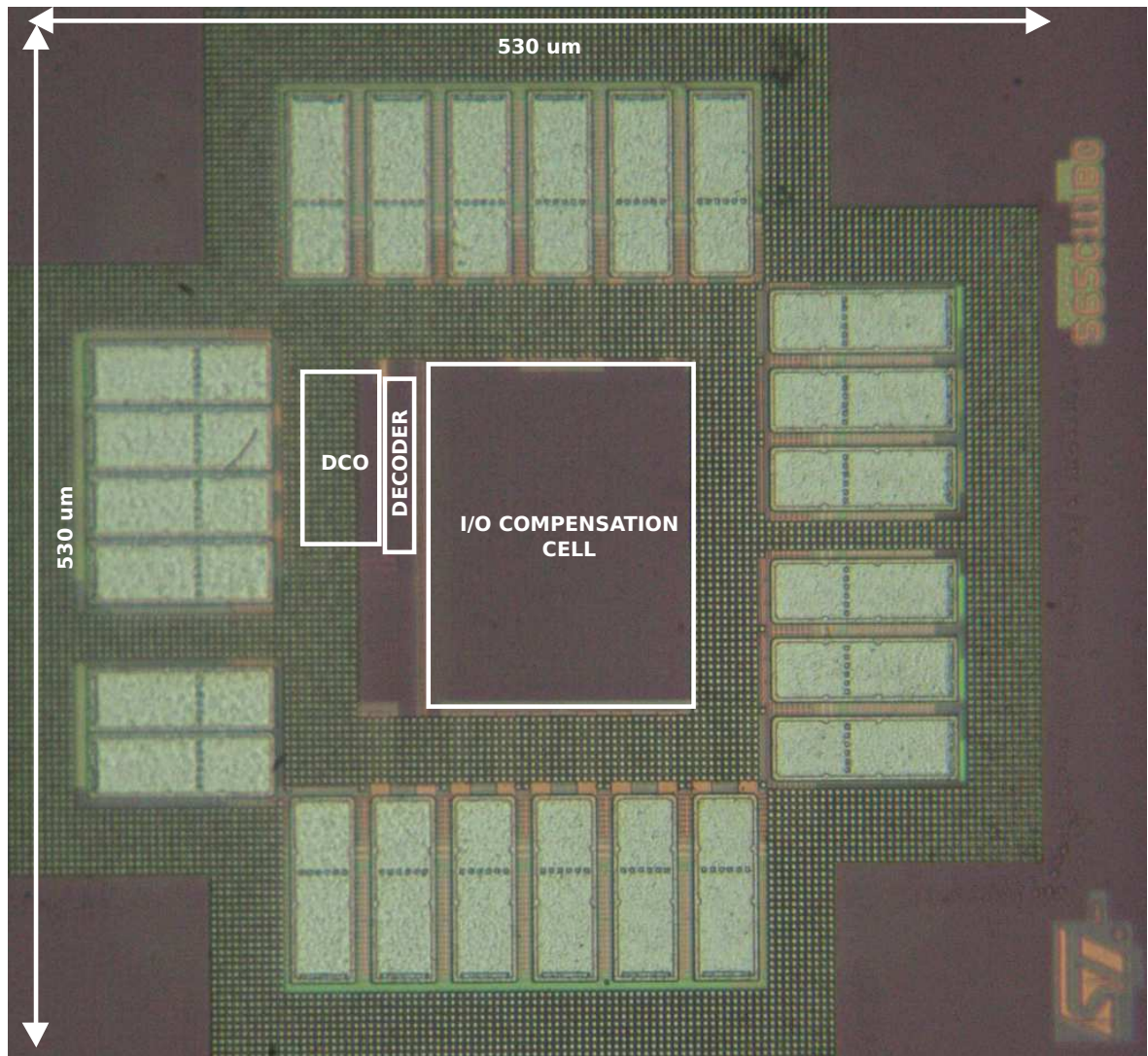


Figure 3.41: Microphotograph of the fabricated chip

3.8 DCO chip test

This section presents the results of the measurements of the fabricated DCO chip. The code-frequency characteristics of the DCO have been measured for free-running oscillator. More details of the measurement set, used tools, methodology and data post-processing procedures can be found in Appendix C and Appendix D.

The Tab. 3.3 presents the performance and parameters summary of the fabricated DCO test chip.

3.8.1 Impact of the supply variations

Fig. 3.42 shows the measured code-frequency characteristic for different supply voltages of the oscillator core. The relative variation of the oscillation frequency for the $\pm 5\%$ supply variation is $-7.8/+7.2\%$ ($-38.8/+35.6$ MHz) for the lowest range of frequencies. The same variation is observed for the highest frequencies $-7.4/+7.4\%$ ($-91.6/+86.4$ MHz).

In the plot Fig. 3.42 the plain lines give the measured results, and the dotted lines give the result predicted by simulation. The difference in the frequency values between the simulation and the measurement is under 10% for all input code range, which is a fairly good result. However, there is a systematic difference of $\sim 14\%$ between the slopes of the code-frequency line (the DCO gain).

The simulated and measured code-frequency characteristics highlight a high sensitivity to the supply variations. However, for any supply voltage within $\pm 5\%$ range, the specified nominal frequency 1 GHz is reachable with an input code far from the limits of the range (0 and 1023).

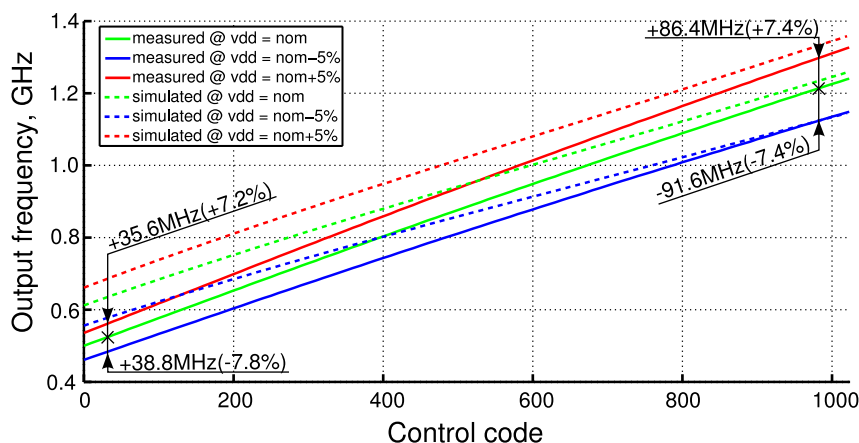


Figure 3.42: Measured and simulated output frequency versus input control code for different supply voltages: nominal and changed by $\pm 5\%$

3.8.2 Chip-to-chip variations

Fig. 3.43 demonstrates the results of characterization of chip-to-chip variations of the code-frequency characteristic. It depicts tuning curves for 7 samples of the fabricated circuit. The

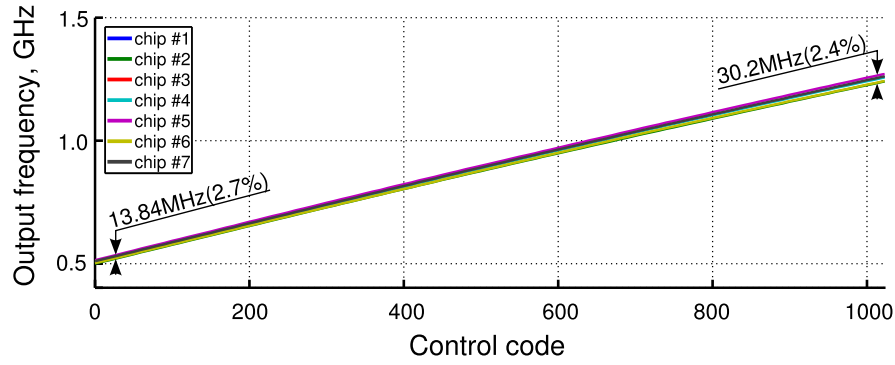


Figure 3.43: **Measured tuning curves for 7 samples of fabricated circuit at VDD=1.1 V**

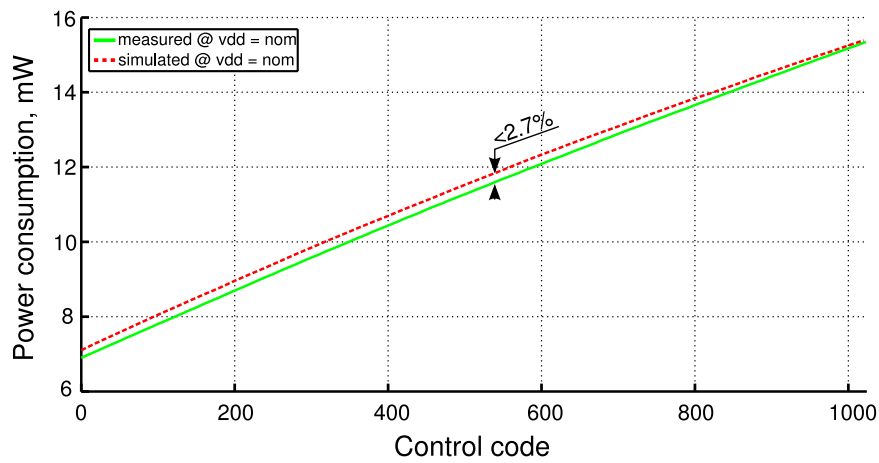


Figure 3.44: **Measured and simulated power consumption versus input control code at VDD=1.1 V**

maximal variation of frequency in overall frequency range does not exceed 2.7 %. Such a small variation is explained by the choice of non-minimal dimensions of the active transistors of the oscillator delays.

The chip-to-chip variation provides an upper bound of local variations between the DCOs used in a ADPLL network. It means that even in the worst cases, the fabrication errors impact on the DCO frequency of the network can easily be compensated by the ADPLL network structure thanks to the large tuning range of the DCO.

3.8.3 Power consumption

Fig. 3.44 depicts the simulated and measured code-power characteristics of the DCO. The two lines are in very good agreement. The power consumption of the DCO does not exceed the calculated value of 15.6 mW. The average power consumption of the DCO core is 6.2 mW/GHz.

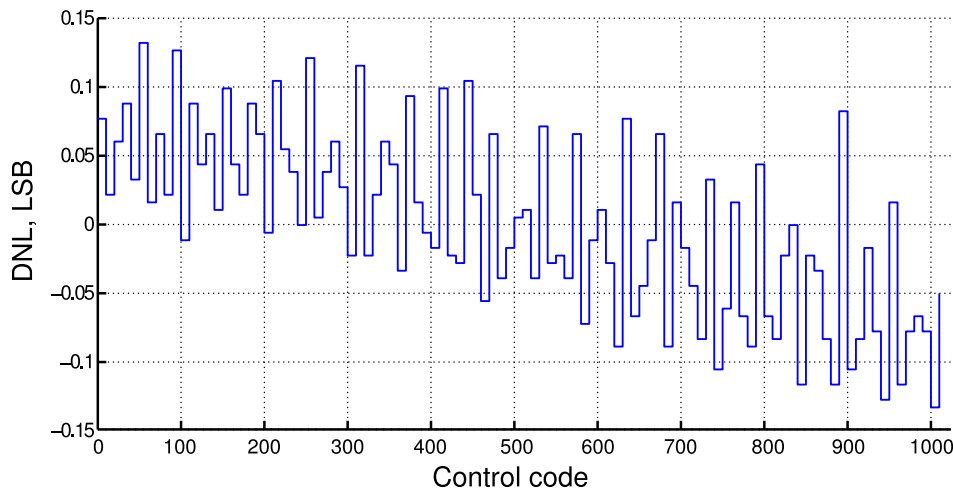


Figure 3.45: Measured differential nonlinearity of the circuit

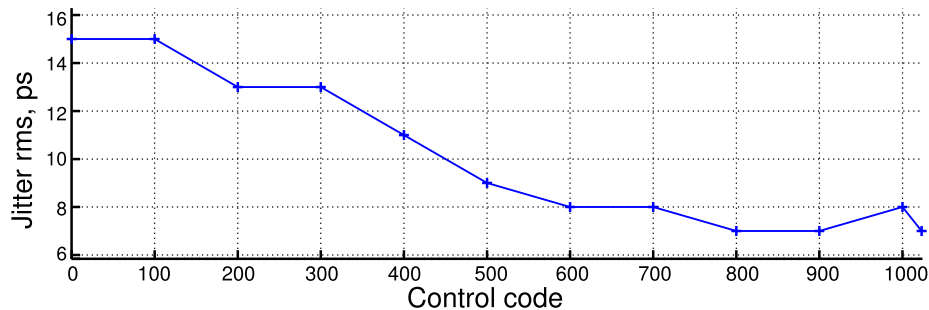


Figure 3.46: Measured rms jitter versus control code at VDD=1.1 V

3.8.4 Linearity

As it can be observed in Fig. 3.42, the DCO has monotonous and linear code-frequency characteristic in the whole frequency range. Fig. 3.45 demonstrates the deviation of the measured tuning steps from the average 1 LSB frequency step. This characteristic is also known as differential nonlinearity (DNL). The maximal deviation observed in Fig. 3.45 does not exceed ± 0.133 average LSB.

3.8.5 Jitter characteristics

Fig. 3.46 shows the measured total jitter (rms) of the free running oscillator within the whole frequency range. In fact, the results of these measurements include not only the jitter of the oscillator, but also the jitter from the various on-chip buffers and pads, jitter floor of the measurement tools, and other parasitic contributors related to off-chip measurements. As one can see, below the nominal frequency the jitter exceeds the specified in Tab. 2.1 clock timing error limit. However, it is expected to be much lower in a closed loop of ADPLL and thus it is expected to satisfy the specified parameter (< 10 ps).

Table 3.3: DCO chip performance summary

<i>Parameter</i>	<i>Value</i>
Central frequency	1.744 GHz
Output divided frequencies	872/218/109 MHz
Tuning range	999~2480 MHz
Gain	1.482 MHz/LSB
Phase noise ¹	-91.12 dBc/Hz @ 1 MHz for 2 GHz
Jitter (rms)	7~15 ps
Supply voltage	1.1 V
Power consumption	15.6 mW @ F_{max} or ~ 6 mW/GHz
Area ²	$\sim 6324 \mu\text{m}^2$

¹Post-layout simulation²DCO core only

3.9 Conclusion

This chapter presented the design of a DCO for distributed digital clock generation, on the basis of the project specifications. A ring CMOS oscillator with 999–2480 MHz tuning range and highly linear code-frequency characteristic has been designed, fabricated and tested in 65 nm CMOS technology. The implemented structure is a 7 stage ring oscillator with an array of the frequency tuning inverting cells connected in parallel and distributed over all 7 stages. These tuning cells are controlled by the digital signal, making the overall structure highly compatible with the digital circuit design tools.

The systematic design methodology of this structure has been developed. It can be easily used for design of a similar DCO with different parameters; it can be incorporated in a digital CAD environment. The critical and frequently used DCO cells has been designed manually (custom layout design).

In order to speed-up the system-level simulations of the clock network, a precise VHDL model of oscillator has been built. It has a large number of options and parameters, which allows a good approximation of the behavior of the fabricated DCO.

A test circuit of the proposed oscillator has been fabricated in 65 nm process. The fabricated prototypes demonstrate a good agreement with target specifications. Finally, these results validated the designed DCO for use as a building block in integrated prototype of ADPLL network based clock generator.

The results of the research on this topic have been published and presented on the international conferences NEWCAS2010 and ISCAS2011 [81, 82].

Chapter 4

Digital blocks of the ADPLL

Contents

4.1	Introduction	81
4.2	Digital phase/frequency error measurement	82
4.3	Implementation of digital PFD	89
4.4	Digital loop control of ADPLL network node	99
4.5	Node programming mechanism	105
4.6	Conclusion	107

4.1 Introduction

Digital nature of most of the blocks is one of the significant advantages of all-digital PLLs. Digital blocks design requires much less resources than analog block design, since an automated design flow can be used. In addition, digital circuits offer a large freedom for reconfiguration and complex processing which can be beneficial in many contexts.

This chapter presents design and implementation of digital blocks of the PLL network. These blocks are *digital frequency comparator* and *error processing block*.

Section 4.2 describes commonly used techniques of digital phase measurement. It proposes a formal definition of digital phase comparator and explains the principle of the frequency comparator. The designed phase comparator is a phase-frequency detector; it is composed on a bang-bang phase detector (BB) for phase error sign measurement and a tapped delay line time-to-digital converter for the absolute phase error measurements. Design of these blocks is described in Section 4.3. Section 4.4 presents the structure and design of the error processing block which is composed of an error combining stage and a proportional-integral filter. Finally, in Section 4.5 we describe the programming interface for these blocks introduced to multiply the configurations in which the implemented prototype can be tested.

4.2 Digital phase/frequency error measurement

4.2.1 Digital versus analog phase comparators

In the literature, there is a confusion about the term "digital phase comparator". This term commonly means *a phase comparator built out of digital blocks*. Whereas appropriate at the time when the PLLs were only analog, it misleads nowadays when both analog and all-digital PLL exist. Hence, first we give a definition of digital and analog phase comparator. We recall that a digital signal is a discrete sequence of numbers belonging to a finite quantification grid.

An *analog phase comparator codes the phase error through an analog signal*. From this point of view, a XOR logic gate (Fig. 4.1) used as phase comparator is analog. Indeed, although the output signal level is discrete ('1' or '0'), the time at which the level change can happen is not quantified. The phase error is coded by the average level of the output pulse width modulated (PWM) signal, which may have any value between low and high levels, and hence, is analog.

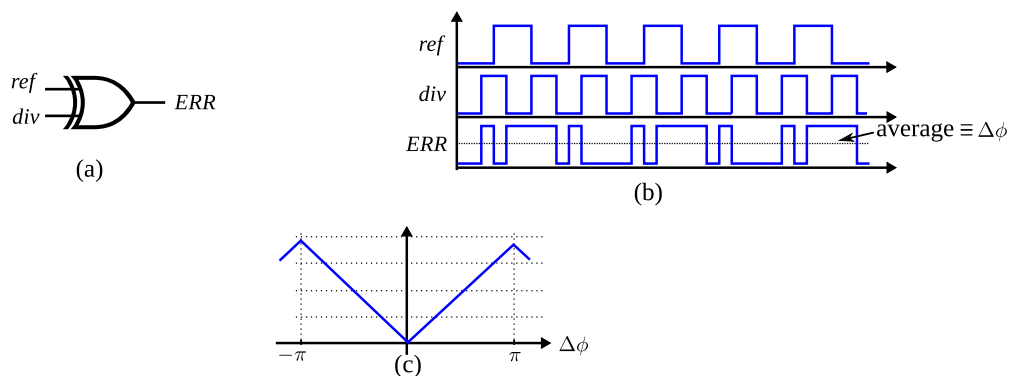


Figure 4.1: **XOR phase comparator:** (a) circuit diagram, (b) output average level - input phase error characteristic and (c) transfer function

The situation is similar to a very widely used phase comparator based on two flip-flops which allows to detect not only the phase error absolute value, but also the phase error sign (Fig. 4.2)[12, 66, 62, 15]. The phase error $\Delta\phi$ is coded as a difference between the average levels of the two analog PWM signals *DN* and *UP*.

Both mentioned phase comparators produce a non-regular sampling of the phase error. Indeed, the value of the phase error is known after the event on the lagging sequence. If the phase difference of the signals varies in time, the sampling is obviously non-regular, and the phase error value is synchronous with one of the input sequences depending on which signal is lagging.

Hence, the aforementioned phase comparators have discrete time (sampled) continuous level output, and hence the output signals are analog. The latter is usually processed by an analog filter: that emphasizes again the fundamentally analog nature of these blocks.

A *digital phase comparator provides a quantified and sampled phase error*. For the phase comparator output to be fully digital, the value of the phase error must be quantified. For this, an analog phase detector is supplemented with a ADC quantifying the phase error. Since in

most analog phase comparators the output phase value is coded through the duration of the output pulses, this ADC is usually a time-to-digital converter (TDC).

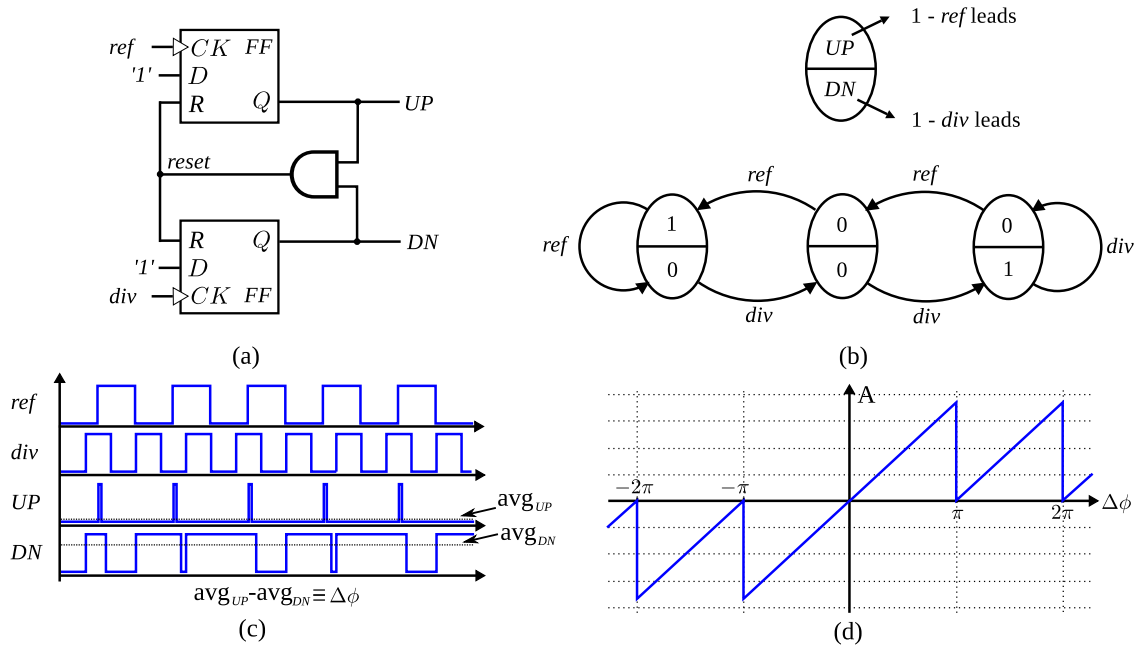


Figure 4.2: **Conventional phase comparator:** (a) circuit diagram, (b) state diagram (c) waveforms and (d) transfer function

4.2.2 Phase comparators versus phase-frequency detectors

Virtually all modern PLL circuits use phase comparators providing the sign of the phase error. They are usually called *phase-frequency detectors* (PFD), since they are able to report on the frequency error. In this subsection we present the principle of the phase error sign detection and we show that the phase error sign is directly related to the frequency error sign.

In this work we consider that the input signals whose phase/frequency difference should be measured are periodic sequences of events, for instance, rising fronts of a two-level quantified signal. For the presentation in this chapter, the following convention is used: we consider that the input signals of the phase comparator are named *ref* (signifies the *reference clock*) and *div* (signify the *divided feedback clock*), and when *ref* is leading, the phase error is positive.

Before making the demonstration, we present a generic behavioral algorithm which describes the detection of the phase error sign (Fig. 4.3).

The phase error sign detection is achieved with a four states finite-state automaton. These states are described by the values of two internal binary signals *MODE* and *SIGN*. A measurement cycle starts when the *MODE* signal is zero and the value of the *SIGN* signal has the value of the last measured error sign. In this state the automaton is ready for a new measurement and waits for an event on one of the inputs. When an event arrived on the input, the *MODE* goes to '1' and the *SIGN* value takes either '0' or '1', depending on the input which

registered the event. After that, the automaton waits for an event on the other input, ignoring all following events on the first input. As an event arrives on the other input, the *MODE* bit goes to '0' and a new measurement cycle can start.

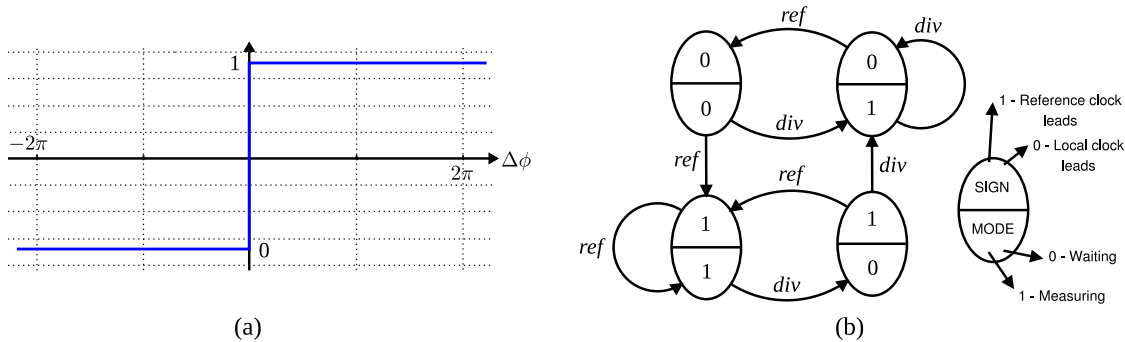


Figure 4.3: **The phase/frequency detector:** transfer function (a) and state diagram (b)

Hence, the *SIGN* signal value indicates which input receives an event first during the last measurement cycle. By convention, when *ref* is leading, *SIGN* is high (a positive phase error). Such a function is called in literature *bang-bang phase detection*, and the corresponding hardware block *bang-bang detector*.

Phase error sign detection. Let us consider a case of two input signals with identical frequencies but with a non-zero phase shift (Fig. 4.4). We consider that the phase measurement cycle starts at the time origin (i.e., at $t = 0$, the *MODE* state is '0'). Depending on the chosen time origin, the first and the following measurements may provide '0' or '1' values for the *SIGN* bit (Fig. 4.4). This fact is related to a 2π modular nature of the phase and the phase error and indicates that in the considered case the phase error sign has no objectively defined value.

However, it is possible to detect a *variation* of the phase sign, and it is what we use for the PLL operation. As shows Fig. 4.5, if one of the signal experiences a phase shift, the *SIGN* signal changes its value.

Frequency error sign detection. Now we present the mechanism of the frequency error sign detection with the described bang-band phase detector.

Suppose two sequences of events, $\{r_i\}$ and $\{d_j\}$, where r_i and d_j are the times at which the clock events occur. Suppose the events are periodical, what means that

$$r_i = T_r i + t_{0r}, \quad (4.1)$$

$$d_j = T_d j + t_{0d}, \quad (4.2)$$

where T_r and T_d are the periods of two sequences, and t_{0r} and t_{0d} is the time instant of the first event of the sequences after the time origine, and $0 \leq t_{0r} < T_r$, $0 \leq t_{0d} < T_d$.

We consider the operation of the automaton since $t = 0$, considering that at this moment the value of the *MODE* bit is '0' (at $t = 0$, the value of the *SIGN* bit does not matter for analysis). Obviously, depending on the initial phase of the signals, the first measurement cycle can give either '1' or '0', whatever the frequency difference is. We prove the following theorem:

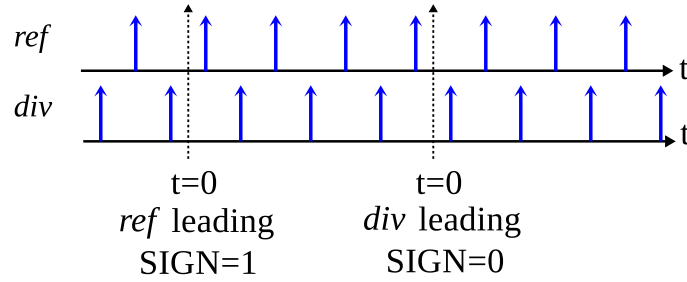


Figure 4.4: Two cases of initial conditions in phase/frequency detector

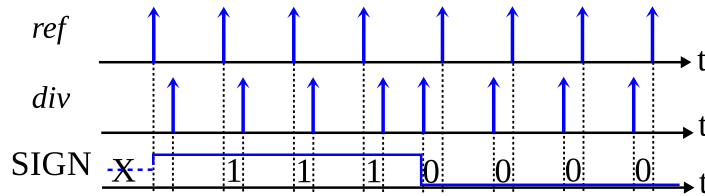


Figure 4.5: Detection of the phase error sign variation

For input signals described by Eq. (4.2), all phase error measurements cycles produce a zero *SIGN* signal if $T_r > T_d$ and one otherwise **maximum after n periods of the fastest sequence**. n is given by the formula

$$n = \text{ceil} \left(\frac{\min(T_d, T_r)}{|T_r - T_d|} \right), \quad (4.3)$$

where $\text{ceil}(\cdot)$ means *rounding toward plus infinity*.

The proof of this theorem is given in Appendix B. The theorem is proven by induction, on the sequence of the measurement cycles executed by the automaton. The proposed proof is for the case $T_r > T_d$ – which is enough given the symmetry of the system.

According to this theorem, a phase comparator measuring the phase error sign is able to measure the sign of the frequency error. The measurement needs a time whose value depends on the initial conditions and is inversely proportional to the difference between the sequence periods. Thus, the frequency error sign measurement is only efficient when the relative frequency difference is large.

4.2.3 The digital PFD architecture

Considering the finite-state automaton studied in the last subsection, we can note that duration of the *MODE* signal provides the time equivalence of the absolute value of the phase error. This signal can be used as an analog output of the phase detector, or passed through an ADC so to obtain a digital output. We used the latter approach to obtain a fully-digital PFD. Its architecture and input-output characteristic is given Fig. 4.6. The *MODE* signal is used to provide the code corresponding to the absolute value of the phase error. This code is then combined with the sign bit so to obtain a signed digital result. The PFD has two outputs, the phase error $e_{ri}[n]$ and $\bar{e}_{ri}[n] = -e_{ri}[n]$, for the reasons explained in Section 2.2.1.

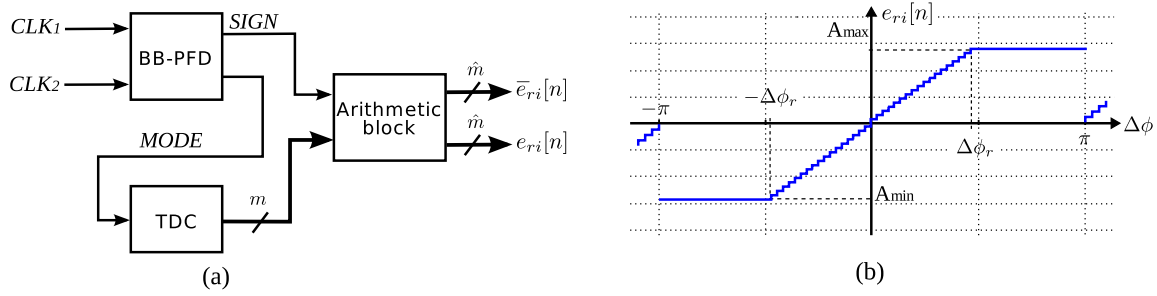


Figure 4.6: **Proposed phase/frequency detector for clock network:** (a) block diagram and (b) transfer function

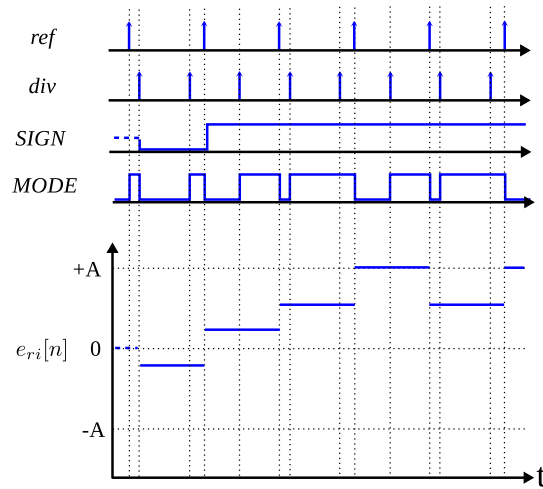


Figure 4.7: **Principle of operation of proposed PFD:** initial fault result about sign of the frequency error can be observed at the beginning; resolved at the next cycle of operation

The dynamic range of the PFD depends on the application context, as well as on the shape of the PFD transfer characteristic (linear or not). For this implementation, a linear characteristic of the digital PFD is chosen. In Chapter 2 we stated that a successful synchronization of PLL network requires a linear PFD transmission characteristic for small errors. Hence, the linear range of the PFD may not necessarily cover the range of $\pm\pi$: starting from certain value $\Delta\phi_r$ of input error, the output may saturate. That allows a reduction of the number of bits of the PFD output while keeping a high measurement precision.

The waveforms in Fig. 4.7 illustrate the operation of the circuit. The initial states of the BB-PFD and TDC are unknown and *MODE* bit has low logical level. Thus, independently of the initial state of the phase error sign, when the *ref* event happens, the *SIGN* becomes '0' and TDC waits for the *div* event, counting the time elapsed from the first *ref* event. After the *div* event happens, the error code is ready, and the *MODE* bit becomes '0' again, waiting for one of the events *div* or *ref*. It is very important to note that the information at the output of the proposed multibit PFD is ready only at the falling edge of the *MODE* bit, as it can be observed on the diagram.

4.2.4 Metastability problem

The model of phase-frequency detector presented in Section 4.2.2 has a fundamental drawback: it does not describe what happens when two sequences produce simultaneous events. In this case, the *MODE* signal has zero duration and the state of the *SIGN* signal is undefined. In a real circuit implementing the BB detector, such a context produces a metastability phenomenon. In addition to that, in the analog signal world, "zero" is a synonym with "small" and hence, the metastability occurs for small phase errors.

Physically, a metastability is a phenomenon which happens in all bistable dynamical systems with a continuous space of the state variables. It is related to an obvious presence of a *metastable* point which is an unstable equilibrium state situated between two stable equilibrium states [26].

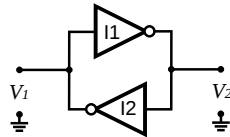


Figure 4.8: Simplest trigger circuit

The simplest digital circuit experiencing a metastability is a trigger (Fig. 4.8). The stable states correspond to $(0, V_{dd})$ and $(V_{dd}, 0)$ voltages at the outputs of two inverters. However, it can be seen that the circuit can remain in the state $(V_{dd}/2, V_{dd}/2)$. This state is unstable: a small perturbation (e.g. noise, temperature) steers the circuit to take one of the stable states. However, depending on the circuit geometry, the time needed to go out of the metastable state may be more or less long and the final stable state is unpredictable. Obviously, the metastability is a great danger for digital circuits. Unfortunately, the metastability phenomena is related to fundamental physical properties and can never be totally excluded, although its probability and the duration time can be reduced by design techniques.

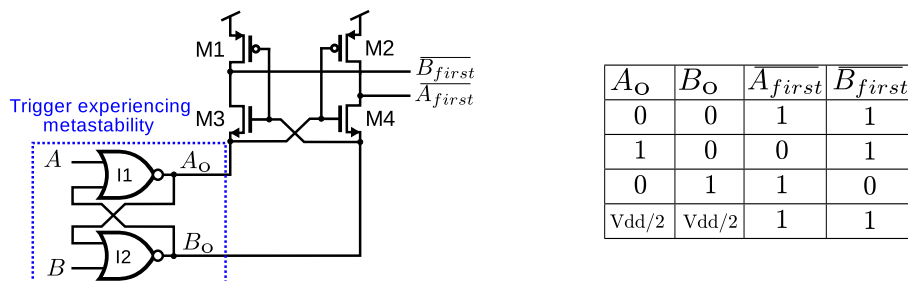


Figure 4.9: Proposed in [67] arbiter circuit

There are two approaches reducing the effect of metastability:

1. Reducing the time constant of the circuit. In this case, the time during which the system can stay in the metastable state is reduced.

2. Filtering the occurred metastability state. The principle is shown Fig. 4.9. If the flip-flop goes to a metastable state, the metastability filter composed of the transistors M1-M4 produces '1' until the difference between the output voltages of the flip-flop do not exceed the transistor (M3 or M4) threshold voltage. After that, the actual state of the flip-flop is propagated to the output. This circuit makes impossible a metastable state at its output, however, it generates a delay of unpredictable value, which is surely able to produce a metastability in the further stages.

4.3 Implementation of digital PFD

4.3.1 Bang-bang detector implementation

The bang-bang detector architecture used for this project is inspired by [67]. As mentioned in Section 4.2.2, the BB-detector measures the sign of the phase error and generates a time interval corresponding to the absolute phase error value.

The used detector architecture consists of two input flip-flops, the arbiter circuit filtering the metastability, the output buffer latch and reset logic (Fig. 4.10).

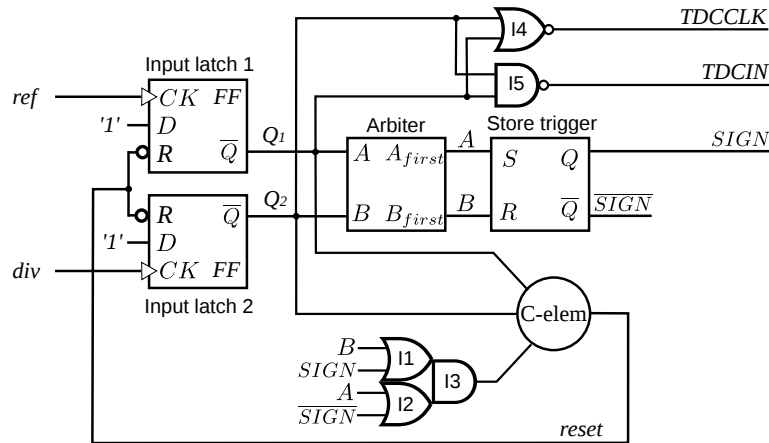


Figure 4.10: **Schematic diagram of the bang-bang phase/frequency detector:** taken from [67]

The input registers detect the input events and generate '0' at the outputs \bar{Q} as the events arrive. The principal role of the flip-flops is to detect the first event arriving on one of the inputs *ref* or *div* and to ignore all subsequent events arriving at the same input till the first event arriving on the other input. In the waiting state, the outputs \bar{Q} are '0', when events arrive, the outputs \bar{Q} return to '1'.

The arbiter plays two roles. The first one is the generation of the signals at the outputs A_{first} and B_{first} which are '1' or '0' depending on which input either *A* or *B* receives an event first. This information is then stored till the end of the measurement cycle. The second role is to filter a possible metastability resulting from simultaneous arriving of events at inputs *A* and *B*. As explained in Subsection 4.2.4, if the internal trigger of the arbiter goes into metastable state, both outputs A_{first} and B_{first} are at '1' level.

The bang-bang detector operates as follows. In the initial (waiting) state, the arbiter inputs are at '1' level, and the arbiter has '1' at both outputs, keeping the buffer trigger in the storage mode. The buffer trigger keeps the *SIGN* value detected from the last measurement cycle. When one of the signals Q_1 and Q_2 goes low, the trigger of the arbiter is set to a well-defined state, and after the second signal goes low, the arbiter trigger is in the storage mode. If the falling edges on *A* and *B* inputs of the arbiter arrive simultaneously or with a vary small time interval, the flip-flop of the arbiter can trap into a metastable state. In this case, the arbiter outputs '1' as explained in the Subsection 4.2.4. In this case, the output buffer trigger is in the storage state, and it outputs the sign value detected from the last measurement

Table 4.1: C-element truth table

A	B	C	Z
0	0	0	0
1	0	0	Z_{n-1}
0	1	0	Z_{n-1}
1	1	0	Z_{n-1}
0	0	1	Z_{n-1}
1	0	1	Z_{n-1}
0	1	1	Z_{n-1}
1	1	1	1

cycle. If the phase error is far from zero and if there is no metastability problem, the sign value is defined just after the arriving of the first event (after the flip-flop delay). Otherwise, the correct value of the phase error sign may be established on the output with a delay whose value is random.

The buffering RS trigger is needed to store the value of the phase error sign during the measurement cycles.

The truth table of the Muller C-element is given in Tab. 4.1. Its role is to generate a reset signal for the input flip-flops only when two conditions are fulfilled simultaneously:

- Events are detected on both inputs by the input registers
- The arbiter is out of the metastabile state, and the output of the buffer register is set to a well-defined state. This condition is verified by the combinatory logic composed of gates I1-I3

The reset of the input latches marks the end of a measurement cycle. The duration of this reset pulse is determined by the loop delay: C-element keeps the value '0' till the input registers reset their states and the arbiter's trigger comes to the state with '0' at both outputs (and '1' at the outputs of the metastability filter). This state corresponds to an initial state of the bang-bang detector.

As we said at the beginning of this subsection, the second role of the bang-bang detector is a generation of the *MEASURE* signal whose duration is equal to the time equivalent of the phase error. This time interval is then quantized by the TDC described in Subsection 4.3.2. The *MEASURE* signal is delimited by two events *TDCIN* and *TDCCLK* generated by a combinatory logic from the outputs of the input latches (Fig. 4.10). The event *TDCIN* marks the start of the time interval, *TDCCLK* marks its end.

The transistor-level schematic of the implemented D-triggers is given in Fig. 4.11. Since the D input of the input latches are always at high level, a simplified topology is used. The detailed schematic of the arbiter is given in Fig. 4.12. The Muller element (Fig. 4.13) is implemented as two cross-coupled inverters (memory element) and with a setup/reset circuit implementing pull-down and pull-up networks (respectively for all ones and all zeroes de-

tection). The resistance in the Muller C-element schematic is used to limit the setup current needed to change the state of the memory.

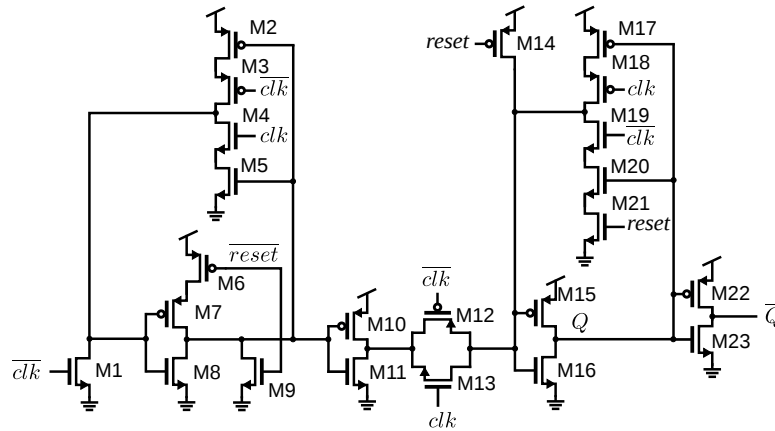


Figure 4.11: **Circuit diagram of the reduced flip-flop:** has only *CLK* input, reset and \bar{Q} output

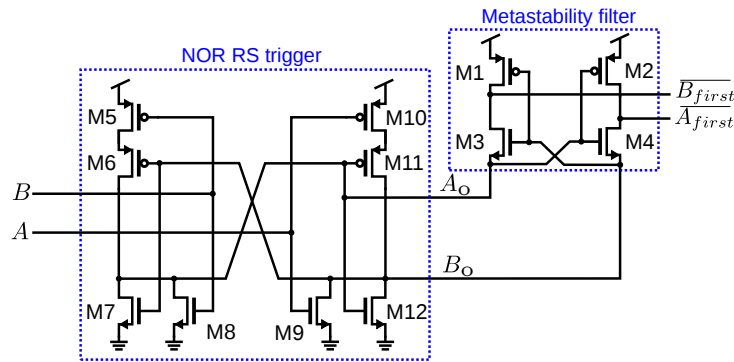


Figure 4.12: **Circuit diagram of the metastability filter with RS-trigger**

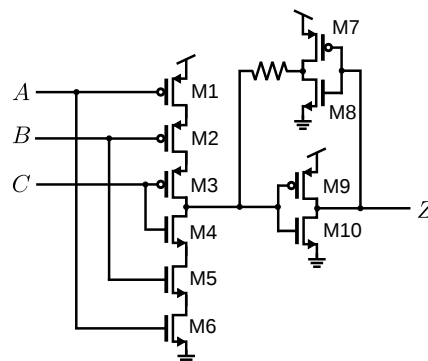


Figure 4.13: **Circuit diagram of the Muller C-element**

As explained in Subsection 4.2.4, the propagation delay of the arbiter is function of the input phase error. To illustrate this, we made a series of three modeling experiments in which the arbiter received input events outstanding by three different small time intervals at which

the metastability happens. Fig. 4.14 presents the signals at the outputs of the RS flip-flop (A_0 and B_0) and the metastability filter (A_{first} and B_{first}). Different curves on the same plots represent the waveforms corresponding to the simulations with three input time intervals.

We can note that the outputs can encounter the metastable states with different durations; in this case the metastability filter outputs present a delay corresponding to the metastability duration, and the output transitions are always free of metastability. The outputs of the metastability filter produce a transition when the flip-flop generates signals with voltage difference by at least one NMOS threshold voltage $V_{th,n}$. The delay of the arbiter tends to increase when the interval between the input events decreases.

Since the measurement cycle lasts till the metastability of the arbiter trigger is solved, the metastability has a direct impact on the delay of the bang-bang detector. Fig. 4.15 presents the relation between the input phase error and the overall PFD loop delay. Plots are obtained with an ideal Spice model of the circuit and with the netlist extracted from the layout accounting for the parasitic components. We can notice that the maximum of the delay and hence the longest metastability correspond to a non-zero input error. This fact highlights an offset in the measurement of the phase error sign. This offset is due to mismatches in propagation time of the signals $Q_1, A, SIGN$ and Q_2, B, \overline{SIGN} and caused by the asymmetry of the circuit layout.

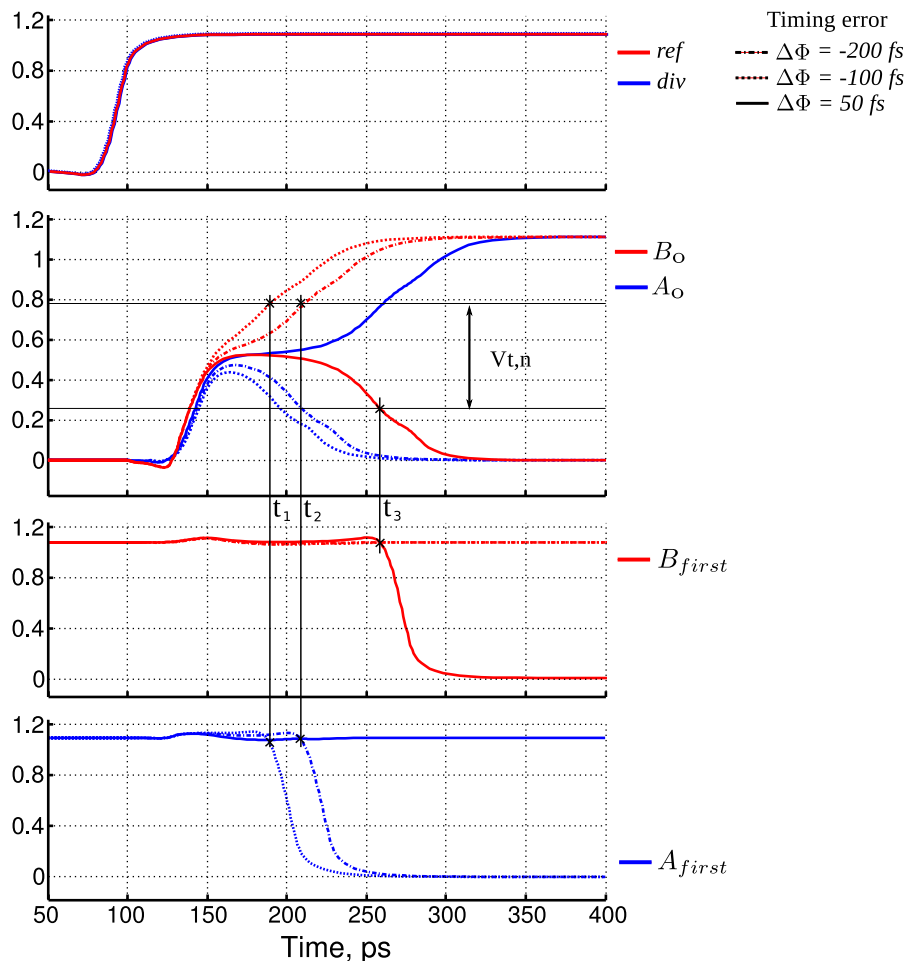


Figure 4.14: Delay of the arbiter as a function of timing error

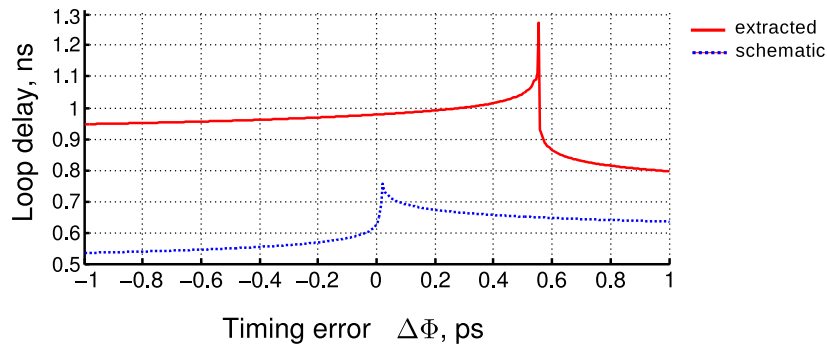


Figure 4.15: **Degradation of the bang-bang loop delay with smaller input errors:** results for ideal transistor model and for extracted from the layout schematic with parasitic components

Fig. 4.16 demonstrates the layout of the designed bang-bang detector and its 3D view. It has dimensions $15.2 \times 5.2 \mu\text{m}$. The height of the layout cell is twice the height of the standard cell.

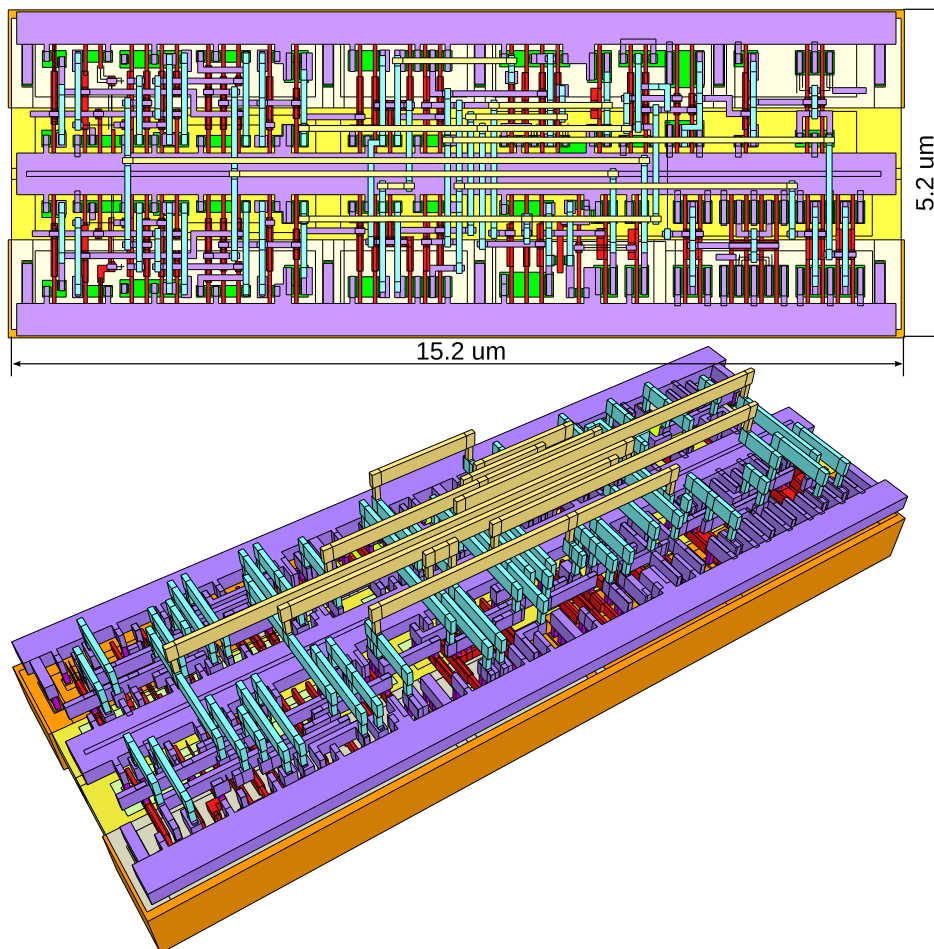


Figure 4.16: **Layout of the bang-bang detector:** CAD tool flat and 3D view

4.3.2 Time-to-digital converter

In this subsection we discuss the architecture of the TDC allowing a measurement of the absolute timing error ΔT between two clock signals.

A TDC is a linear ADC: the main parameters are the resolution (the step size) and the output dynamic range defining the number of bits of the ADC. The resolution of the TDC must be as high as possible; however, it is useless to increase it above the resolution allowed by the formula Eq. (2.4) related to the DCO operation. This formula gives 2 picoseconds. Simple TDC architectures using delay chains have typical resolution of the same order than the elementary buffers delay available in the technology. For CMOS 65 nm this is 30 ps. It is possible to improve the delay resolution by special design techniques (e.g. Vernier delay line), but it is not necessary in the context of our study. Indeed, the proposed clocking technique will be used in very advanced technologies (28 and 32 nm), where the natural delay is much smaller than 30 ps. Hence, for this project, 30 ps of TDC resolution τ_{TDC} specification is fixed.

As mentioned in Subsection 4.2.3, the dynamic range of the phase comparator ($-\Delta\phi_r, +\Delta\phi_r$) may not be large, since in normal operation the error amplitude is small. The dynamic method of desirable synchronized mode selection proposed in Section 2.3.2 is compatible with a low linear dynamic range of the phase comparator characteristic. By consequence, the parameter $\Delta\phi_r$ was chosen to be $\pi/4$. For the nominal output frequency 1 GHz divided by 4, it correspond to 500 ps. This range is obtained with 16.6 measurement steps: we round this value down to 15, so to allow the output TDC word to be coded by 4 bits. When output of the TDC is multiplied by the sign of the phase error, one bit is added, and the output PFD word has 5 bits.

The proposed architecture of the TDC is based on a tapped delay line, and is inspired by the architecture presented in [39]. The delay line implements a discrete set of time (delay) values so defining the output ADC grid. The time interval to be measured is compared with values of this set, and a corresponding digital code is produced. This architecture is similar to flash ADC, where measured signal is compared to discrete reference levels. Fig. 4.17(a) depicts its principle of operation. The time interval to be measured ΔT is defined by the starting and ending events given by the rising fronts of the signals P_1 and P_2 . The starting event is applied to the input of the delay line. The latches detect its propagation depth through the delay line during the measured time interval. In the architecture of Levine and Roberts (Fig. 4.17), output latches are clocked by the outputs of the delay line buffers, and the data inputs of all latches receive the ending event. In such a configuration, the valid output value is synchronous with none of the input events, and complex decoding circuit is needed. In our work, a slightly different topology is used (Fig. 4.18). The data inputs of the latches are connected to the output of the delay line buffers, and the clk inputs of all registers receive the ending event. For these reasons, the starting/ending events are called $TDCIN$ and $TDCCLK$ respectively (they are generated by the bang-bang detector, cf. Fig. 4.10). In this case, the registers store the valid output value synchronously with the ending event: that simplifies the use of the converted information in subsequent blocks.

How the duration of the interval to be measured ΔT is coded in the output word? If initially all output buffers had '0' values, after the rising front on the signal P_1 there is a propagation of '1' through the line. The depth of this propagation is proportional to the elapsed time. Hence, if the P_1 pulse is larger than the time interval to be measured, the outputs of the buffer can be considered as a digital chronometer giving the digitized time elapsed from the starting event. The latches produce the snapshot of the state of the delay line at the moment when the ending event arrives; so the latches contain the thermometer coded integer word representing the interval to be measured. The obtained thermometer coded word is then converted to a binary code and is used to generate the output phase error code of the PFD.

The implemented TDC uses 14 delays which provide 15 quantification levels (0-14 τ_{TDC}) including zero and the saturation level (Fig. 4.18).

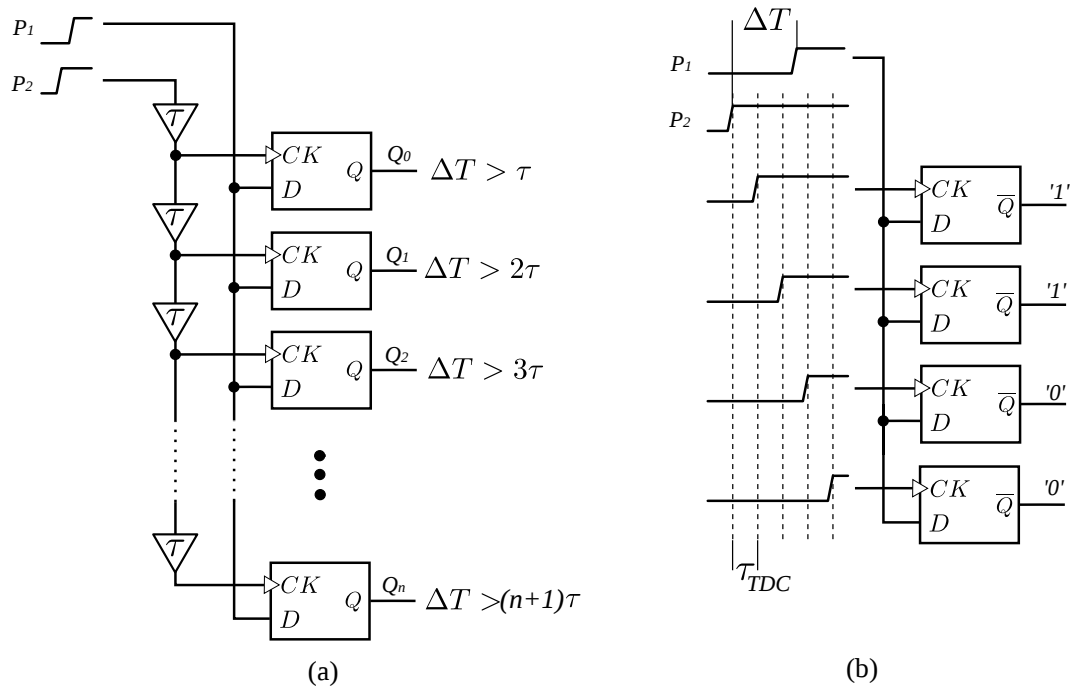


Figure 4.17: **Time-to-digital converter:** (a) block diagram and (b) principle of operation

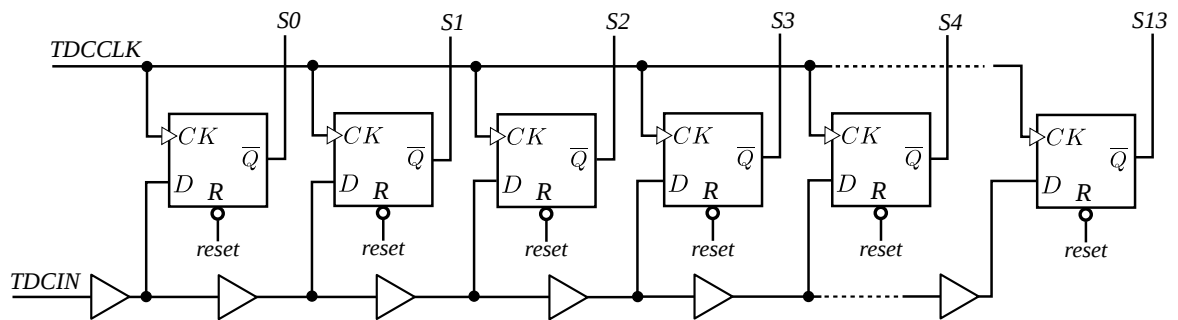


Figure 4.18: **Block diagram of proposed time-to-digital converter**

From the characteristic of the TDC (Fig. 4.19) three regions of TDC operation can be noted:

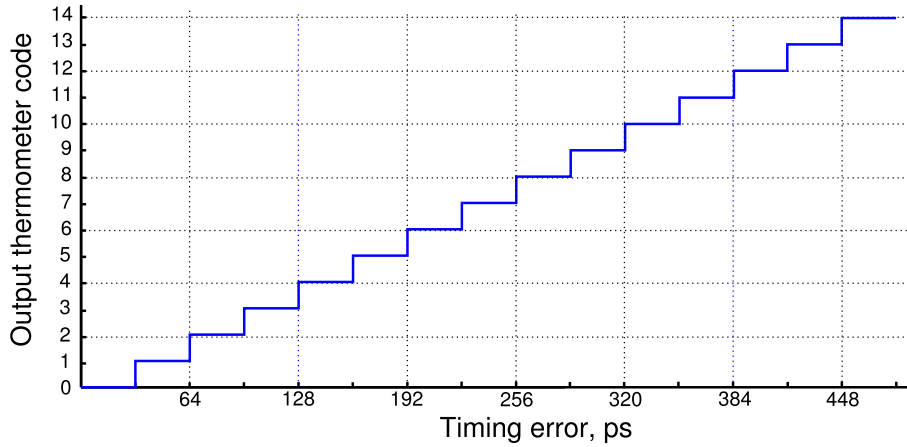


Figure 4.19: **Simulated transfer function of the designed flash time-to-digital converter:** measuring resolution 32 ps, range 32-490 ps

1. The time interval is less than the first delay element: the output register generates a word with all zeros;
2. If the time interval to be measured is greater than the delay of the first buffer but less than the total delay of the line: first bits of the register are at 1, and the others are at zero (Fig. 4.17(b));
3. The input time interval is greater than the overall line delay: all bits of the register are at '1'. The TDC is unable to discriminate the time interval values, and the TDC output is in saturation providing the maximal code.

In that way, the output of the parallel register $S_0 - S_{13}$ provides a thermometer code representing the measured and quantified value of the time interval in the range $(0, 14\tau_{TDC})$.

As for any real quantifier, there is a risk of metastability when the value of the interval to be measured is equal to a quantification step threshold. In such a case, the setup/hold condition may not be met for the output parallel register, and the registered value can have an error of ± 1 LSB.

Precise transistor-level electrical simulations provided a 32 ps quantification step of the TDC. Fig. 4.19 demonstrates the code/error characteristics obtained from the electrical simulations of the designed TDC. The TDC layout has been automatically generated from the schematic view using EDA tools (Fig. 4.20).

4.3.3 Implementation of PFD

The block diagram of the implemented PFD is presented in Fig. 4.6. The PFD is obtained by combining the TDC and BB detector outputs through an arithmetic block. The role of the arithmetic block is a generation of the binary coded 2^s -complement signed word corresponding to the measured phase error. Its absolute value is composed of the thermometer coded output word of the TDC and the sign provided by the BB detector, accordingly to the following formula:

$$PFD_{out} = (TDC_{out} + 1)SIGN. \quad (4.4)$$

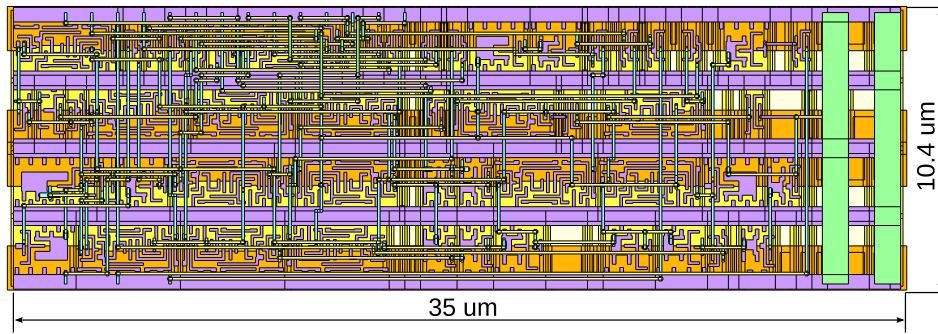


Figure 4.20: **Layout of the TDC employing standard cells**

The TDC outputs values from 0 (the error is less than τ_{TDC}) to 14 (the error is greater than 14τ). Hence, the PFD_{out} is in the range $[-15, -1] \cup [1, 15]$. For the reasons explained in Section 2.2.3, the PFD never outputs zero.

In order to convert the output thermometer code from the TDC and to complete it with information about the error sign issued by the BB detector, an encoder (arithmetic block) is used. The encoder issues a 5 bit 2-complement binary code corresponding to the measured phase error. The encoder has three inputs and two outputs: 14 bit receiving the signal from the TDC, $SIGN$ input receiving the sign information from the BB-detector, the *mode* input (cf. later in this subsection) and two output 5 bit binary signed words: the error $e_{ri}[n]$ and minus the error $\bar{e}_{ri}[n]$.

In order to increase the number of possible combinations for the network test, an additional mode programming input is added for the implemented PFD. This input called *BBen* (Bang-Bang enable) allows a reconfiguration of the PFD into a bang-bang mode. In this mode the TDC information is ignored and output of the PFD can generate ± 1 , depending on the measured sign of the phase error.

The timing of the PFD operation is presented in Fig. 4.21. Here, a typical example is given, when at the new measurement the sign and the value of the phase error are different from the preceding measurement. It should be noted that the PFD output is not latched and hence may output intermediate values (glitches) between the measurements, as specified in Fig. 4.22.

The final assembly of the sub-blocks for the PFD is done according to Fig. 4.6. The designed PFD has a total area of $53 \times 40 \mu\text{m}^2$, where PFD itself occupies $35 \times 20.4 \mu\text{m}^2$. The PFD is surrounded by a double active guard ring in order to isolate sensitive bang-bang circuit and TDC chain from the surrounding digital aggressors.

Fig. 4.23 presents the designed layout. The I/O ports are located on the border of the cell and placed at Metal3 layer. The supply network is organized so to provide the compatibility with the global floorplan of the clock network chip.

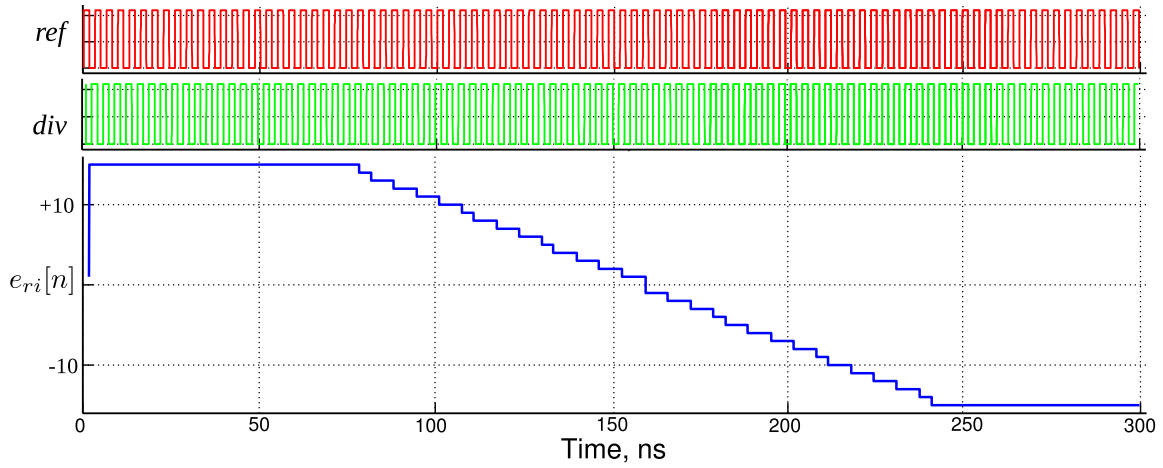


Figure 4.21: Simulation of the designed multi-bit PFD

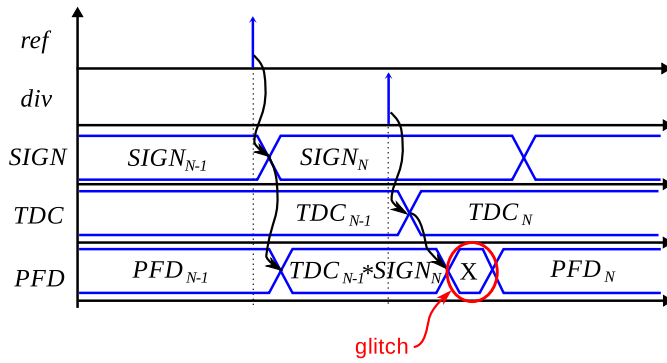


Figure 4.22: Signal flow diagram for proposed PFD:

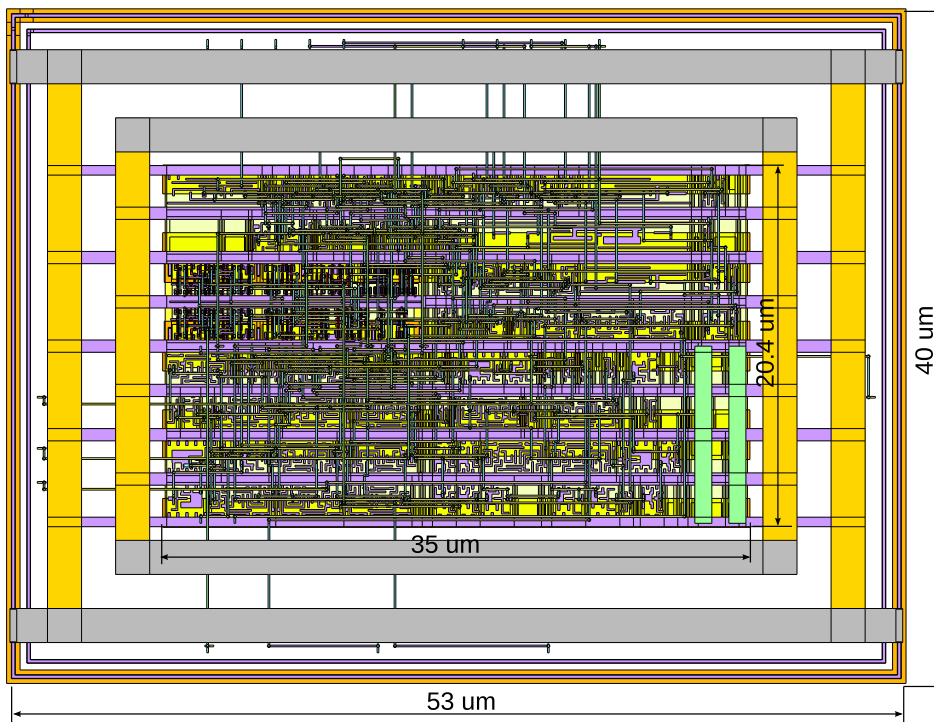


Figure 4.23: Layout of the proposed PFD

4.4 Digital loop control of ADPLL network node

This subsection describes the control block of a ADPLL network node. This block processes the phase errors between the local clock and 2, 3 or 4 neighbors issued by the corresponding PFDs. The purpose of this block is to generate a control word for the input of the DCO. It includes two cascaded elements: an error combining block and a proportional-integral digital filter (cf. Section 2.4).

A detailed schematic of the implemented digital processing block is provided in Fig. 4.24. The block receives four 5 bit input binary signed words, and generates a 10 bit unsigned word for the DCO control (the output of the adder ADD5). This signal is applied to the input of the DCO. The schematic in Fig. 4.24 includes the encoder converting this input binary code to the A, B and C signals necessary to control the DCO core. This encoder is described in Subsection 3.3.2.

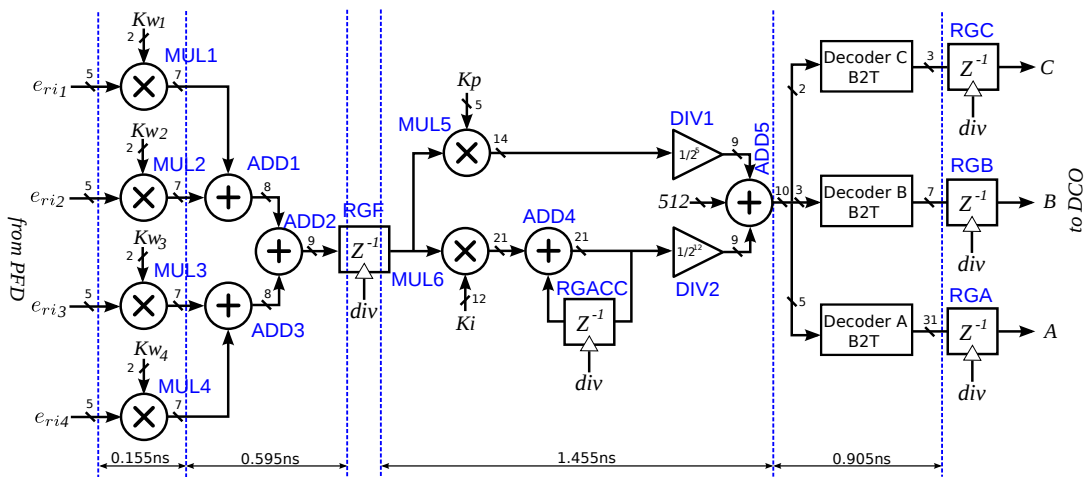


Figure 4.24: **Error signal processing block:** four input gain controllers followed by the four-input adder, PI filter and three B2T decoders. Shown timings are maximal in a worst conditions.

The digital block is sampled with the local divided clock (div signal). The design was made on the basis of the hypothesis of steady-state mode of the network node, in which the output frequency is nearly constant (cf. Section 2.4). In this case the sampling can be considered as regular. In this context, the block can be designed through standard design methodology and EDA tools. The combinatory paths included between two registers must have a delay not greater than the divided clock period. For this circuit, at maximal output clock frequency (1400 MHz) divided by four (the div signal), the minimal filter clock period is 2.8 ns. This provides the main timing constraint for the design.

4.4.1 Error combining block

This block receives four 5 bit 2-complement coded words representing the phase errors with neighbors. These values are passed through four gains blocks (multipliers by constant) MUL1-MUL4 and then summed using three two-inputs adders ADD1-ADD3. The

weighting coefficients of the gain blocks $K_{w1} - K_{w4}$ are programmable (cf. Section 4.5). Each gain can take independently a value in the set $\{0,1,2,4\}$. These values are powers of 2: the product is implemented as a binary shift, so introducing a very small delay of 155 ps, similar to the delay of two or three gates.

The four inputs adder operates with four 7 bit operands and produces a 9 bit sum. This adder is based on the carry look-ahead (CLA) architecture. Its delay is less than 595 ps in the worst case. The output of the adder ADD3 is buffered with a register RGF. This is necessary to comply with the aforementioned timing constraints and keep sufficient timing margin.

4.4.2 PI filter

The transfer function of the PI filter is given by Eq. (2.3). Theoretical investigations [35, 34, 4, 3] provided for the coefficients the following specifications: $\alpha \in \{1 \dots 0.03\}$, $\beta \in \{1 \dots 0.00024\}$

The calculations inside the filter are achieved in *fixed point* arithmetic. For the proportional part, the coefficient α is represented as a ratio of a programmable integer number and a power of 2 integer number:

$$\alpha = \frac{K_p}{2^5}, \quad (4.5)$$

where K_p is integer in the range $\{0, 31\}$. The 9 bit input word is first multiplied with K_p then divided by 2^5 . The fractional part of the result is then ignored, and only the integer part on 9 bits is applied on the input of the last adder ADD5.

The integral coefficient β is defined as a ratio:

$$\beta = \frac{K_i}{2^{12}}, \quad (4.6)$$

where K_i is in the range $(0, 2^{12} - 1)$. The architecture of the integrator path is given in Fig. 4.24. The main point of the integrator design is an appropriate choice of the size of the accumulator register RGACC. Here we summarize the integrator path design procedure.

A particularity of an integrator is an unlimited range of the output when this block is considered alone. Hence, a design of an integrator requires a specification on the range of the output value. This specification is defined by the system-level considerations and is practically ensured by a feedback. For the case of the PLL, the integrator defines the rough value of the frequency of the DCO; hence, the normal operation range is that of the DCO input (0, 1023) (cf. Chapter 3).

The integrator outputs unsigned integer values on 10 bits, receives input integer values on 9 bits, and the integral coefficient is between unity and a fraction of unity having up 12 binary digits after the binary point. To achieve an integration without loss of information, the accumulator must be able to store a number on $12+10=22$ bits. The 12 LSB are then ignored (this is represented by the block dividing by 2^{-12} in Fig. 4.24), and only the integer part on 10 bits is applied on the final adder ADD5.

After the global system reset, the integrator outputs 0, and the initial frequency of the ADPLL network node is the minimal one. If the target frequency is close to the maximal

output frequency, the accumulator should increase its value from 0, and the frequency acquisition time can be long. To reduce the maximal frequency acquisition time, the integrator is preset with an initial value of 512 (the middle of the scale). This is done by adding a constant offset of 512 to the output value of the filter.

The delay reduction is the key requirement during the digital loop control block design. In fact, a delay of PLL feedback is known to degrade the stability margins [17]. In particular, the delay of the proportional part is critical for that [17]. From the point of view of delay reduction, the designed architecture is disadvantaged by the programming feature which adds a complexity to arithmetic operators. For this reason, the whole delay of the combinatory path from the filter input to the DCO input register is greater than one clock period. For this reason, we use the register RGF and the registers RGA, RGB and RGC are introduced, allowing the combinatory operations to be spread over two clock cycles.

4.4.3 Implementation

The implementation of the digital processing block has been done using standard RTL-to-GDS design flow with help of EDA tools. Each block has been described in VHDL language (cf. Appendix A for details). The gate-level netlist was synthesized using standard cell library of STMicroelectronics. Fig. 4.25 presents the designed layout. The block has dimensions $116.8 \times 119.6 \mu\text{m}^2$.

The layout of the signal processing block is shown in Fig. 4.26.

4.4.4 ADPLL simulation results

To validate three designed blocks (the DCO, the PFD and the filter), a single ADPLL was designed and simulated. Fig. 4.27 demonstrates the results of the simulation of an ADPLL which uses transistor-level post layout extracted models of all blocks of the loop. This simulation verifies functionality and performance of the ADPLL. The parameters of this modeling experiments as well as the programmed coefficient values are summarized in a Tab. 4.2. Since only one ADPLL is implemented in this experiment, the digital processing block receives the error from only one PFD. Other processing block inputs are deactivated by setting their input gains $Kw_2 - Kw_4$ to 0.

The plots of Fig. 4.27 present the instantaneous frequency of the DCO (expressed in GHz and with the DCO input code), and the output of the PFD. After the power up, ADPLL is in *program mode* and generated by the script (cf. Appendix C) sources of voltage program the ADPLL to appropriate configuration. The output signals of the ADPLL blocks are irrelevant during this mode. Afterwards, the filter coefficients and the bang-bang mode bit have correct values and the *reset* signal is sent: the ADPLL starts the *frequency acquisition mode*. Once the divided output frequency is close to the reference frequency, ADPLL operates in a *phase acquisition mode*, where it adjusts the phase of the local clock according to the reference signal. In the *phase tracking mode*, the ADPLL output signal is in phase with the reference signal.

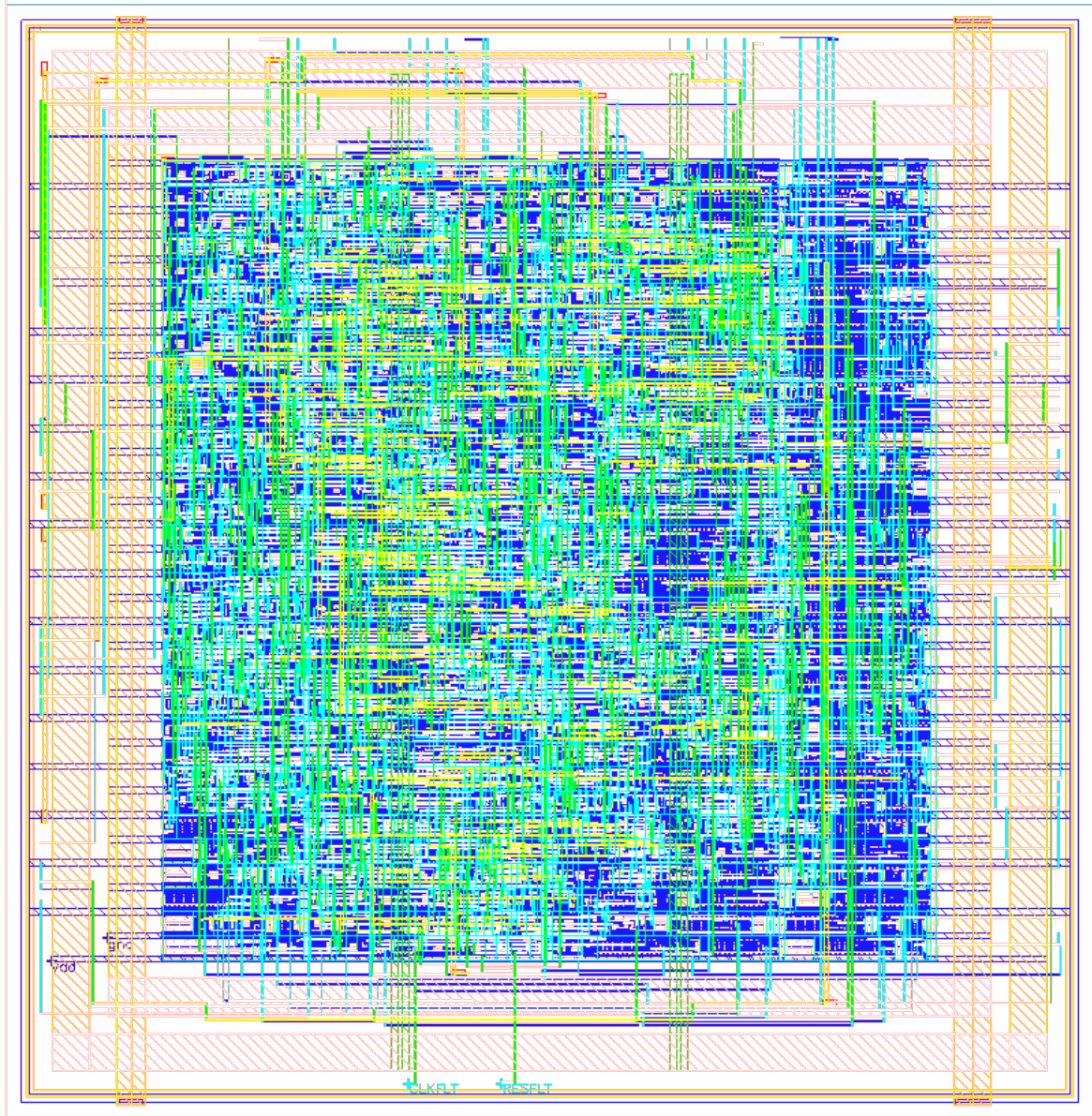


Figure 4.25: Layout of the digital signal processing block

Table 4.2: Simulation parameters and conditions

Parameter	Value
F_{ref}	276 MHz
α	1
β	0.00243
K_p	31
K_i	100
Kw_1	1
Kw_2, Kw_3, Kw_4	0
PFD mode	normal ($BBen='0'$)
F_{DCO} initial	2.05 GHz

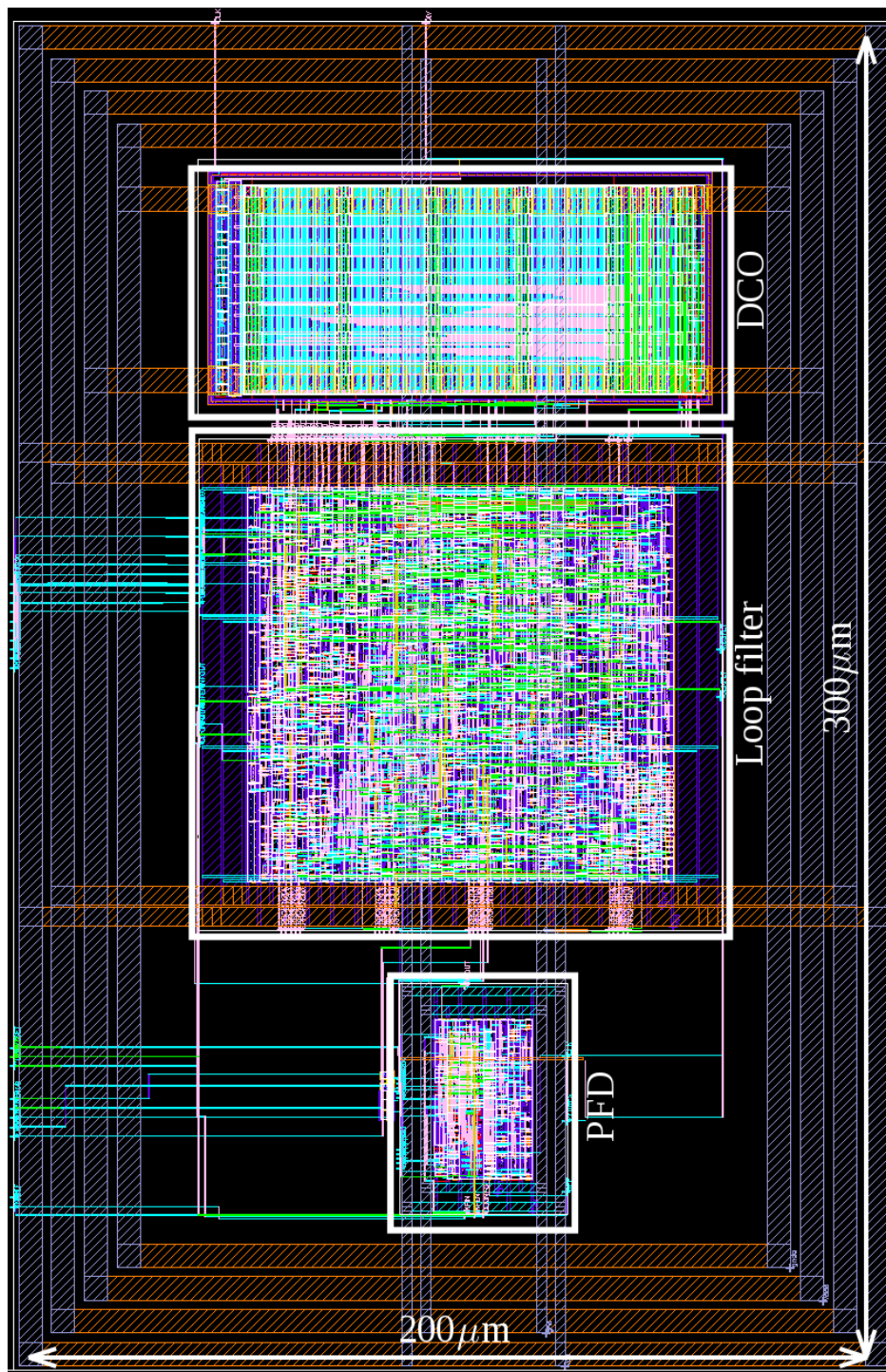


Figure 4.26: **Layout of the test ADPLL:** surrounded by double supply rings; v_{dda}/g_{nda} for separated and clean supply of the oscillator core; v_{dd}/g_{nd} for remaining digital blocks/cells

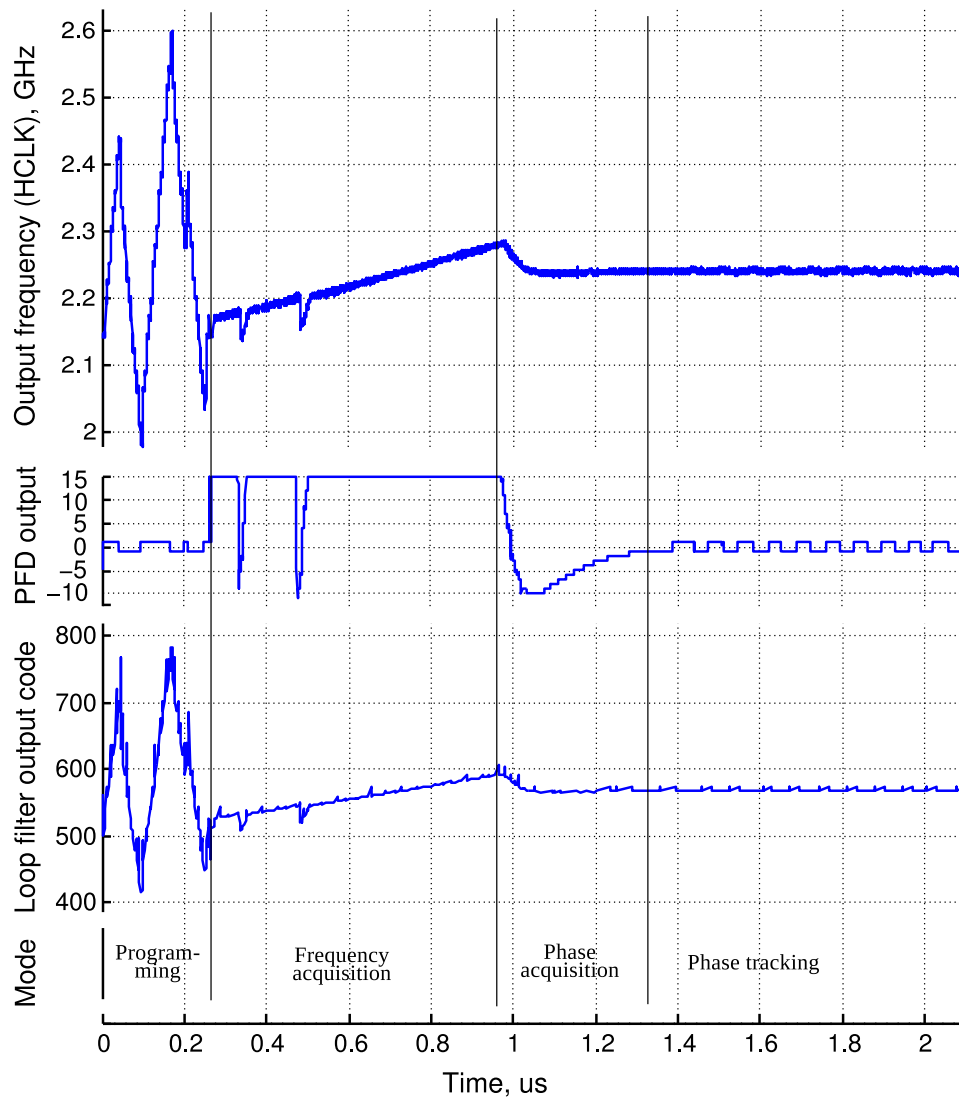


Figure 4.27: Simulation of the designed ADPLL with proposed oscillator, PFD and signal processing block

4.5 Node programming mechanism

A complex programming interface is needed for three reasons.

- For the testing purposes, parameters of the filter and the operation mode of the PFD are programmable. The programming interface should allow a definition of the network parameters before each startup of the network. These parameters are K_p and K_i in the loop filter, $K_{w_1} - K_{w_4}$ in the gain controller and $BBen$ bit in the PFD.
- The start of normal operation of the network is given by a reset signal: the reset should be generated after the network parameters are programmed.
- The dynamic mode selection technique chosen for our system (cf. Section 2.3.2) requires an interface allowing a *dynamic* (on-fly) reconfiguration of the network. The reconfiguration process must not perturb the operation of the network.

The programming interface must be flexible and extendable, so to be easily adaptable to the topology of the network (number of nodes, etc.). Since the number of the bits to be programmed is large (25 per node and 1 per PFD), serial interface should be used for the programming sequence transmission.

The designed serial programming interface(SPI) fulfills the above mentioned requirements. It is composed of two registers and one flip-flop (Fig. 4.28). The register XI0 is a serial-to-parallel converting register receiving the input series data on SDA_i and generating a parallel word at its outputs *parallel data out*. The actual values of the outputs of XI0 represent the values the bits to be programmed. However, during the input sequence reading, the *parallel data out* outputs have transient meaningless values. For this reason, the outputs of this register are not applied directly to the node block, but to the storage parallel register XI2 playing the role of a buffer. This loading is ordered by a global UPD signal.

This programming interface complies with the above mentioned specifications:

- The startup of the ADPLL network operation can be perfectly controlled: the global reset signal controlling the initial state of all registers of the network is not applied to the programming interface register. In this way, once the network is programmed, the global reset can be applied and the network starts from a well-established state.
- If the network parameters need to be modified during a regular network operation (for example, the weight coefficients K_{w_i}), the new values are firstly loaded in the register XI0. Then the UPD signal is sent, and after a delay, the network operates with updated parameters. This delay is short comparing to one clock cycle of the DIV signal. It is equal to the propagation time of the UPD signal and the setup delay of the parallel register (few tens of picoseconds).
- The programming interface is easily extendable by cascading (Fig. 4.29). The programming interfaces of two blocks are joined by connecting the last output bit of the register XI0 of one block to the SDA_i of the another block. The two blocks share the

same *UPD* and *SCK* signals. In this way, for any length of the programming sequence, the interface requires only three external pads: *SCK*, *SDA* and *UDP* (clock, data and load signals).

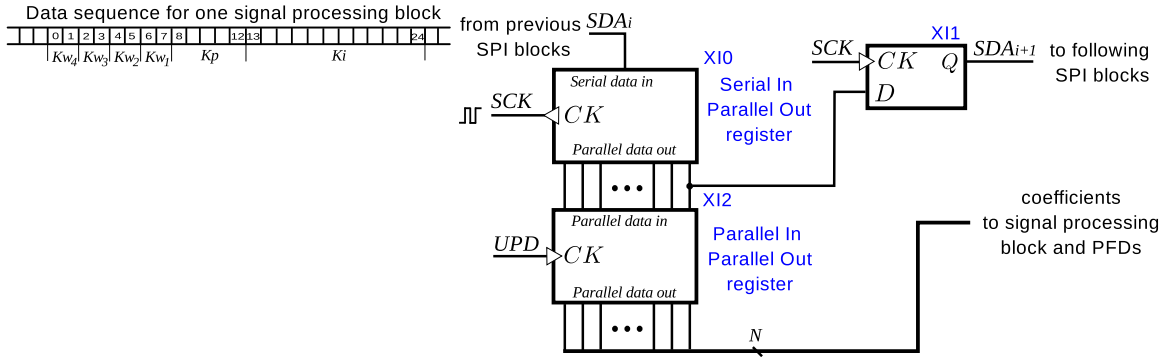


Figure 4.28: Schematic of the programming interface

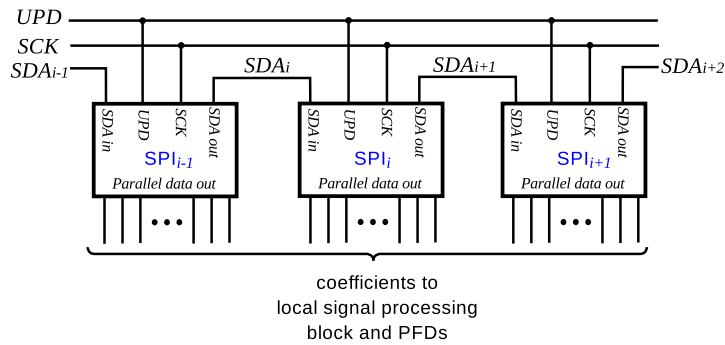


Figure 4.29: Cascading the programming interfaces of several blocks

4.6 Conclusion

This chapter presented the structure and design of purely digital blocks of the all-digital PLLS: the phase-frequency detector and the error processing block.

In the first part of this chapter we have reviewed and explicitly explained the fundamental difference between phase comparator in analog and digital PLLs. We have demonstrated the principles of phase and frequency error measurement in a digital PFD. As a result, an original phase comparator architecture has been proposed for the clock distribution network. The proposed phase comparator measures the absolute value of the phase error as well as the sign of the error; for this reason such device is called Phase Frequency Detector. Some results of the research on this topic have been published and presented on the international conference ISCAS2011 [29].

The second part of this chapter describes the proposed error signal processing block. It captures and processes the result of measurements from multiple PFDs and produces the control code for the local oscillator. Moreover, it offers the possibility to configure the network topology and coefficients of the loop filter. The drawbacks of such a flexibility necessary for the first ADPLL network prototype is large area and large processing delays. In the application-specific versions of the ADPLL network, this block can be optimized, primarily by reducing the number and the range of the programmable coefficients.

At the end of the chapter the programming interface is described.

In order to validate the designed blocks (the digital blocks and the DCO), they were assembled and configured as a single input ADPLL. The designed ADPLL was successfully validated by a post-layout simulation.

Chapter 5

Clock network implementation

Contents

5.1	Introduction	109
5.2	FPGA prototyping	111
5.3	Silicon implementation of the clock network	122
5.4	Measurement results	130
5.5	Conclusion	141

5.1 Introduction

The presented PhD project aims at the validation of theoretical investigation of an ADPLL clocking network carried out in the frame of the research project *HODISS* in **CEA-LETI** and **SUPELEC** laboratories [3, 34, 35, 36, 4], and a validation of the ADPLL network based clock generation technique. Such a validation requires a design of an ASIC with the network in an advanced CMOS technology. This chapter presents the main steps of this design, which includes FPGA prototyping of the architecture, design of the chip floorplan and layout.

The FPGA prototyping is a conventional verification step in the digital ASIC design flow. In particular, it allows a validation of the functionality of the designed systems or some of its blocks and detect potential problems and errors before the ASIC fabrication against very small additional design efforts. In our case, the FPGA prototyping allows :

- validation of the programming interface;
- validation of the design of the error processing block;
- undesirable synchronization mode elimination techniques;
- functional validation of the ADPLL network, in particular the synchronization in phase and undesirable synchronization mode elimination.

The key limitations of the FPGA prototyping concern the operation frequency and the impossibility to implement properly the mixed signal blocks: the PFD and the DCO. The

ASIC design aims at the validation of the feasibility of the architecture on a real submicron CMOS technology platform.

The implemented clock network has 16 nodes. It is configured as 4×4 Cartesian 2 dimensional mesh (Fig. 5.1). Each node is placed in the center of its own SCA and is composed of the ADPLL blocks whose design is addressed in preceding Chapter 3 and Chapter 4. Each SCA contains DCO and an error processing block, and PFDs are shared between the couples of neighboring SCA. The reference clock is injected at a corner node. The implemented network topology is chosen so to compare the designed clocking system with the unique implementation of PLL network based clock generator reported in literature [24].

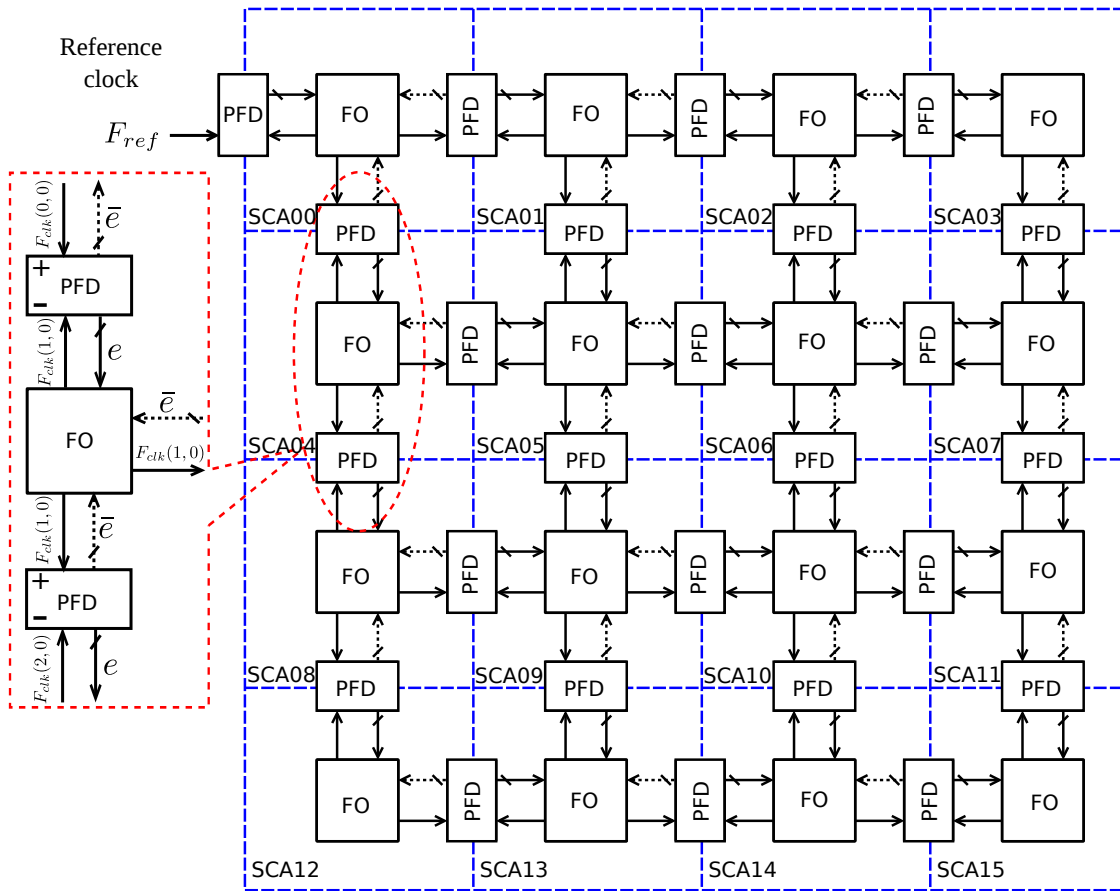


Figure 5.1: Structure of the implemented clock network

5.2 FPGA prototyping

As mentioned, a FPGA prototyping allows a partial validation of the ASIC design, through a verification of the essential functional features of the designed clocking network. The validation is only partial because of limitations inherent to FPGA prototyping. Despite these limitations, we have already developed the VHDL models of ADPLL blocks for the behavioral simulations (cf. Chapter 2) which could be synthesized and reused at this step of validation with minor modifications (e.g. loop filter and BB-PFD).

Here we summarize the principal issues limiting the truthfulness of the FPGA prototype comparing to an ASIC:

- **Difficulty of implementation of analog/mixed features.** From the point of view of ADPLL applications, the key limitation of the FPGA prototyping flow is an impossibility to implement a pure continuous-time delay. In particular, the VHDL code with time definitions

```
A <= B after delay
```

which are used for behavioral modeling the TDC is not synthesizable (cf. Section 2.5). Hence, the TDC which uses continuous-time delays of the gates in ASIC cannot be implemented through a standard FPGA design flow. For instance, described earlier behavioural VHDL model of the DCO uses expression like

```
clk <= not clk after period
```

where *period* is a time-type variable, which cant be synthesized for FPGA environment. Moreover, the designed DCO needs non-standard CMOS tri-state inverters (cf. Subsection 3.2.3) which are not available in FPGA. By consequence, the FPGA implementation of the ADPLL requires a different architecture for the TDC and DCO, and the silicon design of these two blocks cannot be verified through FPGA prototyping.

- **Frequency.** The typical clock frequency of the commercially available FPGA chips is in a range of hundreds of MHz: that is obviously insufficient for the prototyping of an oscillator generating gigahertz frequency signal. By consequence, a downscaling of the frequencies is needed for the FPGA prototype.

The frequency downscaling of the FPGA prototype of the ADPLL network follows the following principle: all timing parameters of the system are scaled linearly with the same scaling factor α :

$$\begin{aligned} f^{fpga} / f^{asic} &= \alpha, \\ t^{fpga} / t^{asic} &= 1/\alpha. \end{aligned} \quad (5.1)$$

Here f and t denote the frequencies and the time parameters of the FPGA and ASIC systems.

The next two subsections present the design of the DCO, of the TDC and the procedure of choice of optimal frequency scaling.

5.2.1 Synthesizable DCO

The straightforward DCO architecture including a DAC and a VCO (Fig. 3.1) cannot be implemented in an FPGA platform.

However, the well-known direct digital frequency synthesis (DDFS) technique can be used to synthesize a fully digital DCO, which, in the context of ADPLL, behaves exactly as a mixed-signal DCO whose design is presented in Section 3.3.

The DDFS consists of two steps. The first step is the synthesis of the digital phase for the oscillator. A digital phase is a digital sequence $\{s_i\}_{i \in \mathbb{N}}$ defined in the discrete time given by the external clock with period T_{clk}^{fpga} . The sequence $\{s_i\}$ provides a saw-tooth digital signal with period equal to the period of the signal to be synthesized (Fig. 5.2).

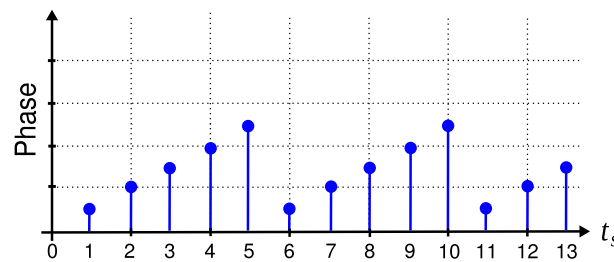


Figure 5.2: **Repeating discrete ramp function in the DDFS**

The second step is the use of the sequence values to generate a function $f(s_i)$ defining the waveform of the periodic signal. The output signal is digital, and can be converted to an analog representation with a DAC if required. In a contrast with the mixed-signal DCO, the output signal of the DDFS oscillator is synchronous with the clock defining the digital phase sequence, and the period of the obtained DCO signal T_{DCO}^{fpga} is necessarily a multiple of the period of this clock $T_{clk DCO}^{fpga}$. If $T_{DCO}^{asic} \ll T_{DCO}^{fpga}$, the DDFS DCO is a fair model of a proper mixed-signal DCO.

For the ADPLL prototyping, only the first step of the DDFS is needed. Indeed, the ADPLL senses the events which mark the beginning of the DCO periods, and ignores the waveform generated by the DCO.

The phase synthesis is achieved with a programmable counter/divider receiving the FPGA clock signal and generating an increasing digital phase sequence at its output. When the counter output reaches the maximal value, the count starts from zero, so generating the phase waveform as in Fig. 5.2. The ADPLL DCO may use the overload output of the counter which marks the end of the current period and the beginning of a new period. In such a context, the counter is used as a programmable frequency divider whose division coefficient is equal to the desired period of the output sequence (measured in number of $T_{clk DCO}^{fpga}$). This period (the duration of one teeth of the saw) can be modulated by loading the initial output value K of the counter at the beginning of the new cycle (Fig. 5.3(a)). In this way, the period of the output sequence is given by:

$$T_{DCO}^{fpga} = T_{clk DCO}^{fpga} (2^N - K) \quad (5.2)$$

where N is the number of bits of the counter.

The corresponding period-code and frequency-code DCO characteristics are given in Fig. 5.3(b,c). As seen in the figure, the code-frequency characteristic of FPGA DCO is nonlinear, in a contrast to that of ASIC DCO. However, it is not critical for the ADPLL prototyping, since the target ADPLL operation mode is when the output frequency is settled, and the input DCO code varies in a small range, so to correct the residual phase error. Consequently, the fluctuations of the frequencies in a network will be small and the DCO characteristic can be considered as locally linear (cf. Fig. 5.3(c)).

However, because of nonlinearity of the FPGA DCO frequency-code characteristic, it is not possible to ensure a linear scaling (Eq. (5.1)) at all frequencies. Hence, the downscaling is defined for a particular frequency considered as nominal. Arbitrarily, we define that the the nominal divided frequency of the ASIC DCO F_n^{asic} is in the middle of the tuning range, is equal to 218 MHz and corresponds to the code 512.

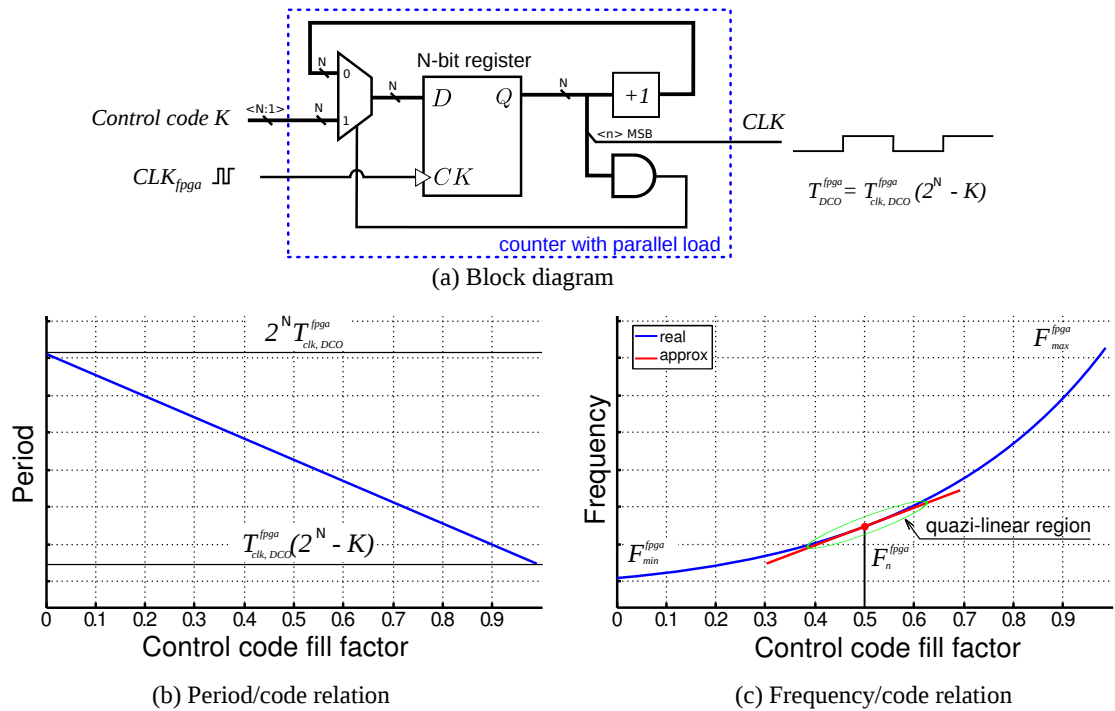


Figure 5.3: **Schematic diagram of the proposed FPGA implementation of the oscillator:** (a) schematic, (b) period/code and (c) frequency/code characteristics

The FPGA-prototyped ADPLL must emulate the ASIC system, however, it is clear that the FPGA based clock generator cannot output signals at the same frequencies as the original system. The maximal internal FPGA clock frequency is a hundred of megahertz, and, as stated, the synthesized frequency must be much lower than the DDFS DCO clock frequency. Hence, a frequency downscaling is necessary. The following calculation allows an identification of the necessary downscaling of the DCO frequency so to provide a representative model of the ASIC DCO.

Given that the period tuning step of the FPGA DCO is equal to the period of the FPGA clock generator T_{DCO}^{fpga} used for the DCO, the frequency step of the FPGA DCO ΔF_n^{fpga} is

related to the FPGA DCO nominal output frequency F_n^{fpga} by the following relation :

$$\Delta F_n^{fpga} = F_n^{fpga} - \frac{1}{\frac{1}{F_n^{fpga}} + T_{DCO}^{fpga}}. \quad (5.3)$$

At the same time, we want the relation between the nominal frequency and the frequency gain to be the same in the FPGA and in the ASIC DCO (cf. Eq. (5.1)). Hence, the second relation between F_n^{fpga} and ΔF_n^{fpga} is:

$$\frac{F_n^{fpga}}{\Delta F_n^{fpga}} = \frac{F_n^{asic}}{\Delta F_n^{asic}} \quad (5.4)$$

These two equations has one free parameter which is T_{DCO}^{fpga} . It cannot be superior to the minimal period of the FPGA internal clock, whose typical frequency is a hundred of MHz, depending on the used FPGA platform. Its actual value depends on the constraint about the scaling of the TDC, which is discussed in the next subsection.

We want the FPGA based DCO to have the same number of steps as the ASIC DCO. Hence, the maximal and minimal frequencies of the FPGA DCO are given by

$$F_{min}^{fpga} = \frac{1}{1/F_n^{fpga} + 511T_{DCO}^{fpga}} \quad (5.5)$$

and

$$F_{max}^{fpga} = \frac{1}{1/F_n^{fpga} - 512T_{DCO}^{fpga}}. \quad (5.6)$$

It should be noticed that since the frequency-code characteristic of the FPGA based DCO is nonlinear, the scaling (5.1) is not valid for the the two latter parameters.

5.2.2 FPGA based TDC

The proposed TDC for FPGA prototyping is a digital chronometer counting the number of external high frequency clock cycles during the interval to be measured. The interval length is specified by the MODE pulse duration (Fig. 5.4). The pulse is applied to the counter enable input EN . The counter output increments till the end of the pulse. The XI1 register stores the result synchronously with the falling edge of the input pulse. At a small delay time, the $RESET$ input of the counter receives an active level, so preparing the counter to a new measurement cycle.

Comparing the counter-based TDC with a delay line based TDC, it should be noted that the both quantize the input interval duration. In the case of the counter-based TDC, the quantization step is equal to the input clock period. However, in the counter-based TDC, the start of the input pulse is asynchronous. It means that the first quantization step can be less than one period of the TDC clock and relative accuracy will be limited by ± 0.5 LSB. On the contrary, in the delay-based TDC the first quantization step is always equal to the delay of the first delay element, since its operation is always synchronous with the input pulse. This

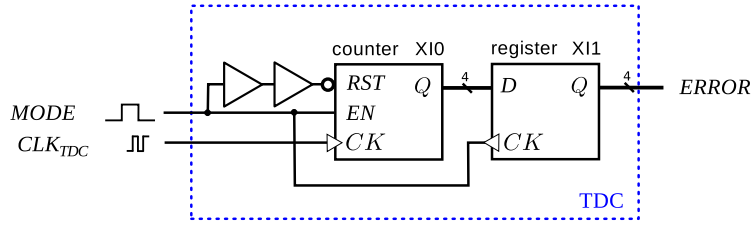


Figure 5.4: **Conventional phase detector**: (a) circuit diagram, (b) state diagram (c) waveforms and (d) transfer function

results in a different behavior for small errors inferior to the quantization step, which the delay line TDC may sometimes identify as '1', whereas the delay line based TDC always outputs '0'.

It is obvious that such a TDC cannot measure synchronization errors of gigahertz frequency signals, since the frequency of the TDC clock should be much higher than the frequency of the signals to be measured. Hence, again, a frequency downscaling is necessary. The only time parameter of the TDC is the quantization step τ_{TDC}^{fpga} , which must be related to the ASIC TDC quantization step τ_{TDC}^{asic} by the same scaling factor α as the DCO time/frequency parameters of both ASIC and FPGA systems:

$$\tau_{TDC}^{fpga} / \tau_{TDC}^{asic} = F_n^{asic} / F_n^{fpga}. \quad (5.7)$$

The free parameter of the Equation (5.7) is the clock period of the FPGA based TDC τ_{TDC}^{fpga} (or T_{clkTDC}^{fpga}). From the Eq. (5.7), Eq. (5.3) and Eq. (5.4) and from the parameters of the blocks designed for ASIC, the ratio between the clock frequencies of the DCO and TDC was calculated:

$$F_{clkDCO}^{fpga} / F_{clkTDC}^{fpga} = 8.915 \quad (5.8)$$

where $F_{clkTDC}^{fpga} = 1 / \tau_{TDC}^{fpga}$.

Hence, the F_{clkDCO}^{fpga} should be set at the maximal clock frequency available in the used FPGA platform, the other frequency parameters are calculated through the Eq. (5.7), Eq. (5.3), Eq. (5.4) and Eq. (5.8). The Tab. 5.1 summarizes the parameters of the TDC and of the DCO implemented for the FPGA prototype of the ADPLL network.

The complete description of the synthesizable VHDL code of the proposed DCO and TDC can be found in Appendix A.

5.2.3 Experimental results

The FPGA prototype of the network whose architecture is given Fig. 5.1 was implemented. The TDC and DCO blocks were designed as described in the last two subsections. The digital processing block and the programming interface were synthesized from the same code as that used for the ASIC design.

The system was implemented on the Altera evaluation test board with Cyclone II EP2 C70F672C6 chip. The basic information about this board, as well as the information about

Table 5.1: Parameters of the FPGA and ASIC implementations of the DCO

Parameter	ASIC	FPGA
F_n	218 MHz*	48.904 kHz
ΔF_n	185.25 kHz*	38.236 Hz
F_{min}	125 MHz*	34.916 kHz
F_{max}	310 MHz*	81.486 kHz
τ_{TDC}	32 ps	143 ns
$F_{clk_{DCO}}^{fpga}$	–	62.5 MHz
$F_{clk_{TDC}}^{fpga}$	–	7.01 MHz

* for DIV clock

measurement set can be found in Appendix E. The synthesis and implementation were performed in Altera Quartus II environment. The FPGA was programmed, and controlled on the fly by Altera USB blaster cable. The reference signal for the input of the clock network was synthesized by external generator with high precision and temporal stability.

The behavior of the system was observed with help of a digital oscilloscope at the points indicated in Fig. 5.5. The signals at each output of the PFD and DCO were observed. In such a way, we have the information about the phases of the local clock signals and by consequence, we know the phase error between them.

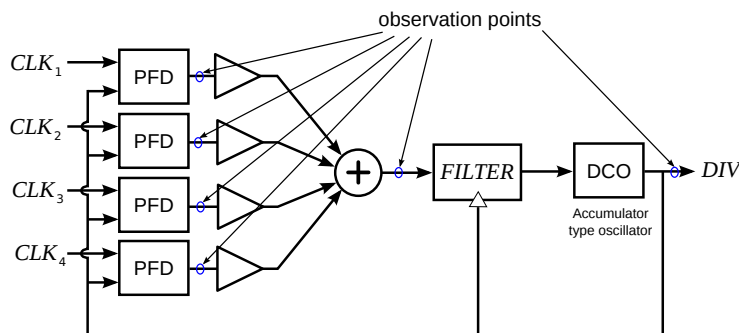


Figure 5.5: Block diagram of the node in a FPGA prototype with observation points

The first experiment was done with an ideal network, in which all DCOs have the same initial frequency (corresponding to 512 DCO code) and the same initial phase. In this configuration, the network converged to a synchronized state. The input (reference) clock frequency is set to 51.13 kHz and corresponds to DCO control integer code 566. The network has bidirectional configuration and with the following values of the signal processing block: gains $K_{w_1} - K_{w_4} = 1$, $K_p = 1$ and $K_i = 0.0028$. In Fig. 5.6 we can see that with these conditions, after the transitional process, all clocks have the same frequency and phase as reference clock.

However, this behavior cannot be representative of the real behavior of an ASIC network. Indeed, in VLSI implementation, due to the local variations and local drop of the supply voltage, the local frequencies of the nodes differ and the local oscillators do not start with the

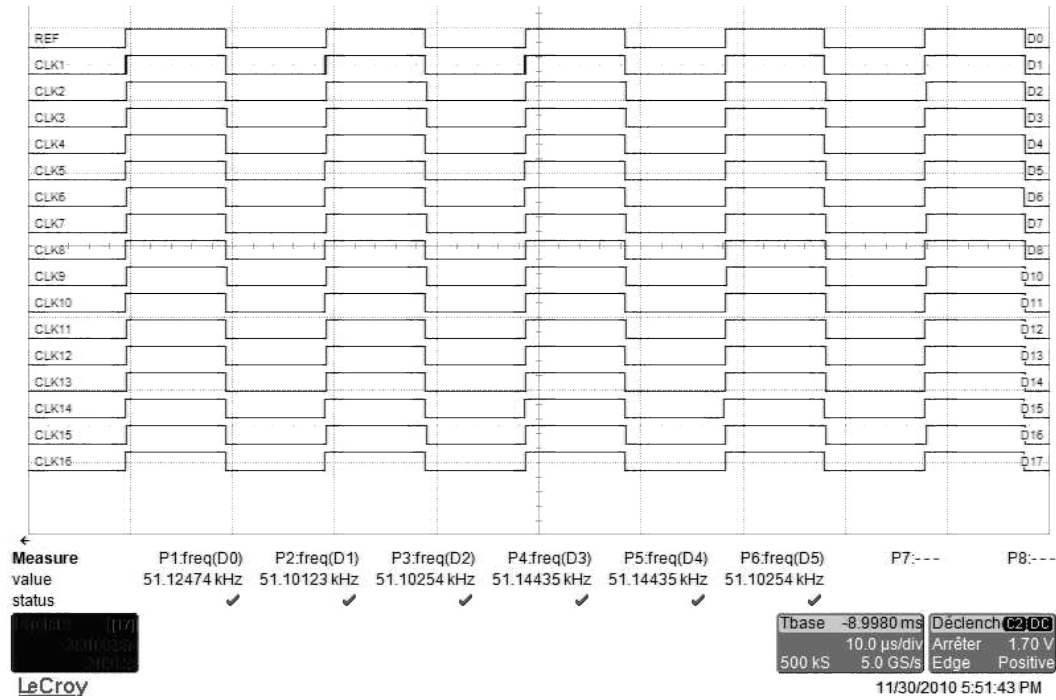


Figure 5.6: **Local clock signals together with reference:** bidirectional configuration, initial frequencies of nodes are equal

same phase. In particular, in the idealized configuration used in the experiment, undesirable synchronized modes (i.e. mode-locks) were not observed.

In the next experiment we randomized the initial conditions of the network, by diversifying the initial phases of the local oscillators. The initial frequencies of the DCO are set to the values distributed around the nominal value with a dispersion $\pm 20\%$. This dispersion is slightly superior to what was observed for the DCO frequency variation on the DCO test chips (cf. Fig. 3.43 of Chapter 3). Fig. 5.7 presents the outputs of the local oscillators after the transient process. A static phase error between clock signals is observed. Fig. 5.8 demonstrates the waveforms of the clock signals from the neighbors of the Node 10 and total error signal in Node 10 processed by its filter: whereas the phase errors are non-zero, the total error is zero, and the frequencies of all local oscillators are the same. This state is a typical for a mode-lock.

The next experiment aims at a verification of the proposed mode-lock elimination technique, which is based on a dynamic reconfiguration of the clock network. The set-up of the network is achieved in two stages. In first stage, the network operates in unidirectional mode. The phase/frequency information from the reference clock is propagated to the opposite corner of the network, without any feedback. The value of the filter coefficients remains the same as in previous experiments, the value of the control block gains are $K_{w_1}, K_{w_4}=0$ and $K_{w_2}, K_{w_3}=1$. Fig. 5.9 presents the operation in unidirectional mode after a transient process. There exist static accumulative phase errors. The phase errors increase from the corner node directly coupled with the reference signal, toward the opposite corner where error reaches the highest value. However, these errors are smaller than in a mode-lock state as one can observe by comparing Fig. 5.10 showing the neighbor clocks of the Node 11, and Fig. 5.8.

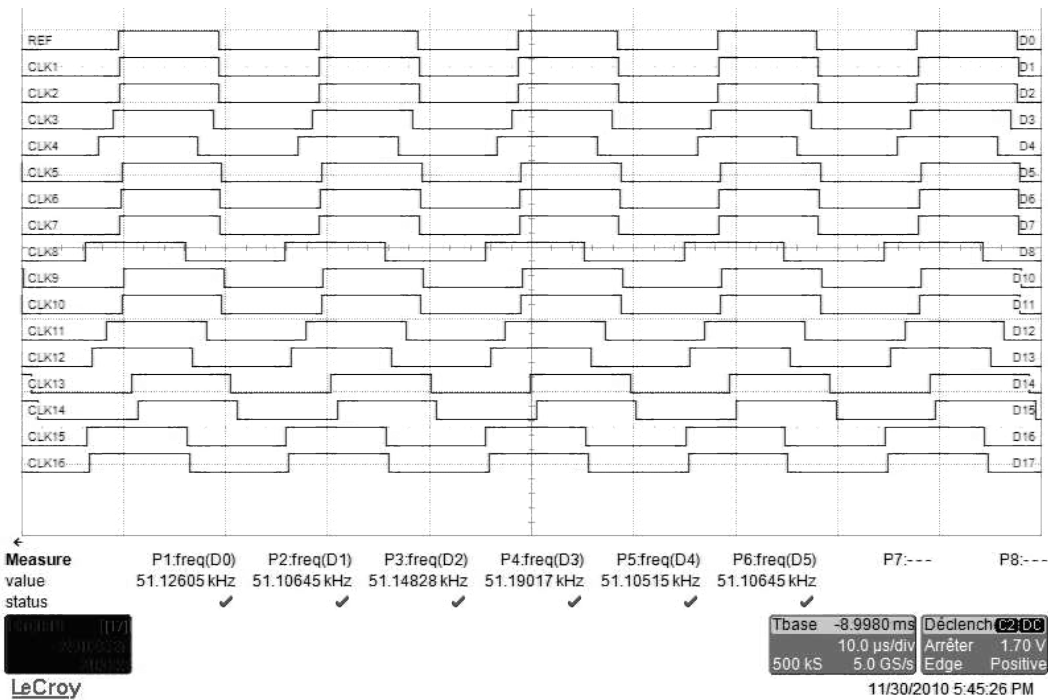


Figure 5.7: **Local clock signals together with reference:** bidirectional configuration, initial frequencies of nodes are different

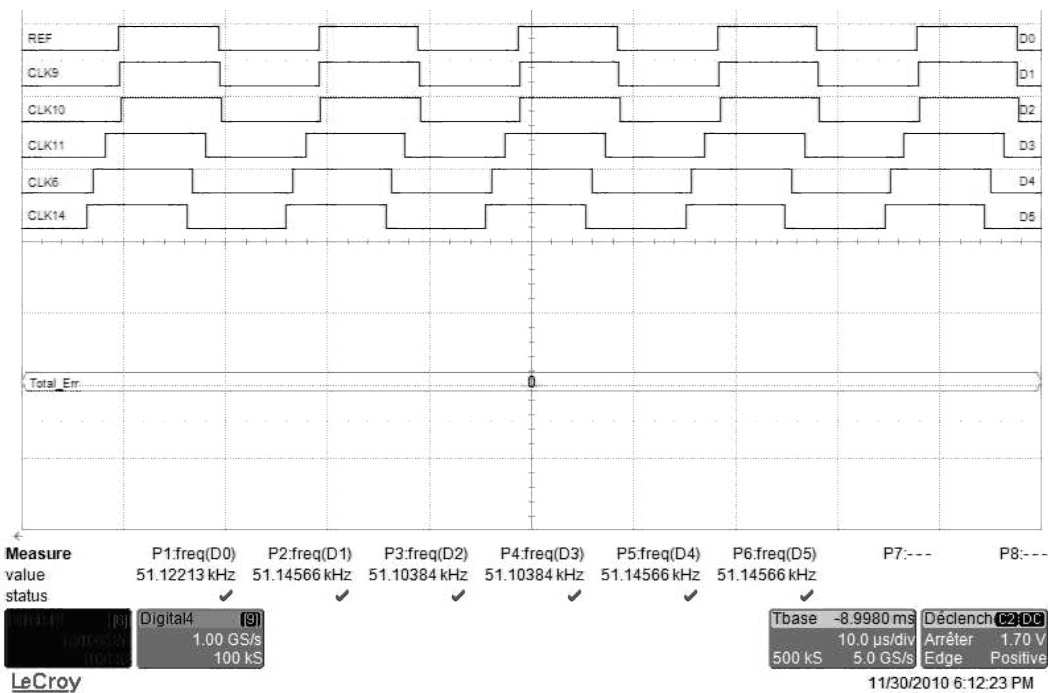


Figure 5.8: **Local clock signals around Node 10 together with reference and integer sum of the errors:** bidirectional configuration, initial frequencies of nodes are different, undesired synchronized state

In the second stage, the network is configured to operate in the bidirectional mode. The bidirectional mode is switched on once the unidirectionally configured network set up in a steady state mode. The mode switching is done by setting to 1 the gains K_{w_1}, K_{w_4} . Fig. 5.11 shows the steady state operation in the second stage. These plots indicate that the static

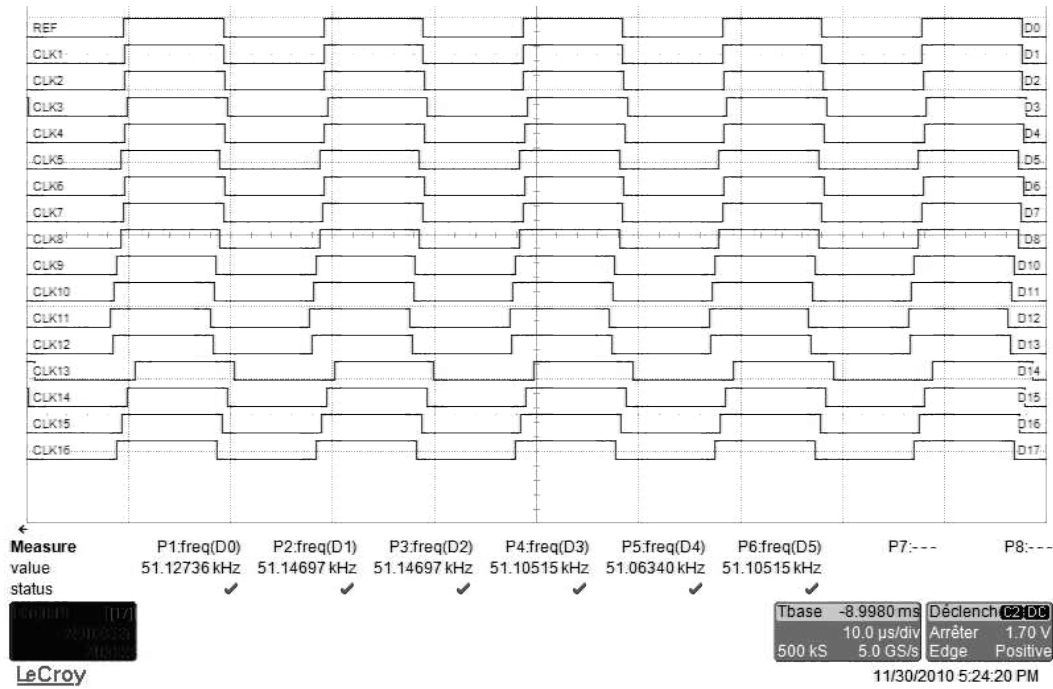


Figure 5.9: Local clock signals together with reference: unidirectional configuration, initial frequencies of nodes are different

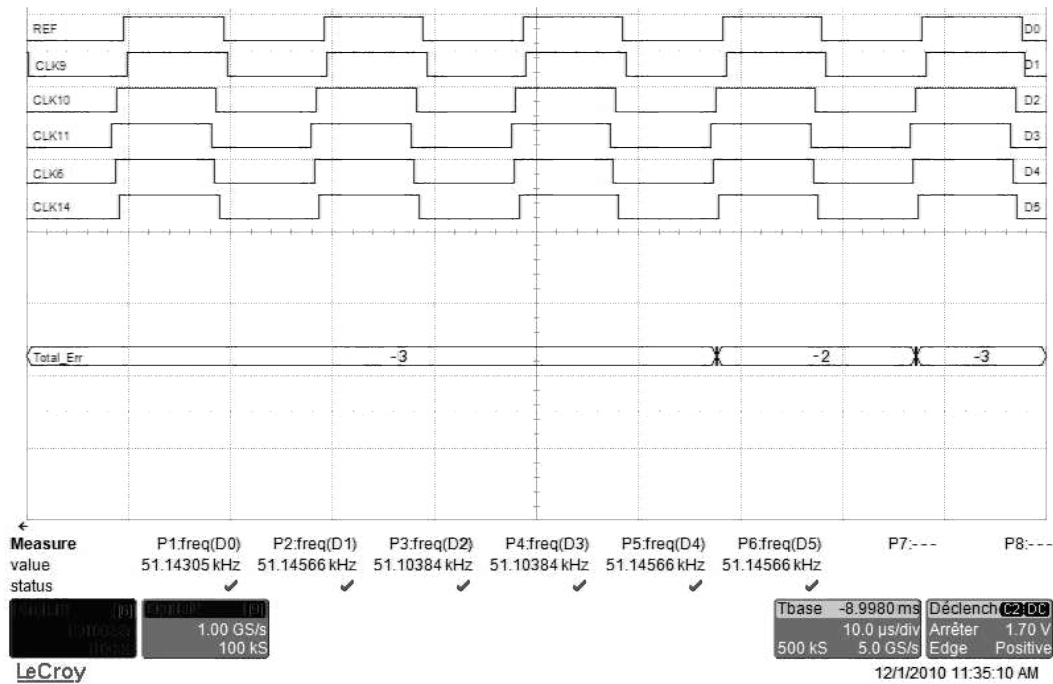


Figure 5.10: Local clock signals around Node 10 together with reference and integer sum of the errors: unidirectional configuration, initial frequencies of nodes are different, static errors exist

residual phase errors appeared in a previous unidirectional mode are compensated and minimized. The plots of Fig. 5.12 demonstrate that they are small. Observation of the PFD outputs demonstrates that the errors between neighboring oscillators do not exceed 2 phase errors quantization steps.

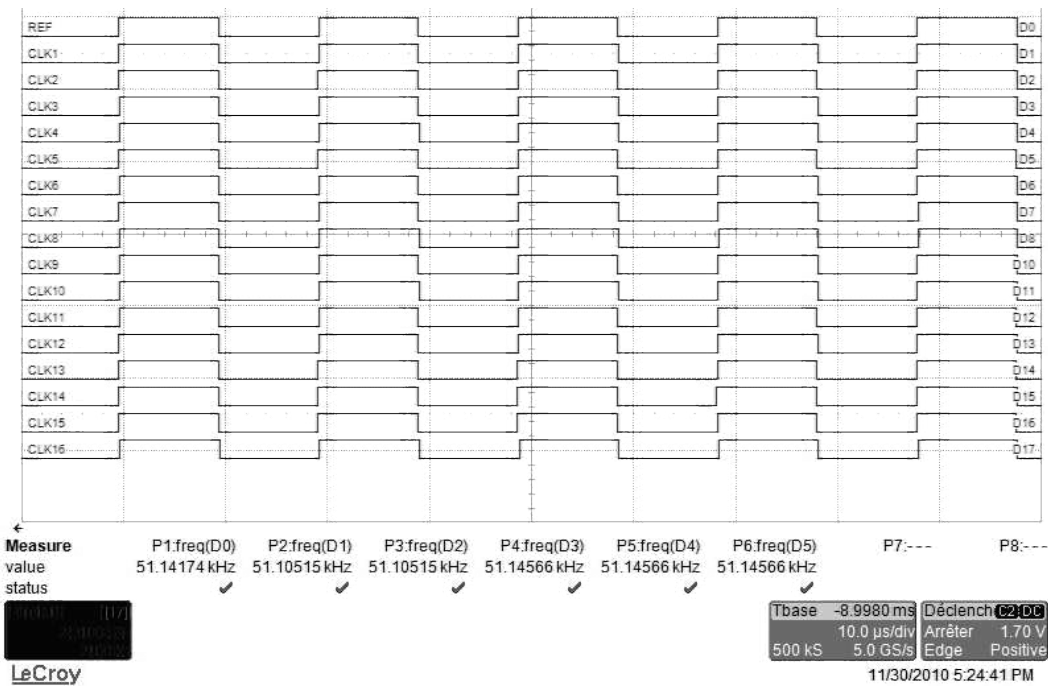


Figure 5.11: **Local clock signals together with reference:** bidirectional configuration, initial frequencies of nodes are different

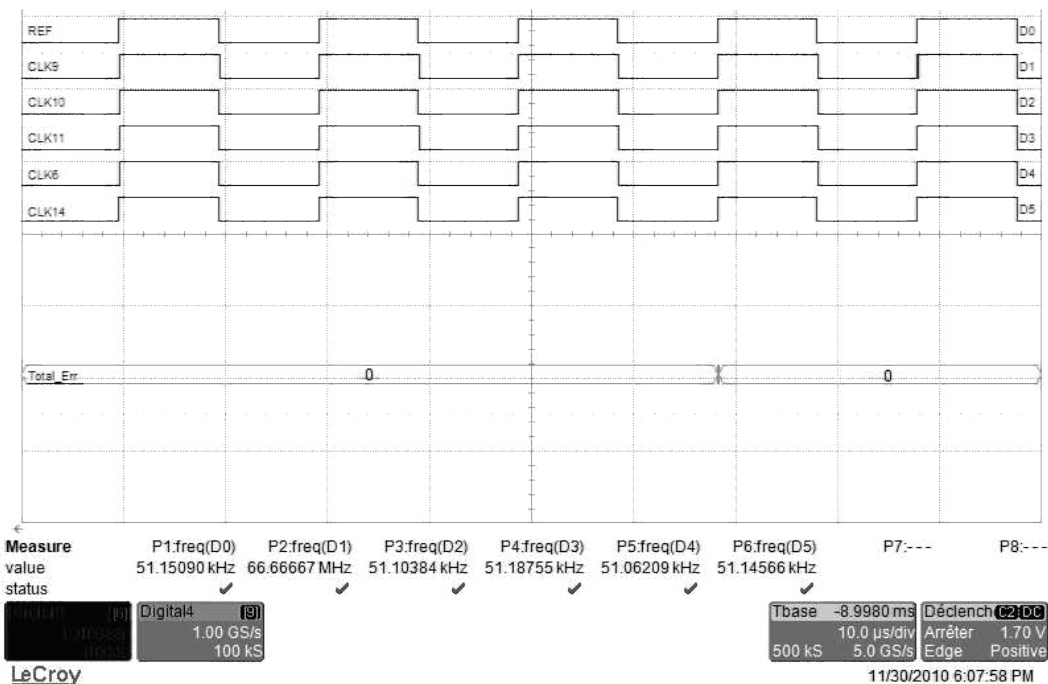


Figure 5.12: **Local clock signals around Node 10 together with reference and integer sum of the errors:** bidirectional configuration, initial frequencies of nodes are different, synchronized state with zero total error

5.2.4 FPGA prototyping: conclusion

The test of the prototyped ADPLL network provided satisfying results. The network behaves as predicted by the theory. The FPGA prototyping validated the design of the digital blocks of the ASIC system, and functionally validated the designed ADPLL network architecture.

Moreover, this step allowed an exploration of the behavior of the network for different parameters, and in different operation modes. The FPGA prototyping provides a strong basis for the further design of the ASIC.

5.3 Silicon implementation of the clock network

This section discusses practical issues of the ASIC design of the ADPLL network prototype in a 65 nm CMOS technology, with use of the PFD, DCO and digital error processing block presented in the previous chapters. This section presents the chip floorplan, the design of the supply network and the DFT issues. Simulation results of the extracted from layout network will be presented.

The chip implements the ADPLL network whose architecture is given in Fig. 5.1. To the blocks of this architecture which we call *core* are added *service* blocks necessary for the chip test and evaluation. The service blocks implement the programming function and the output interface allowing a measurement of the internal chip signals.

5.3.1 Floorplan of the test chip

The core network of the chip is composed from three elementary blocks: the PFD, the digital processing block and the DCO whose designs were presented in the previous chapters. A preliminary floorplan of the implemented core circuit is given in Fig. 5.13. The core area is partitioned into SCAs in which are placed the DCOs and the digital error processing blocks. In the prototype chip the network DCO must be spaced by some distance, in order to create realistic delays and so to approach real conditions of use of the clock generator. In real conditions, the remaining space of the chip would be used by the functional circuit (e.g. processor core), but the designed prototype implements only the clock generator, and the space between the clock generator blocks is left empty. The actual (implemented) distance between the DCO is limited by the maximal chip area defined by budget issues. Going forward, we notice that in this particular case, the chip size is I/O constrained, since the dimensions and number of I/O pads are predominant. The compromise consisted in implementation of 16 rectangular SCAs of size $220 \times 180 \mu\text{m}^2$.

Each PFD is placed on a border between two SCAs, so to have a symmetric position with regard to the couple of neighboring DCOs whose output signal phases are compared.

The PFD and the DCO are surrounded by a double active guard ring which ensures a good noise isolation. The digital processing blocks are surrounded by a simple guard ring.

The DIV output of each local clock is loaded by the digital processing block of the node: the latter generates a noise in the supply network, so imitating realistic operation conditions of a clock generator.

The programming interface is distributed over all PFDs and error processing blocks according to Fig. 4.29 and Fig. 5.16, so it is integrated with these blocks and does not require a specific placement effort. The Subsection 5.3.2 provide more information about the chip programming interface design.

The task of the output interface is to provide a possibility to output internal chip signals to the chip pads. The output interface block is placed in the middle of the chip (denoted as DFT in Fig. 5.13), so to allow a *star* connection to the observed signals and to the chip pad ring. More information about the output interface design is given in Subsection 5.3.2.

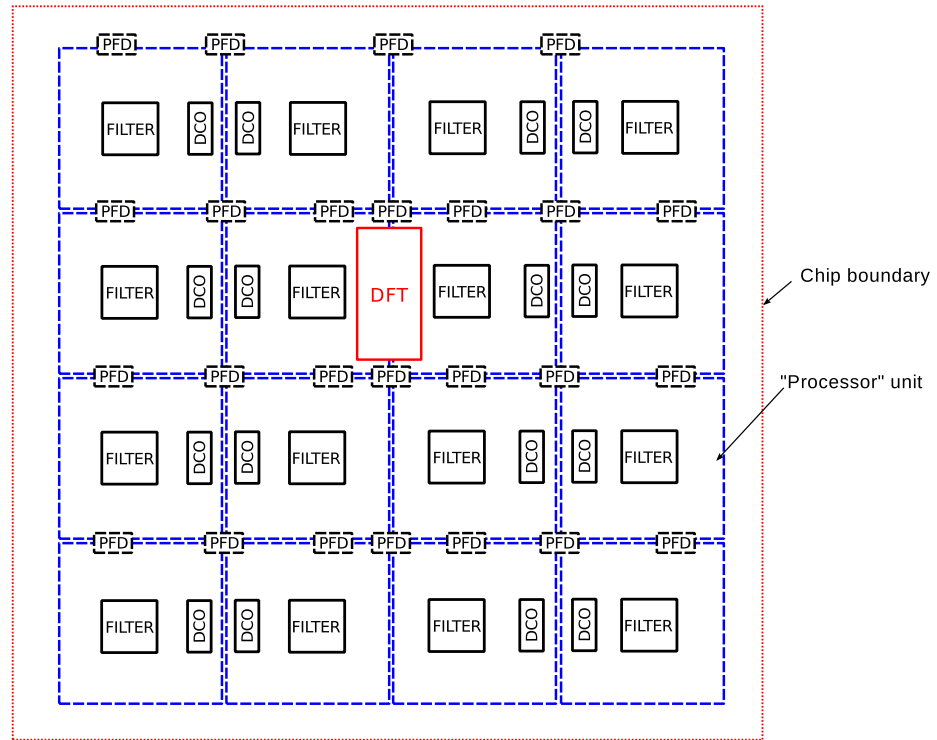


Figure 5.13: **Preliminary floorplan of the test chip**

The chip implies two supplies: one for the PFD, digital error processing blocks, DFT block and the rest digital circuitry of the system, and an analog supply for the DCO blocks. The separation of the supply networks allows a minimization of the noise impact of the DCO operation on the digital circuits. The supply organization is presented on Fig. 5.14.

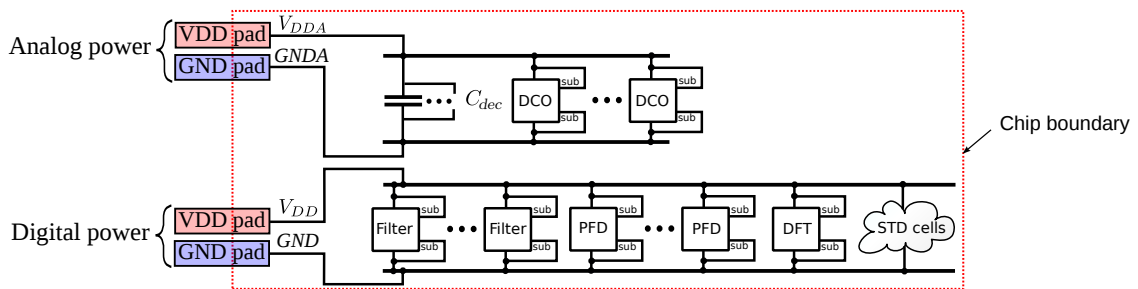


Figure 5.14: **Power supply scheme of the test chip**

Over all blocks of the ADPLL network, the DCO is the most sensitive toward supply voltage variation. For that reason, the IR drop in the supply distribution network was studied for this block more carefully. The detailed schematic including parasitic elements was extracted from the layout and then simulated with a help of Eldo simulator. To estimate the IR drop in the worst case, the extracted DCO supply network was loaded with 16 DCO operating on maximal frequency and at highest temperature of 125°C. Fig. 5.15 presents the results of this simulation. The maximal drop of the supply voltage at the DCO supply pins is within desired 5 % margin.

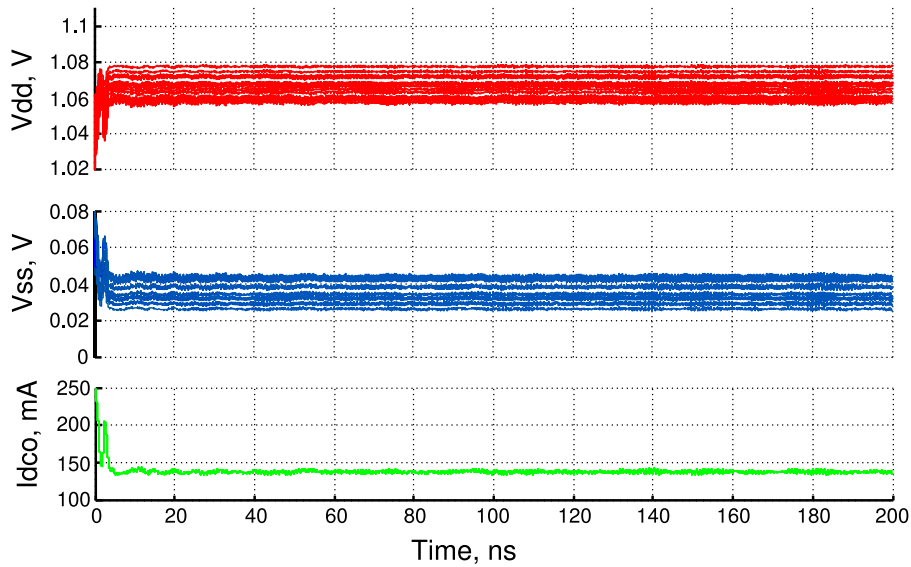


Figure 5.15: **Simulation of the extracted from layout supply distribution network:** loaded with oscillators operating at maximal frequency and temperature 125°C

5.3.2 Design for test

As stated in [74], pp.46-47, DFT is an important stage of a chip design which should be a part of design process of any integrated circuit. The term DFT designates all issues, techniques and strategies making possible test, validation and verification of the chip after the fabrication. The choice of the DFT strategy is particularly valuable in our case, since the designed chip is the first prototype of a ADPLL network designed in the project. Therefore, DFT block(s) must be indivisible part of the chip prototype. The DFT related functions are the following:

- The programming of the chip, which directly relates to controllability concept ([74], p.114). This functionality allows a reconfiguration of the topology of the network, a choice of the loop filter coefficients and the PFD operation modes. It multiplies the number of configurations in which the chip can be tested, so increasing the impact of the chip test on the validation of the concept of ADPLL network based clock generation.
- The management of the chip signals to be output off the chip, which relates to observability concept ([74], p.114). Since the number of pads is limited, a strategy of the chip test should be developed, so to maximize the knowledge about the chip operation during the test.

The next two subsections present these two issues.

Chip programming

The programming interface for each block was presented in Section 4.5. As explained in that section, the interface is designed so to be extendable: if their programming interfaces are cascaded, any number of blocks can be programmed with the single programming cycle.

The required number of the input pads (3 pads) does not depend on the number of cascaded blocks.

The topology of the programming interface cascading on the designed chip is given in Fig. 5.16. The corresponding programming sequence for the developed network consists of S bits

$$S = N \cdot (S_i + S_p + S_{path}) + M \cdot S_{pfd} \quad (5.9)$$

where $N=16$ is number of the network nodes, $S_i=12$ is width of the integral path coefficient, $S_p=5$ of the proportional path coefficient, $S_{path}=2$ is the width is the input gain control coefficients, $M=25$ is number of the PFD, $S_{pfd}=1$ is width of the PFD operation settings word. For the designed network, the total length of a programming sequence is 425 bits. They can be transferred with a high rate (tens of MHz) and in this way a total time for the network programming procedure will be negligible ($\leq 43 \mu s$ @ 10MHz).

The connection topology of the programmable blocks of the network is depicted in a Fig. 5.16.

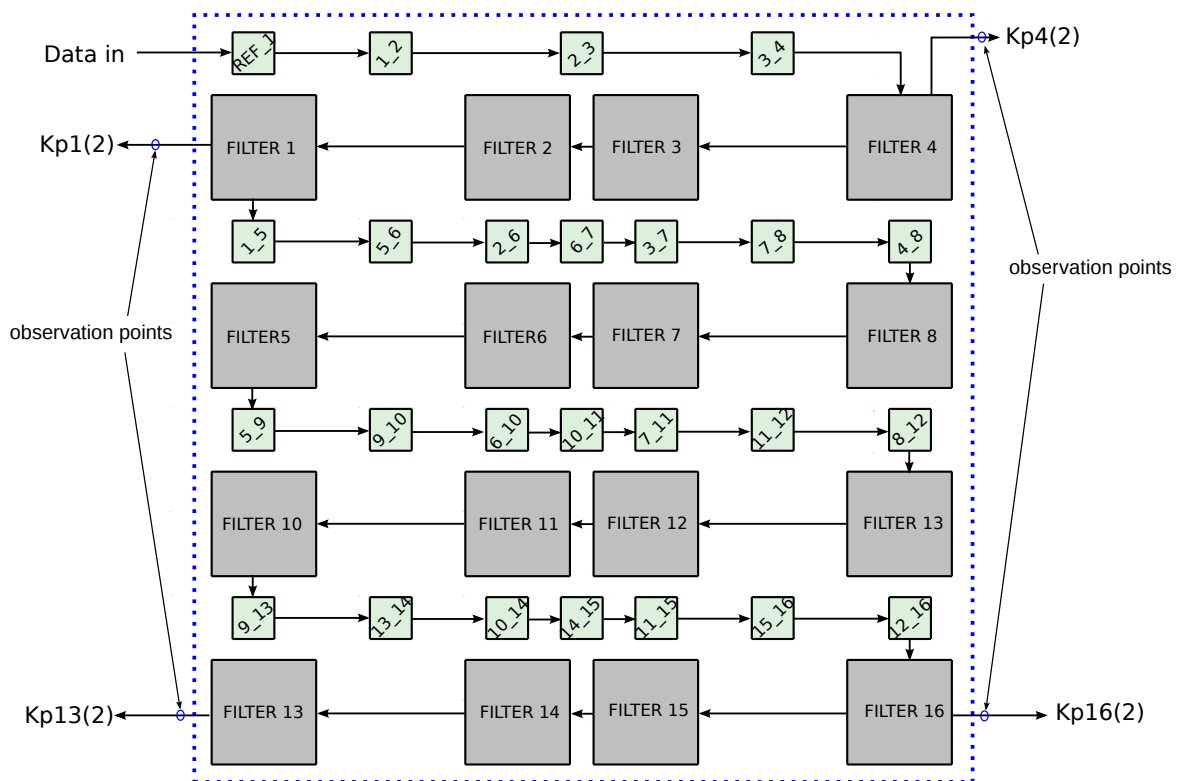


Figure 5.16: The connection sequence of the programmable blocks of the network

Output signal management

An exhaustive test of the chip requires an observation of a big number of signals (Fig. 5.17). These signals are:

- DCO outputs (16 one bit outputs);
- PFD outputs (25×5 bits outputs);

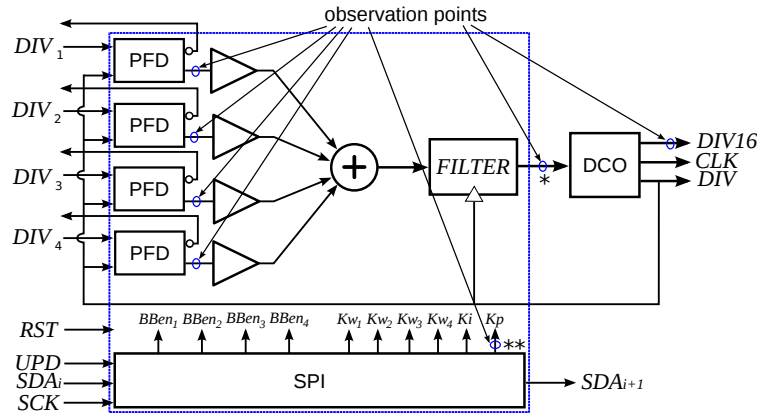


Figure 5.17: **Block diagram of the node in an ASIC implementation:** * only for node coupled with reference, ** only for corner nodes

- digital processing block outputs (16×10 bits outputs).

The total number of these signals exceeds 300. The cost of direct outputting of these signals to the pads is prohibitive, given the chip area limitation. Moreover, a routing of the signals throughout the chip area would increase the parasitic component and may degrade the performance of the chip. In order to reduce the number of the pads, we have decided to multiplex a part of the mentioned signals. The following four restricting choices were made:

1. The outputs of only 4 PFD can be observed simultaneously. This number 4 is chosen since it allows an observation of all phase errors associated with one network node. We note that the PFD outputs provide a very important information about the phase error measured in immediate proximity of the DCO, and the measurement precision is better than provided by the off-chip measurement. To implement this feature, all the PFD outputs are routed toward the chip pads through a multichannel multiplexer (Fig. 5.18). It is used for the selection of four outputs of the PFD which can be observed at the same time. Selection is achieved by submitting the appropriate bit sequence via a serial interface (similar to that used for the network programming but implemented separately) with signals SCK_{sel} and SDA_{sel} .

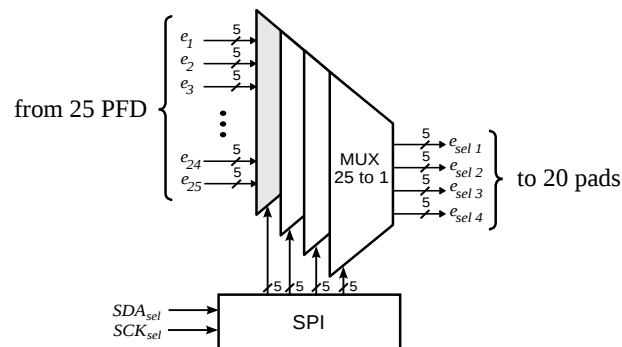


Figure 5.18: **Multichannel multiplexer with serial interface for the PFD output commutation**

2. The 10-bits output of only one of the 16 digital processing blocks is connected to the external pads. The network node whose signal is selected to be observed is the node SCA00 directly coupled with the external reference. The observation of this signal allows an estimation of the ADPLL network state (frequency acquisition, mode-lock, converged). The information about the remaining nodes of the network can be retrieved from the outputs of the PFD and clocks of the oscillators.
3. Four pads are reserved for the programming interface testing. These pads are connected to four outputs of the programming interface parallel register selected in different parts of the network, in particular, the third bit of the coefficient K_p of four corner nodes (Fig. 5.16, signals $Kp1(2)$, $Kp4(2)$, $Kp13(2)$, $Kp16(2)$). Given the known programming sequence, the states of these four pads can be compared with the expected value, hence indicating a successful reception of the sequence.
4. The 16 DCO $DIV16$ outputs are connected directly to the 16 external pads. The signal $DIV16$ is a 16 times divided clock generated by DCO, its frequency range is from 63 MHz to 155 MHz. It is chosen for the off-chip observation of the output clock signals due to the bandwidth limitations of the output IO pads from the standard library (which is limited to ≈ 250 MHz).

Complete detailed block diagram of the core network and DFT blocks is given Fig. F.1 of the Appendix F.

5.3.3 Test chip layout

The assembling of the clock network block has been done in a standard digital design flow with a help of EDA tools. The actual layout of the chip is given in Fig. 5.19. The size of the chip is $1390 \times 1470 \mu\text{m}^2$ where clocking network core occupies $900 \times 800 \mu\text{m}^2$, I/O compensation cell $208 \times 196 \mu\text{m}^2$ and DFT circuitry around $80 \times 150 \mu\text{m}^2$. The empty space within clock core is filled with substrate connection fillers and buffers, and the empty space outside of the clock core is filled with decoupling capacitors.

The microphotograph of the fabricated and packaged test circuit is shown in a Fig. 5.20.

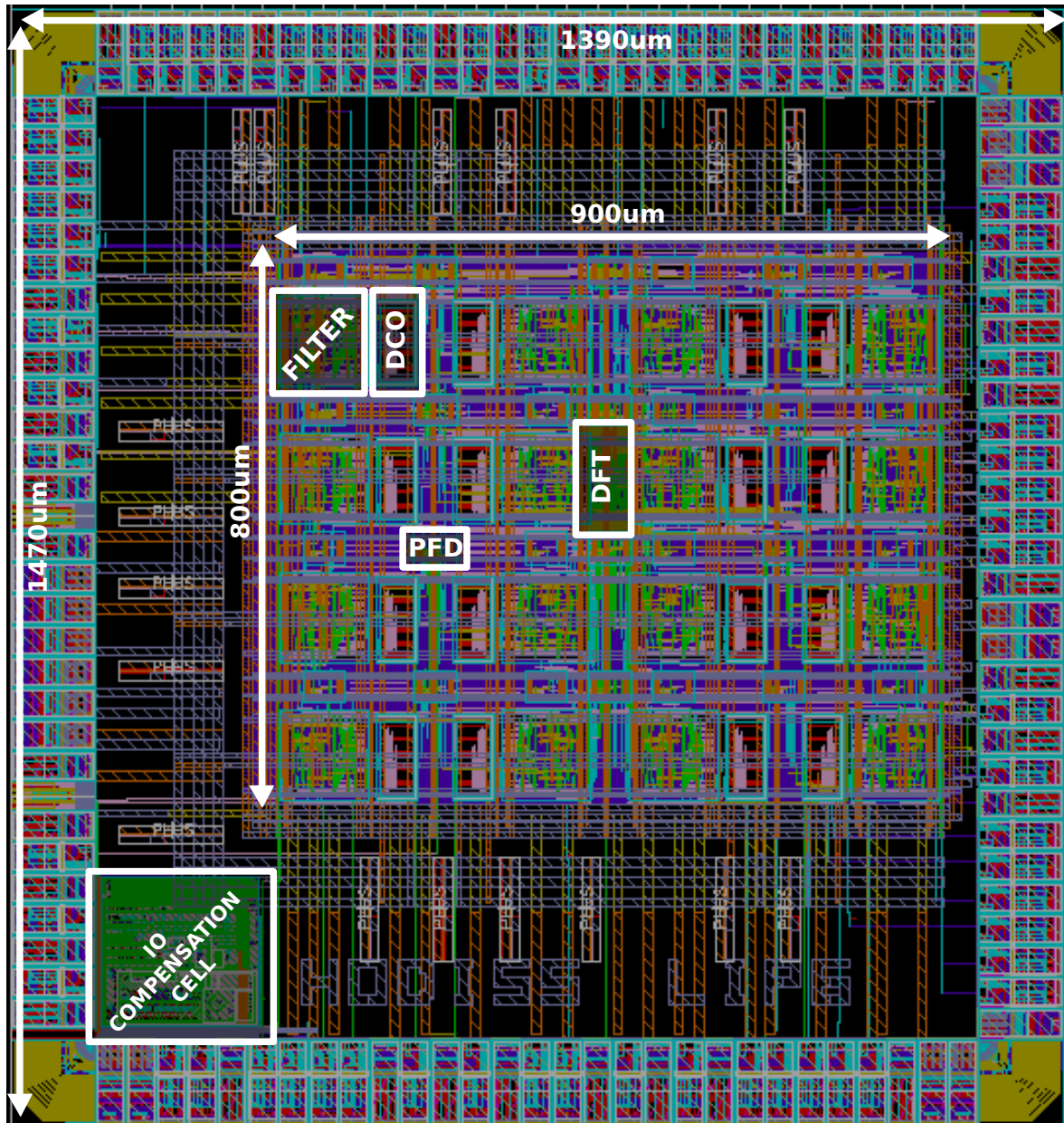


Figure 5.19: Layout of the test chip of the clock network

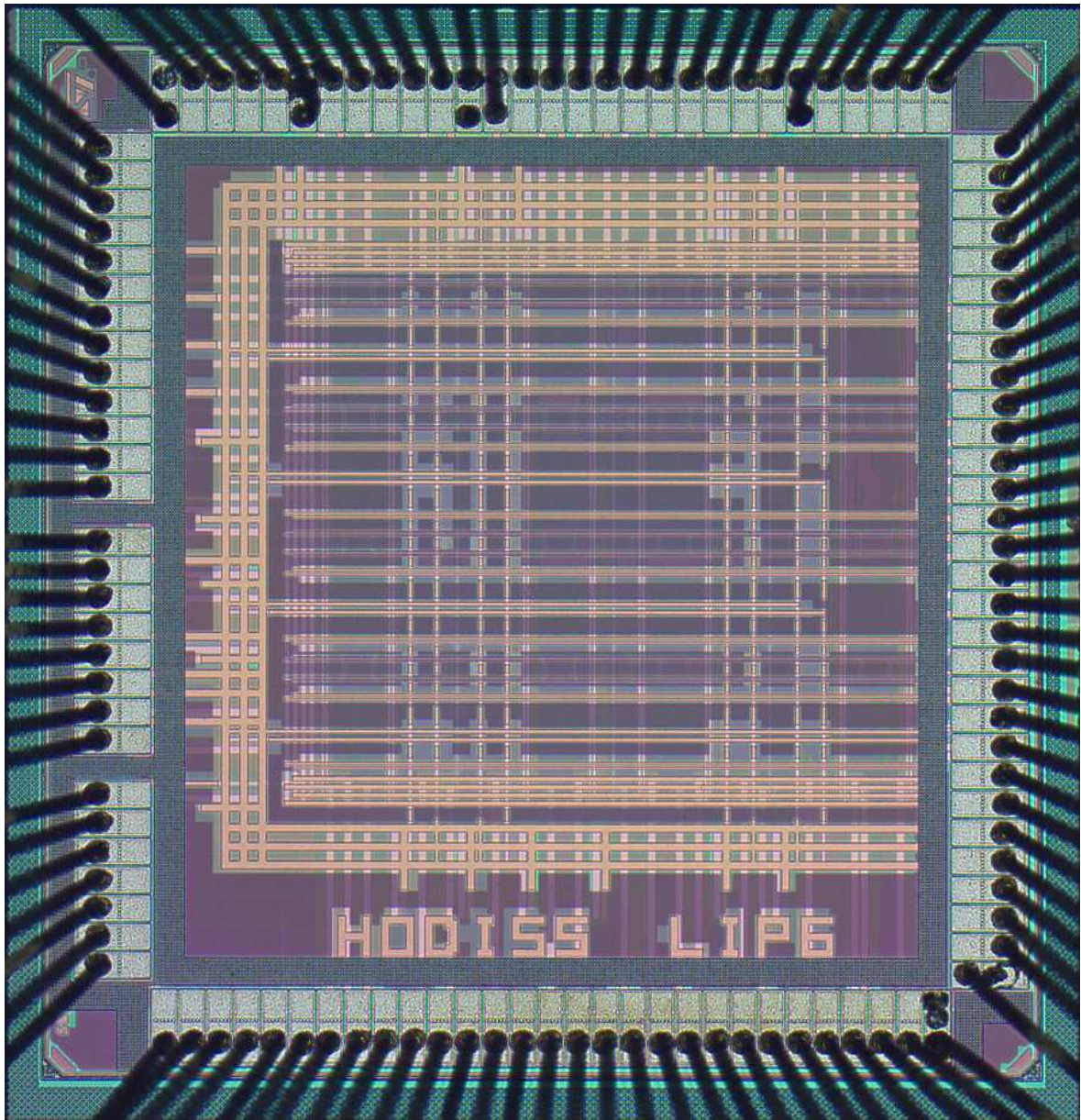


Figure 5.20: Microphotograph of the fabricated clocking network test circuit

5.4 Measurement results

This section presents the results of the measurements of the fabricated test chip. The tests were performed using multifunctional clock generator for reference clock, digital sampling oscilloscope with 4 inputs for acquisition of clock signals and their preliminary analysis, logical analyzer for acquisition of digital 5 bit outputs from PFDs, multifunctional digital voltmeter for voltage, current and power consumption measurements and adjustable power supply. More details about the measurement set, used tools, methodology and data post-processing procedures can be found in Appendix C and Appendix D.

We performed several experiments which characterize the fabricated prototype. The basic conditions of these experiments are summarized in Tab. 5.2.

Table 5.2: Clock network test chip measurement conditions

<i>Parameter</i>	<i>Value</i>
Supply voltage	1.2 V $\pm 10\%$
Environment temperature	24°C
Input reference frequency	200 MHz
K_p ¹	1.0 $\pm 10\%$
K_i ¹	0.0028 $\pm 10\%$
Network configuration	uni- and bidirectional
PFD mode	normal

¹Normalized to single input node

5.4.1 Initial frequencies of DCOs

The goal of this experiment is to determine some initial conditions of the network start-up. From this experiment we determined the frequencies of the free running DCOs with the same control code after the power up. The variation of these frequencies allowed us to estimate the impact of the PVT variations on the network.

In order to have the same control code and set it to the central value of 512 (i.e. middle of the tuning range), the digital circuitry of all nodes have been forced by the global reset signal to fix their states. The processed results of this experiment are shown in Fig. 5.21. As one can see, the dispersion of the initial frequencies of local clocks *CLK* can be characterized by repetitive pattern (drop of the frequency in Nodes 1-2-3-4 then in nodes 5-6-7-8 and so on) and global positive trend of the frequency between these patterns. These effects are explained mainly by the IR drop in power distribution network and sensitivity of the DCO to the variation of supply voltage. The trend observed in Fig. 5.21 is due the 2D mesh topology of the power distribution network. The maximum difference between frequencies is 42.7 MHz.

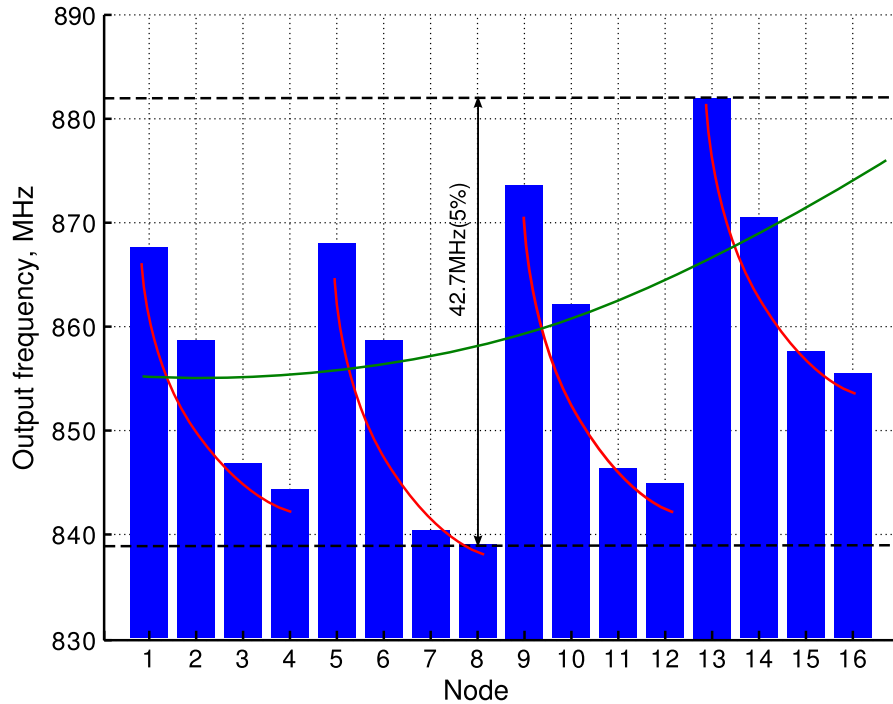


Figure 5.21: Measured initial frequencies of the DCOs

5.4.2 Supply voltage sensitivity

The measurement of supply voltage sensitivity of oscillators was performed for all nodes. The DCO control codes have been set to the central value and the variation of the supply was $\pm 10\%$ of the 1.2 V. The results of measurement is depicted in Fig. 5.22. This plot shows the averaged curve for 16 oscillators, since the sensitivities of oscillators were close. The average sensitivity is ≈ 910 MHz/V, which is in a good match with the measurement performed on chip with single oscillator (c.f. Section 3.8). Both of them highlight a high sensitivity to the supply variations. However, as we stated in Section 3.8 for any supply voltage within $\pm 5\%$ range, the specified nominal frequency 1 GHz is reachable with an input code far from the limits of the range (0 and 1023).

5.4.3 Power consumption

The power consumption has been measured for central oscillation frequency with supply voltage varied in $\pm 10\%$ range. Fig. 5.23 depicts the results of this measurement; separately for oscillators and digital circuitry. The current consumption of oscillators is in a good agreement with measurement of the consumption of the oscillators in DCO test chips. The power consumption is ≈ 98.47 mW/GHz, which gives ≈ 6.15 mW/GHz per oscillator (i.e. divided by 16); for the DCO chip prototype this value was measured as ≈ 6.2 mW/GHz

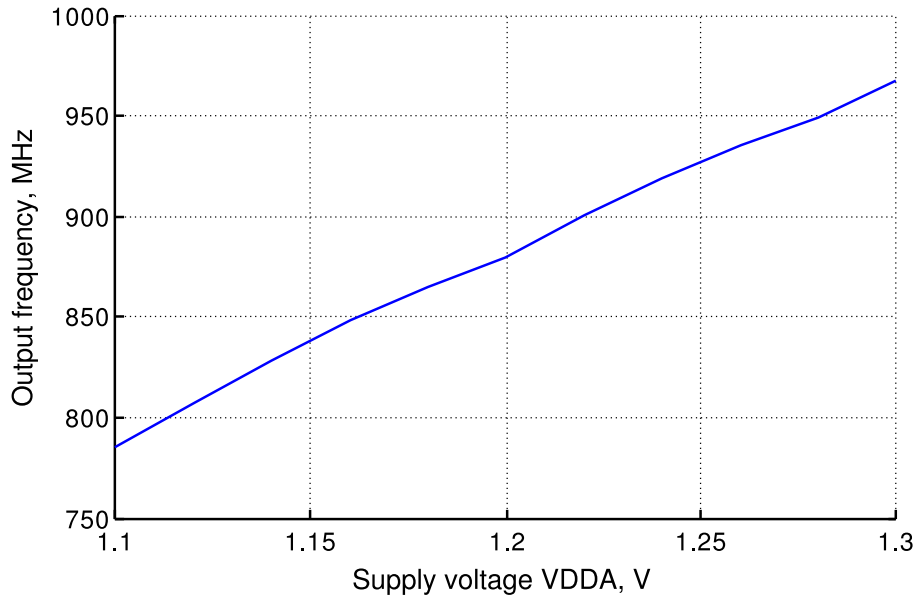


Figure 5.22: **Measured supply voltage sensitivity of the DCOs**

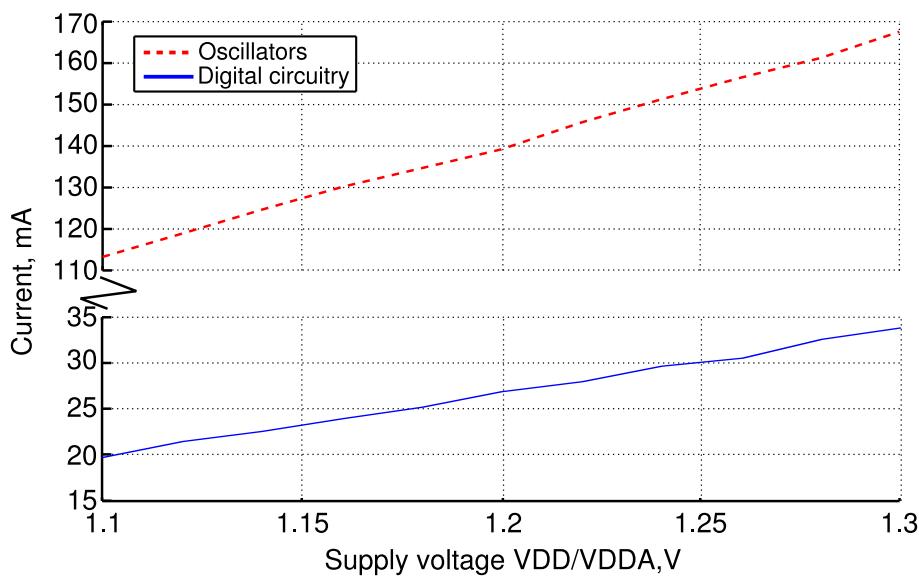


Figure 5.23: **Measured current consumption of the network**

5.4.4 Bidirectional configuration

The main goal of this experiment is to measure the performance of the network in bidirectional configuration (Fig. 2.18): phase errors between nodes in steady-state. A secondary goal is an observation of undesired synchronization modes (mode-locks) in the network.

Before all timing/frequency measurements, we have characterized the source of the reference clock, which ADPLL network will track. Its histogram is shown in Fig. 5.24. For all following experiments we operated with 200 MHz reference clock.

For this experiment the network has been separated into nine zones (Fig. 5.25), since we have a limited amount of channels in oscilloscope and digital analyzer (4 for oscilloscope and 4 5 bit channels for logical analyzer). Consequently, the results are divided into nine

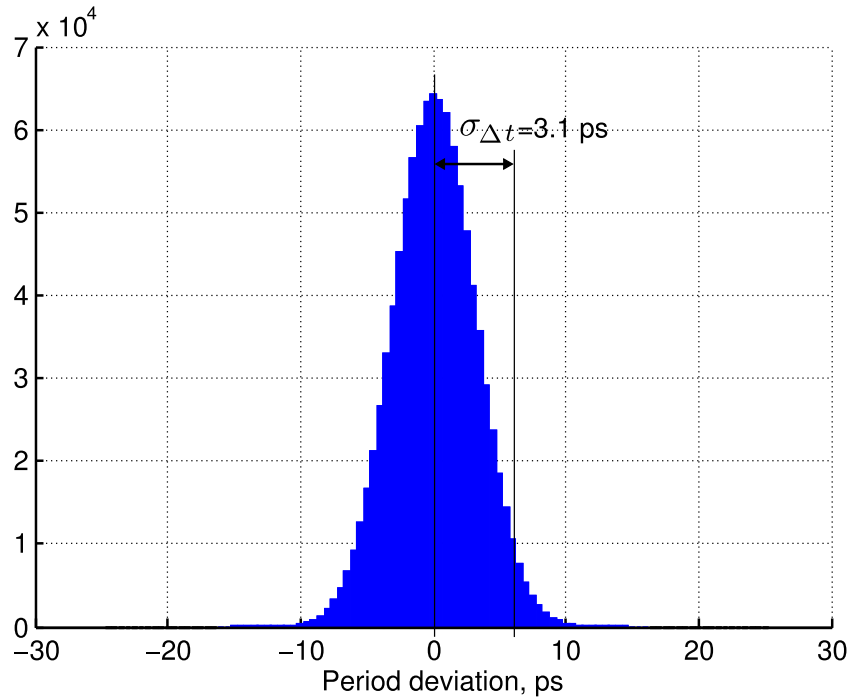


Figure 5.24: **Histogram of the reference clock:** captured for 200 MHz

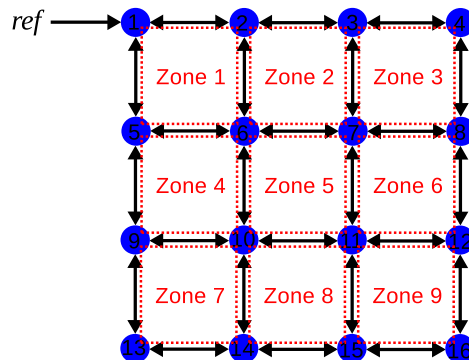


Figure 5.25: **Zones of the network for the measurement**

groups. We show here only the plot for the Zone 9, which is the most remote zone from the reference clock input. The plots from remaining 8 zones are shown in Appendix C.

As one can see in Fig. 5.26, the phases of all 16 nodes converged. The maximum error in Zone 9 does not exceed ± 2 steps of PFD quantization step (Fig. 5.27). This result signifies that timing errors between local clocks are within ± 60 ps range.

An important note is that for 500 network start-ups we haven't observed the mode-locking states.

5.4.5 Unidirectional configuration

The test of the network in unidirectional configuration allows a comparison between the network performances with bidirectional configuration. Theoretically, the performances of unidirectional network are poorer than the performances of bidirectional network, although

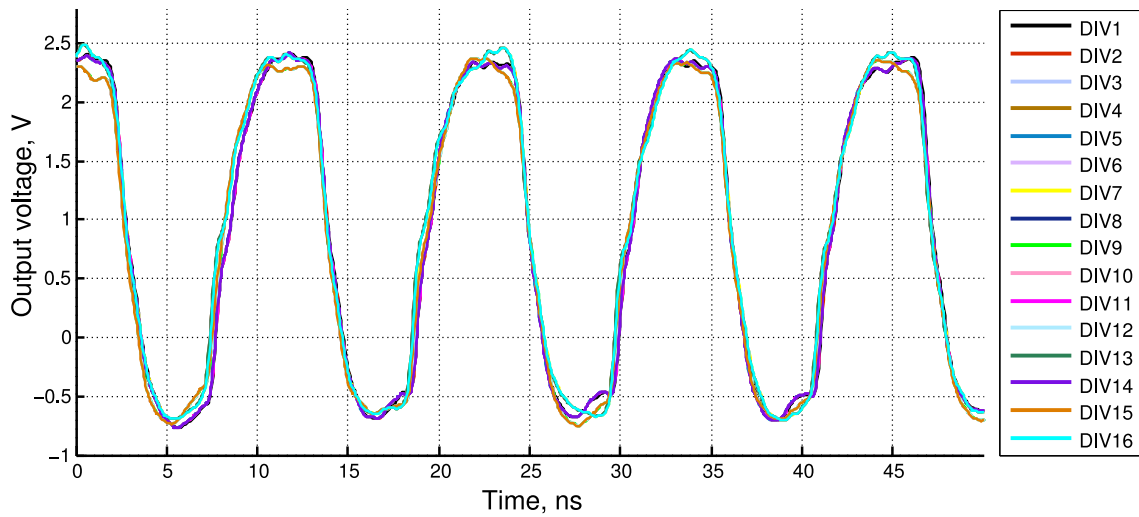


Figure 5.26: **Synchronous clocks in the bidirectional configuration**

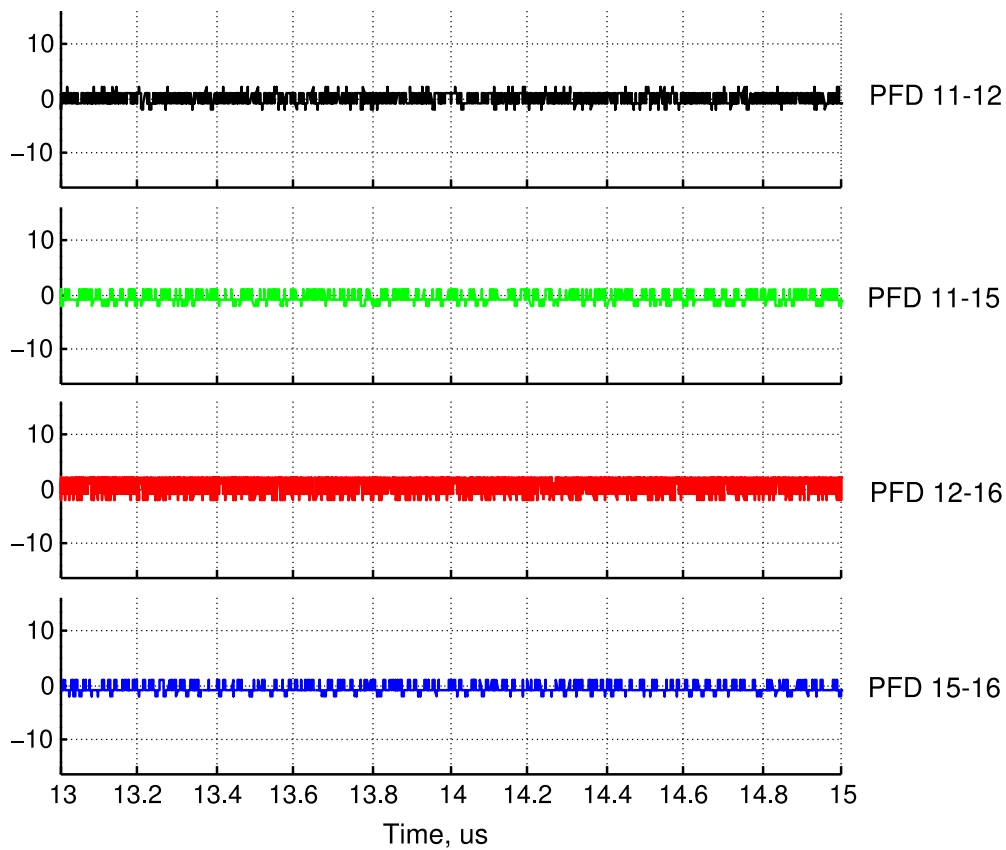


Figure 5.27: **Outputs of the PFDs from the Zone 9 in bidirectional configuration: Nodes 11, 12, 15 and 16**

the implementation of an unidirectional network is much easier (e.g., there is no stability problem). To make this comparison in practice, the network was programmed in an unidirectional configuration according to Fig. 2.12.

The conditions of this experiment is the same as for bidirectional configuration. The measurements have been performed for all 9 zones of the network. In order to compare them with bidirectional configuration, Fig. 5.28 shows the errors from the Zone 9. The maximum

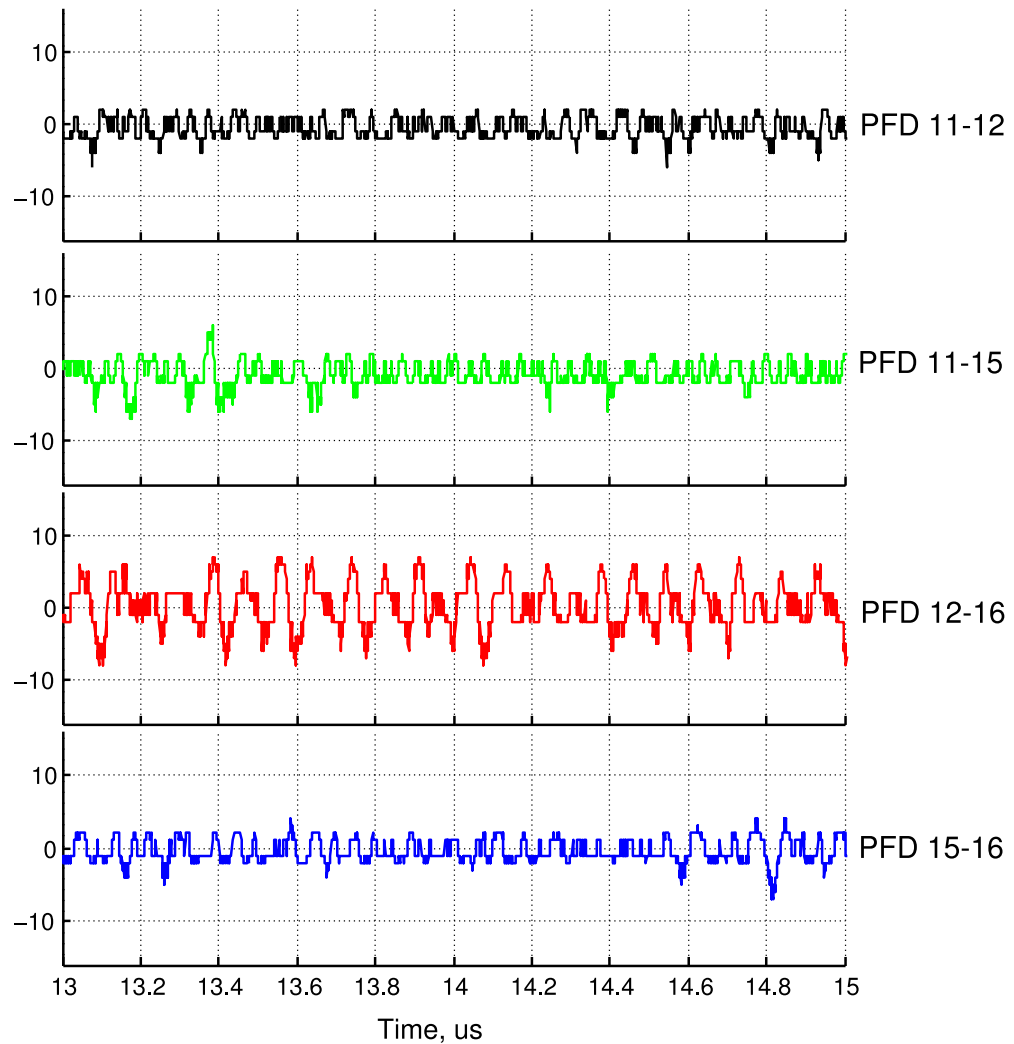


Figure 5.28: **Outputs of the PFDs from the Zone 9 in unidirectional configuration: Nodes 11, 12, 15 and 16**

error is within ± 7 steps of PFD quantization step, which gives ≈ 210 ps timing error or 0.08π for the 200 MHz clock signal. This confirms the theoretical prediction about larger phase errors in unidirectional network.

5.4.6 Corner to corner timing errors

The goal of this experiment is to know the error between nodes at topological distance more than 1: for example between corner nodes. For this experiment we provide only the captured waveforms of the clocks, since we don't have on-chip instruments of error measurement between these nodes. Fig. 5.29 demonstrates the waveforms of the clock signals from Nodes 1,4,13 and 16. The maximum timing error is 290 ps. This error is much higher than the error obtained for neighboring nodes (≈ 60 ps). Indeed, the geometry of the network is such that a "good quality" synchronization is only guaranteed for the neighboring nodes; this result is not surprising.

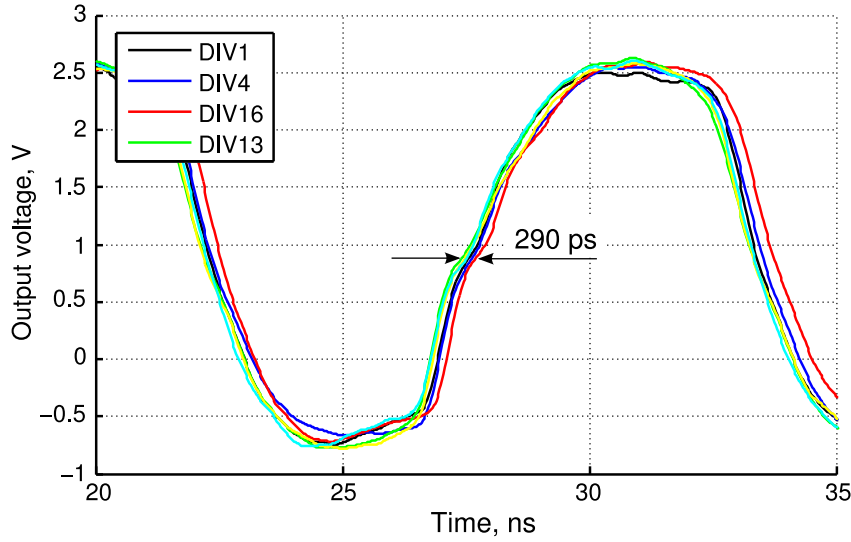


Figure 5.29: **Timing error between corner nodes**

5.4.7 Study of mode-locking phenomenon

This experiment is motivated by absence of observed mode-lock states in fully-connected bidirectional network. This result is in a contradiction with modeling and the theory: the possibility of mode-locking is one of particular properties of a PLL networks which is always mentioned in theoretical studies. The non-observation of undesirable synchronized states may be explained by small attraction basin of these modes, and hence by small probability to observe them. This fact can also be explained by the impact of the environment of a real CMOS circuitry (e.g. supply and substrate noise), which haven't been considered during the theoretical study and simulations.

Therefore, a research of mode-lock states was repeated with reduced network configuration (Fig. 5.30), where theoretically mode-lock must occur with high probability. Four Nodes 1, 2, 5 and 6 were isolated from the remaining network nodes and feedback signals to Node 1 from neighboring Nodes 2 and 5 were cut. In such a configuration, for 500 attempts of perturbation by global reset we have observed a mode-lock 4 times. One of those states has been captured and characterized.

Fig. 5.31 demonstrates the captured clocks in the mode-lock state: they are misaligned and have constant non-zero timing error. Fig. 5.32 shows the output signals of the PFDs in this state. As one can see, the sums of the errors for Nodes 2, 5 and 6 are compensating each other. This result is in good agreement with the theoretical case illustrated in Fig. 2.10. It can be easily verified by calculating the signals *Total error* for these nodes

$$\begin{aligned}
 Total\ error_6 &= e_{ri}[2,6] + e_{ri}[5,6] = +15 - 15 = 0 \\
 Total\ error_2 &= -e_{ri}[1,2] + e_{ri}[2,6] = -15 + 15 = 0 \\
 Total\ error_5 &= -e_{ri}[1,5] + e_{ri}[5,6] = +15 - 15 = 0
 \end{aligned} \tag{5.10}$$

The signs of errors PFD 1-2 and PFD 1-5 are taken negative because we capture the errors only from the noninverted outputs of the PFDs, therefore a correction of the sign is required to the data shown in Fig. 5.32.

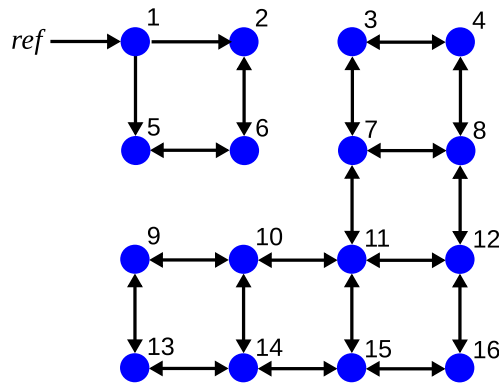


Figure 5.30: **Configuration of the network for mode-locking study:** Nodes 1, 2, 5 and 6 isolated from the network

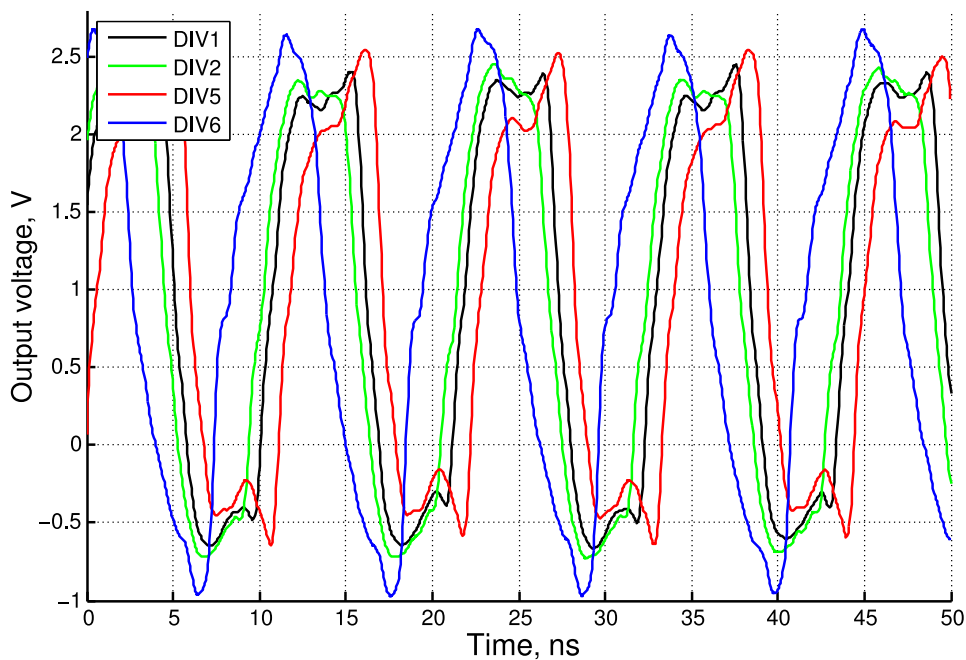


Figure 5.31: **Misaligned clocks in the mode-locking state**

5.4.8 Coefficient variation

The goal is to observe the impact of the variation of the coefficients K_p and K_i . They have been tuned within $\pm 10\%$ range from the nominal specified value. The result of their impact has been observed at the Node 11 during the transient process.

Fig. 5.33 demonstrates the nine curves given instantaneous frequency evolution during the transient process in Node 11. It is expected that a variation of the integral coefficient changes the convergence time. However, the main result of this observation is that we can't see significant oscillations at the end of the convergence, that means that the network remains stable even with this variation of the K_i coefficient. The impact on stability concerning the variation of K_p , however, is negligible.

The Tab. 5.3 presents the performance and parameters summary of the measurements of the test chip.

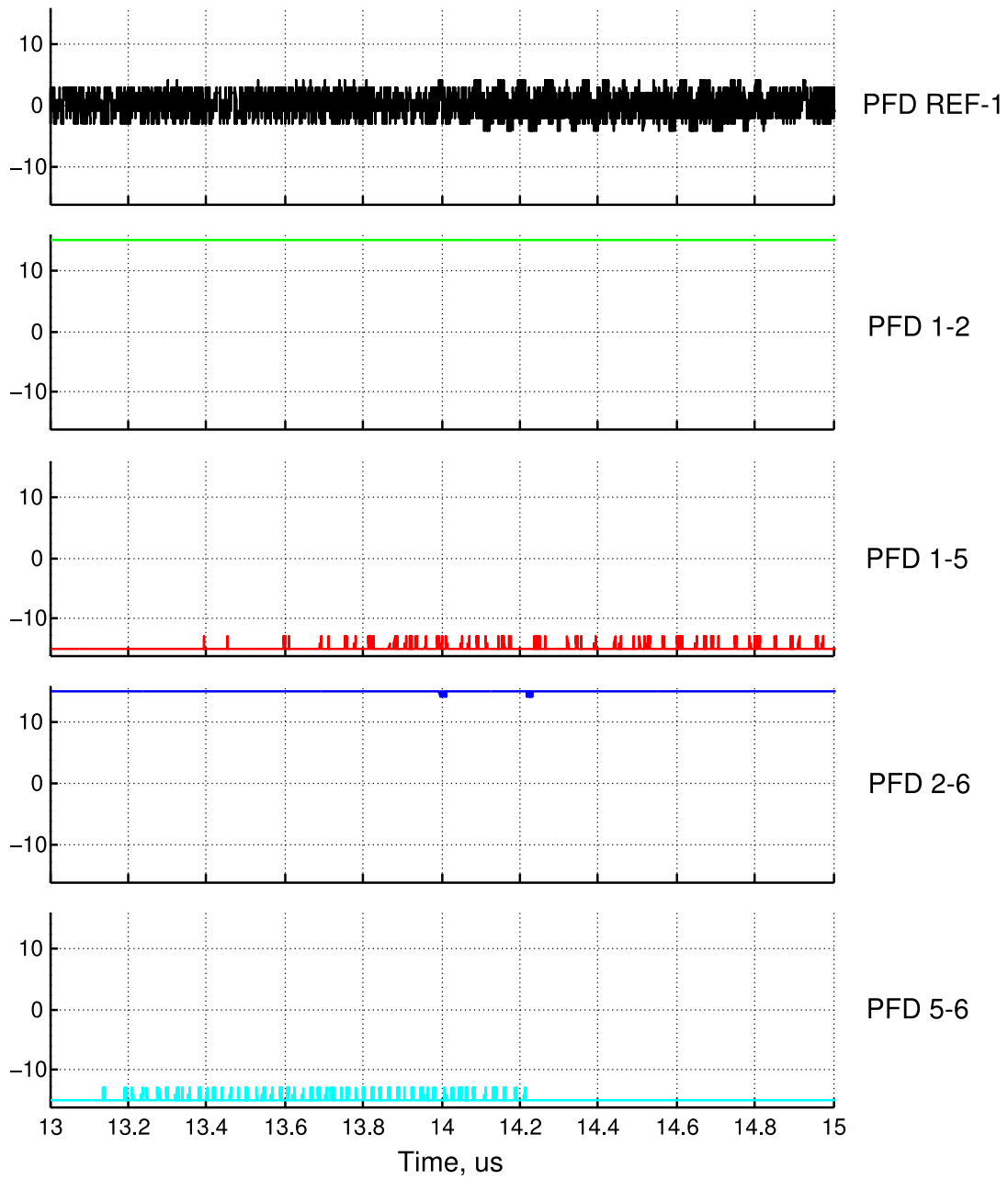


Figure 5.32: Outputs of the PFDs in the mode-locking state

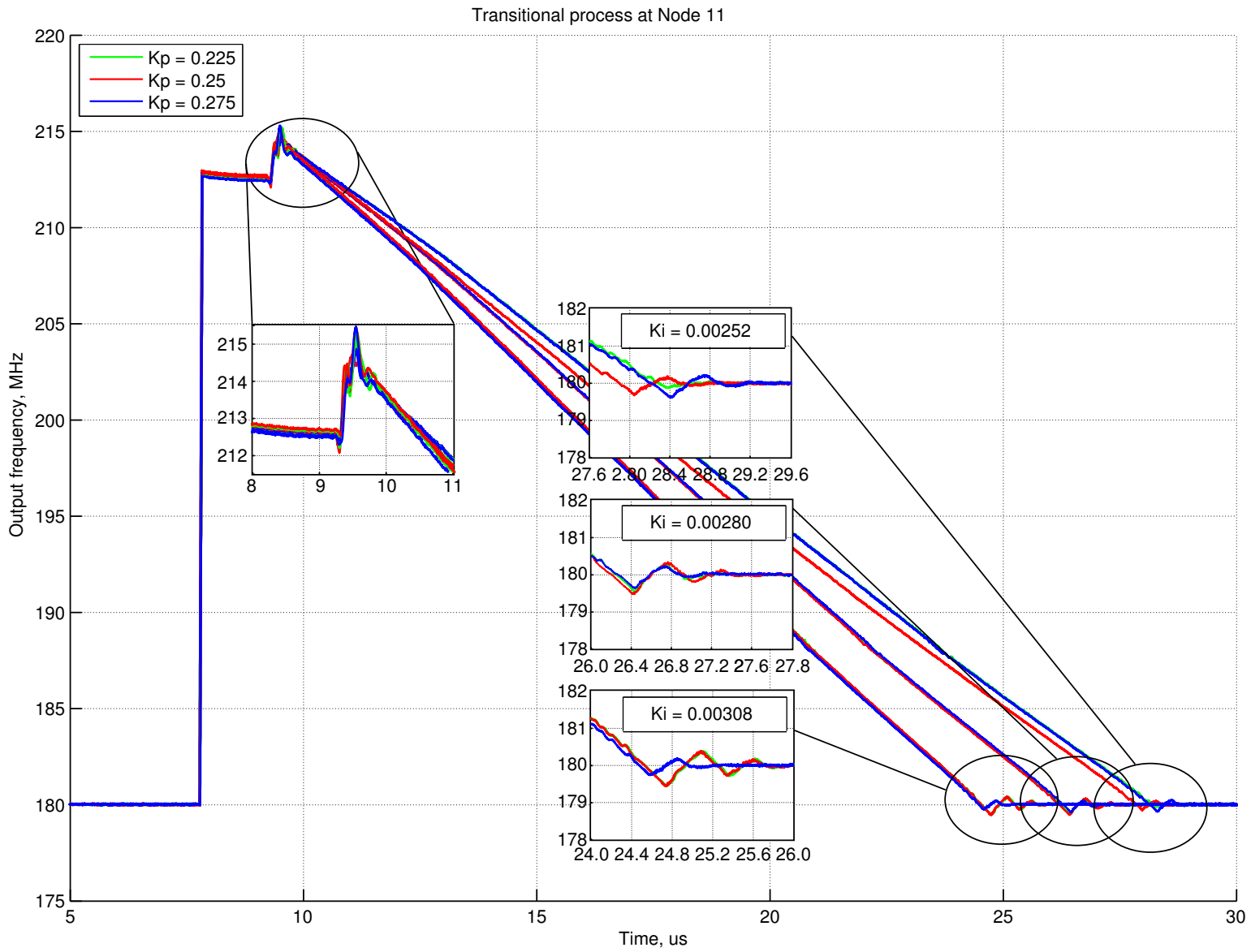


Figure 5.33: Transitional process in the Node 11 depending on variation of the coefficient K_p and K_i

Table 5.3: Network test chip measurements summary

<i>Parameter</i>	<i>Value</i>
Central frequency of the SCA	870 MHz
Frequency range	550~1190 MHz
Timing error ¹	< 60 ps
Convergence rate ²	$\approx 5 \text{ MHz}/\mu\text{s}$
Supply voltage	1.2 V
Power consumption ³	186.2 mW @ $F_{clk} = 800 \text{ MHz}$
Clocking core area	$\sim 0.72 \text{ mm}^2$
Chip area	$\sim 2.04 \text{ mm}^2$

¹ between neighbor nodes² with coefficients mentioned in Tab. 5.2³ analog and digital

5.5 Conclusion

In this chapter we presented two implementations of 16 node network of the proposed clocking system. Both have demonstrated a validity of the studied clocking solution and validated the results of the theoretical study.

The first prototype is based on a FPGA Altera Cyclon II platform. The principal challenges of this experiment are the implementation and emulation of the mixed-signal blocks such as TDC and DCO in a 100 % digital environment. A frequency downscaling allowed an implementation of ADPLL network which is isomorphic with the original ASIC based implementation, and has similar behavior although operating at lower frequencies. The measurement results show that the operation of the proposed clock generator is similar to what predicted in theoretical studies (presence of mode-locking, numeric parameters of the transient, validity of the developed methods allowing a desirable state selection, etc.). The results of the work on FPGA implementation were published in proceedings of FPT2011 conference [61].

The second prototype implements the same ADPLL network architecture in CMOS 65 nm technology, employing already designed blocks of the ADPLL. The objectives were to assess the area, power and performance of the system.

The test of the fabricated circuit shows that a silicon implementation of digital distributed clock generator is feasible. The results of measurements demonstrated that clocking network has good matching with theoretical studies. Surprisingly, the impact of two negative phenomenons predicted in theory has not been confirmed. Firstly, undesirable synchronized states (mode-locking), were not observed during operation of the circuit in a normal bidirectional configuration with full-connected network. It has been observed only in a small 4-nodes network where, according to the theory, it can exist with a high probability. The second effect is related to the variation of the filter coefficients. In contrast with what is expected by the theory, a $\pm 10\%$ variation of these coefficients does not affect the performances of the network.

Finally, the results of performance measurement show that clocking network has timing error between neighboring clocking domain as demanded in specification.

Chapter 6

Conclusion and Perspectives

6.1 Thesis summary and conclusions

The role of the clocking system is vital for high performance chips. With evolution of silicon technologies this role even rises, since the difficulties come from the decreasing ratio between period of clock signals and delay of the wires: they become compatible and design of the clocking system becomes a very challenging task. Studying the state of the art, we have analyzed different clocking techniques, which nowadays have variety of principles of operation. In addition, being concurrent, they sometimes complement each other, since there is no ideal or universal solution for variety of modern digital chips.

The objective of this research was to study the feasibility of the synchronous approach in modern multiprocessor SoCs. The theoretical study, analysis, various simulations on different abstraction levels, FPGA prototyping and CMOS implementation have been the principal bricks of this work.

The research showed that it is possible to continue use of synchronous approach with a multioscillator clocking technique based on network of distributed all-digital phase locked loops. In comparison to numerous conventional clocking techniques, proposed clocking scheme outperforms them. It is more reliable, scalable and can be implemented in a large fully synchronous SoCs. One of the strongest features of such a system is the absence of skew: with the network dimensions growth the skew remains negligible (even for emerging integration solutions as 3D ICs [47]). The jitter attenuation is another strong feature. This is possible thanks to the coupling of oscillators in a phase domain via the distributed ADPLLs, which attenuate the jitter and restore the local clocks. Thus, even with noisy reference clock source, the ADPLLs mitigate its impact and local clocks stay unaffected.

The system level simulations of the developed VHDL and Matlab models helped to verify and justify the mathematical background of this work performed by members of the research team in a context of HODISS project. Thanks to this cooperation, the technique has been sufficiently studied, characterized. That provided enough support for the implementation steps.

As a initial step of implementation, the FPGA prototyping of the system allowed to verify developed distributed clocking network. We have modeled different possible cases of start-

up, perturbations during steady state operation and proposed techniques of mode-locking elimination.

During the research we have developed in 65 nm CMOS technology all main building blocks of proposed distributed ADPLL: a digitally controlled oscillator, a loop filter and a phase/frequency detector. These blocks mainly use digital cells from the foundry standard libraries. This means that first, the system design flow is a common with SoC implementation flow, and second, digital nature of the system offers the portability, reconfigurability and compatibility with DFT.

The designed DCO, remaining a mixed circuit, requires more attention during implementation and a special design flow. We proposed a cell-based design flow, which helps to reduce the efforts of designer comparing to other fully custom design flows. The flow has been verified and proven to be reliable by a successful implementation of the oscillator test circuit, followed by the measurement and characterization steps.

The designed clocking network has been fabricated in 65 nm CMOS technology. The characterization of the test circuit showed that distributed clocking network is fully operational and demonstrates parameters with a good matching with specification. The measured timing error between neighboring nodes in a fully connected bidirectional configuration was less than ± 2 steps of the PFD resolution, i.e. 60 ps. In a unidirectional configuration, as was predicted in Section 2.3.2, the error is slightly higher and measured to be ± 7 steps of the PFD. It is important to mention, that the problem of mode-locking pointed by Gill Pratt in [49] has never been observed in the bidirectional configuration. Therefore, proposed techniques of desired mode selection (c.f. Section 2.3.2) have not been tested. Moreover, in the network configuration where according to the theoretical study mode-locking should appear with a high probability, the problem of false locking has been observed only 4 times amongst 500. This result, despite discordance with theory, shows the reliability of the distributed clocking approach. The measured frequency limits of the network are 550~1190 MHz. The frequency tuning step according to the measurements for DCO and network chip prototypes is 1.482 MHz/LSB. The clocking network consumes 186.2 mW at nominal supply voltage and 800 MHz clock.

Finally, while more and more massively parallel architectures (e.g. Network-on-Chip (NoC) [70]) gain ground on the field of high performance computing, our clocking network will easily adapt and satisfy the requirements of this trend. As a last word, we think that studied approach deserves to evolve from the emerging technique to the state of the art and become common technique of the clock generation and distribution in a future SoCs.

6.2 Future work

As we stated in introduction, the objectives of this research have not included the development of the ready-to-use system. Thus, the logical scenario for the continuation of the research in this area aims the integration of proposed clocking system in a real academic or industrial multiprocessor SoC. This is possible in collaboration with other research teams, institutions or industrial players.

Other natural evolution of this work are various optimizations and improvements of existing architecture on different levels. The system has a large degree of freedom for the configuration of the network and its nodes. It is possible to extend the functionality, increase the performance and reliability of the system.

For this moment, we have spent some time on analysis of the obtained results and we can propose several improvement on different hierarchical levels. They concern architecture of the network itself, or the architecture of its nodes, or the design flow:

System level improvements

Analyzing the result of simulation of the 16-node network, we found that frequency acquisition time is too large. In fact, this parameter has not been a part of the specification at the beginning of the research. The clocking network supposed to work at the fixed frequency. For the future implementation, where we will implement dynamic scaling of frequency, this parameter will be critical. In a current network configuration it roughly depends on the integer coefficient of the loop filter: as smaller it is, as slower the frequencies are tuned. In particular, it is possible to implement a combination of techniques exploited in [42] (but transferred to digital domain) for global control and our technique for compensation of inaccuracies as a local control.

For better compatibility and integrability of the system with SoC it is possible to implement a control interface/system for our clocking network. According to the current load, processor core can scale the local frequency through this interface, and respectively scale the consumed power.

The lock/unlock indicators will be also suitable for future implementation. These indicators will told to the SoC that all clocks are synchronous and blocks between SCAs could operate in a normal synchronous mode.

Circuit level improvements

As we mentioned in chapter dedicated to design of oscillator, the DCO is a most challenging block of the ADPLL. The DCO implemented during this work can be improved. It was designed only for this particular implementation as a demonstrator. Thus, it will be reasonable to pay more attention to basic characteristics, so its figure-of-merit will be compatible with the state-of-the-art oscillators.

On our opinion, the most promising and suitable evolution of DCO is differential ring architecture. This type of oscillator rejects the noises from supply and substrate, it has odd

number of stages (control circuitry is less complex), it demonstrates better noise characteristics. Moreover, the cell-based methodology developed during this thesis could be applied on this oscillator with minimal efforts.

The second critical block is PFD. Its resolution can be improved; the number of quantification steps could be increased while the measurement range remained the same. This feature will decrease the timing error between the SCAs and increase the precision of on-chip measurements, necessary for the characterization of the clock signals.

Methodology

Most of the efforts during the implementation cycle were spent on a DCO. With the proposed cell-based approach, the design still requires a lot of custom actions. The general design flow is not optimized and automated. Therefore, in order to provide designers an efficient instrument, it is reasonable to develop a fully automated design flow. According to the required properties, the tool or script must automatically and with minimal interaction with user, generate a complete layout of the oscillator or at least its core part.

Appendices

Appendix A

VHDL models of the ADPLL blocks

This appendix presents the behavioral and RTL-level VHDL models of the ADPLL blocks: a BB-detector, a TDC, a digital signal processing block (i.e. loop filter) and DCO.

Listing A.1: VHDL model of the digitally-controlled oscillator

```
=====
-- Title       : DCO.VHDL
-- Design      : ADPLL
-- Author       : Eldar Zianbetov
-- Revision    : 2.2
=====
-- File        : DCO.VHDL.vhd
-- Generated   : Fri Nov 19 20:31:11 2010
=====
-- Description : DCO VHDL model with transistor level model close parameters
-- parameters
=====
-- Revision history:
-- 1.0: First implementation of the digitally controlled period oscillator.
-- Nonlinear. Simplified. Integer input.
-- 1.1: Digitally controlled frequency oscillator. Only DIV clock generation.
-- 1.2: Adding jitter with a Gaussian distribution.
-- 2.0: Very precise model. Jitter and wander was implemented.
-- CLK and DIV clocks generation. Real dividers. Thermometer code input.
-- 2.1: Implementation of the nonlinearities of the real transistor model
-- DCO via lookup table. Approximation for each 10 steps.
-- 2.2: DIV16 added.
=====

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
use IEEE.MATHREAL.all;
use IEEE.NUMERIC_STD.all;

entity DCO is
    generic (
```

```

        RESOLUTION : time := 1fs;           — finest time resolution, fs
        DCO_INIT_DELAY: time := 1ps;       — initial oscillation delay, ns
        DELTA_F: real := 0.0;             — frequency offset, MHz
        DCO_JRMS: time := 1821fs;        — jitter rms, fs — 1821fs
        DCO_WRMS: time := 0fs            — wander rms, ps
    );

    port (
        C : in STD_LOGIC_VECTOR(2 downto 0);
        B : in STD_LOGIC_VECTOR(6 downto 0);
        A : in STD_LOGIC_VECTOR(30 downto 0);
        CLK, DIV, DIV16: out std_logic
    );
end DCO ;

```

architecture behavioral of DCO is

```

=====
    signal enable : std_logic := '0';
    signal DCO_IN : STD_LOGIC_VECTOR(9 downto 0) := "0000000000";
    signal DCO_CODE : integer range 0 to 1023 := 0;
    signal frequency_dco : real := 1.0;
    signal smp: bit := '0';
    signal period0: time := 1ns;
    signal Q1, Q2, Q3: std_logic := '0';
    signal D1, D2, D3: std_logic := '1';
    signal CLK_TMP: std_logic;
    constant Fdco0: real := 596.6261802347145;
    constant Fdco10: real := 604.6399058580719;
    constant Fdco20: real := 612.2181039477546;
    constant Fdco30: real := 620.4924863866559;
    constant Fdco40: real := 628.587191012035;
    constant Fdco50: real := 636.2879874148636;
    constant Fdco60: real := 644.6599293553946;
    constant Fdco70: real := 652.528101193582;
    constant Fdco80: real := 660.3392023106581;
    constant Fdco90: real := 668.2849922747387;
    constant Fdco100: real := 676.7485968717576;
    constant Fdco110: real := 684.3754151175424;
    constant Fdco120: real := 692.4725202175025;
    constant Fdco130: real := 700.6542187934881;
    constant Fdco140: real := 708.6183730993197;
    constant Fdco150: real := 716.2698439754596;
    constant Fdco160: real := 724.8272897614409;
    constant Fdco170: real := 732.7955415139165;
    constant Fdco180: real := 740.4014607104778;
    constant Fdco190: real := 749.0200290437814;
    constant Fdco200: real := 757.2036658964837;
    constant Fdco210: real := 764.9341524994047;
    constant Fdco220: real := 773.359395261577;
    constant Fdco230: real := 781.8198184056989;
    constant Fdco240: real := 789.7084525747032;

```

```
constant Fdco250: real := 797.5825888966222;  
constant Fdco260: real := 806.0983541332629;  
constant Fdco270: real := 814.0452456985863;  
constant Fdco280: real := 821.9643602101477;  
constant Fdco290: real := 830.3470986487244;  
constant Fdco300: real := 838.2637440954217;  
constant Fdco310: real := 845.9910302424402;  
constant Fdco320: real := 854.5484969954177;  
constant Fdco330: real := 862.3366519735556;  
constant Fdco340: real := 870.1389392374318;  
constant Fdco350: real := 878.5219433898591;  
constant Fdco360: real := 886.4225655077747;  
constant Fdco370: real := 893.9615815882334;  
constant Fdco380: real := 902.4867221610754;  
constant Fdco390: real := 910.390209891708;  
constant Fdco400: real := 918.0653098147459;  
constant Fdco410: real := 925.8766562964984;  
constant Fdco420: real := 934.4690159624928;  
constant Fdco430: real := 942.2005903433492;  
constant Fdco440: real := 949.8322803303478;  
constant Fdco450: real := 958.6047566692444;  
constant Fdco460: real := 966.5573328045971;  
constant Fdco470: real := 974.0930784632939;  
constant Fdco480: real := 982.5471932834675;  
constant Fdco490: real := 990.2753779025781;  
constant Fdco500: real := 997.9453681687198;  
constant Fdco510: real := 1006.0952150554171;  
constant Fdco520: real := 1013.9637470799168;  
constant Fdco530: real := 1021.6146601703941;  
constant Fdco540: real := 1029.8530847095792;  
constant Fdco550: real := 1037.6680775515226;  
constant Fdco560: real := 1045.2588391728674;  
constant Fdco570: real := 1052.956809356187;  
constant Fdco580: real := 1061.2167423438319;  
constant Fdco590: real := 1068.713512804244;  
constant Fdco600: real := 1076.4193216361084;  
constant Fdco610: real := 1084.5420550369508;  
constant Fdco620: real := 1092.2362914617662;  
constant Fdco630: real := 1099.5438822277718;  
constant Fdco640: real := 1108.0608013763468;  
constant Fdco650: real := 1115.7192882295063;  
constant Fdco660: real := 1123.1398645889263;  
constant Fdco670: real := 1131.2060287685661;  
constant Fdco680: real := 1139.4984412599516;  
constant Fdco690: real := 1146.925892019041;  
constant Fdco700: real := 1154.8838154127665;  
constant Fdco710: real := 1162.6814510097015;  
constant Fdco720: real := 1170.3092011299474;  
constant Fdco730: real := 1177.7856009467657;  
constant Fdco740: real := 1185.9799565277052;
```

```

constant Fdco750: real := 1193.405596128358;
constant Fdco760: real := 1200.9931704039638;
constant Fdco770: real := 1209.0006376990383;
constant Fdco780: real := 1216.5725210572038;
constant Fdco790: real := 1223.9319797621741;
constant Fdco800: real := 1232.1807645971317;
constant Fdco810: real := 1239.6253519056654;
constant Fdco820: real := 1247.0384995420187;
constant Fdco830: real := 1255.0318831570103;
constant Fdco840: real := 1262.796075819176;
constant Fdco850: real := 1270.0551493554792;
constant Fdco860: real := 1278.144577365454;
constant Fdco870: real := 1285.7962979659781;
constant Fdco880: real := 1293.2958956610794;
constant Fdco890: real := 1300.6314471199796;
constant Fdco900: real := 1309.22883473;
constant Fdco910: real := 1316.6530798258746;
constant Fdco920: real := 1324.1472392812693;
constant Fdco930: real := 1332.1043164674122;
constant Fdco940: real := 1339.7494132551067;
constant Fdco950: real := 1346.9673435586197;
constant Fdco960: real := 1355.103842137449;
constant Fdco970: real := 1362.4293092613063;
constant Fdco980: real := 1369.9724275430524;
constant Fdco990: real := 1377.8068718008845;
constant Fdco1000: real := 1385.4544242791204;
constant Fdco1010: real := 1392.6702518570743;
constant Fdco1020: real := 1400.8027386009467;
constant Fdco1023: real := 1402.7880888557286;

```

begin

=====

DECODER.A:

```

with A select — turning back to binary...
DCO_IN(9 downto 5) <= "00000" when "00000000000000000000000000000000",
"00001" when "00000000000000000000000000000001" ,
"00010" when "00000000000000000000000000000011" ,
"00011" when "00000000000000000000000000000111" ,
"00100" when "00000000000000000000000000000111" ,
"00101" when "000000000000000000000000000001111" ,
"00110" when "0000000000000000000000000000011111" ,
"00111" when "0000000000000000000000000000011111" ,
"01000" when "0000000000000000000000000000111111" ,
"01001" when "0000000000000000000000000000111111" ,
"01010" when "00000000000000000000000000001111111" ,
"01011" when "000000000000000000000000000011111111" ,
"01100" when "000000000000000000000000000011111111" ,
"01101" when "000000000000000000000000000011111111" ,
"01110" when "000000000000000000000000000011111111" ,
"01111" when "000000000000000000000000000011111111" ,

```

```

"10000" when "00000000000000001111111111111111" ,
"10001" when "00000000000000001111111111111111" ,
"10010" when "00000000000000001111111111111111" ,
"10011" when "00000000000000001111111111111111" ,
"10100" when "00000000000011111111111111111111" ,
"10101" when "00000000000011111111111111111111" ,
"10110" when "00000000001111111111111111111111" ,
"10111" when "00000000011111111111111111111111" ,
"11000" when "00000001111111111111111111111111" ,
"11001" when "00000011111111111111111111111111" ,
"11010" when "00000111111111111111111111111111" ,
"11011" when "00001111111111111111111111111111" ,
"11100" when "00011111111111111111111111111111" ,
"11101" when "00111111111111111111111111111111" ,
"11110" when "01111111111111111111111111111111" ,
"11111" when "11111111111111111111111111111111" ,
"_____" when others ;

```

DECODER_B:

```

with B select
DCO_IN(4 downto 2) <= "000" when "0000000" ,
"001" when "0000001" ,
"010" when "0000011" ,
"011" when "0000111" ,
"100" when "0001111" ,
"101" when "0011111" ,
"110" when "0111111" ,
"111" when "1111111" ,
"____" when others ;

```

DECODER_C:

```

with C select
DCO_IN(1 downto 0) <= "00" when "000" ,
"01" when "001" ,
"10" when "011" ,
"11" when "111" ,
"____" when others ;

```

BINARY_TO_INTEGER_CONVERTER:

```

DCO_CODE <= conv_integer(DCO_IN);           — turning back to integer ...

```

CODE_TO_FREQUENCY_CONVERTER:

```

with DCO_CODE select
frequency_dco <= DELTA_F+Fdco0 when 0,
DELTA_F+(Fdco0+(Fdco10-Fdco0)*0.1) when 1,
DELTA_F+(Fdco0+(Fdco10-Fdco0)*0.2) when 2,
DELTA_F+(Fdco0+(Fdco10-Fdco0)*0.3) when 3,
DELTA_F+(Fdco0+(Fdco10-Fdco0)*0.4) when 4,
DELTA_F+(Fdco0+(Fdco10-Fdco0)*0.5) when 5,
DELTA_F+(Fdco0+(Fdco10-Fdco0)*0.6) when 6,

```

```

DELTA_F+(Fdco0+(Fdco10-Fdco0)*0.7) when 7,
DELTA_F+(Fdco0+(Fdco10-Fdco0)*0.8) when 8,
DELTA_F+(Fdco0+(Fdco10-Fdco0)*0.9) when 9,
-----
---====APPROXIMATION TABLE
---==== *****
-----
DELTA_F+(Fdco1010+(Fdco1020-Fdco1010)*0.1) when 1011,
DELTA_F+(Fdco1010+(Fdco1020-Fdco1010)*0.2) when 1012,
DELTA_F+(Fdco1010+(Fdco1020-Fdco1010)*0.3) when 1013,
DELTA_F+(Fdco1010+(Fdco1020-Fdco1010)*0.4) when 1014,
DELTA_F+(Fdco1010+(Fdco1020-Fdco1010)*0.5) when 1015,
DELTA_F+(Fdco1010+(Fdco1020-Fdco1010)*0.6) when 1016,
DELTA_F+(Fdco1010+(Fdco1020-Fdco1010)*0.7) when 1017,
DELTA_F+(Fdco1010+(Fdco1020-Fdco1010)*0.8) when 1018,
DELTA_F+(Fdco1010+(Fdco1020-Fdco1010)*0.9) when 1019,
DELTA_F+Fdco1020 when 1020,
DELTA_F+(Fdco1020+(Fdco1023-Fdco1020)*1.0/3.0) when 1021,
DELTA_F+(Fdco1020+(Fdco1023-Fdco1020)*2.0/3.0) when 1022,
DELTA_F+Fdco1020 when 1023,
1000.0 when others;
-----
PERIOD_SYNTHESIZER:
    --- calculating the period of oscillation
    period0 <= RESOLUTION * (1000000000.0/frequency_dco);
-----
PERIOD_CONTROLLED_OSCILLATOR:
    process (smp)
        variable initial: boolean := true;
        variable jitter: time := 0ns;
        variable jitter_prev: time := 0ns;
        variable wander: time := 0ns;
        variable period: time := 1ns;
        variable s1, s2, s3, s4: positive;
        variable x1, x2, x3, x4, randvar1, randvar2: real := 0.0;
    begin
        if initial then
            ---adjust the next period
            period := period0;
            ---add Gaussian-distributed jitter
            uniform(s1, s2, x1);
            uniform(s1, s2, x2);
            randvar1 := sqrt(-2.0*log(x1))*cos(2.0*MATH.PI*x2);
            jitter := randvar1 * DCO_JRMS;
            period := period+jitter-jitter_prev;
            jitter_prev := jitter;
            ---add Gaussian-distributed wander
            uniform(s3, s4, x3);
            uniform(s3, s4, x4);
            randvar2 := sqrt(-2.0*log(x3))*cos(2.0*MATH.PI*x4);

```

```

wander := randvar2 * DCO.WRMS;
period := period+wander;
--clock with 50% duty cycle
--CLK <= '1','0' after period/2;
smp <= not smp after period;
--temp internal clk 50% duty cycle
CLK.TMP <= '1','0' after period/2;

else
period := period0+DCO_INIT_DELAY; -- CURRENTLY NOT IN USE
--CLK <= '0'; -- CURRENTLY NOT IN USE
CLK.TMP <= '0'; -- CURRENTLY NOT IN USE
smp <= '1'; -- first transition
initial := false; -- CURRENTLY NOT IN USE
end if;
end process;
=====
-- operational delay 1ps
enable <= '1' after 1ps;
=====
-- output of high freq. clock
CLK <= CLK.TMP and enable;
=====
DIVIDER_2_1:
process (CLK.TMP)
begin
if CLK.TMP'event and CLK.TMP='1' then
Q1 <= D1;
end if;
D1 <= not Q1;
end process;
=====
DIVIDER_2_2:
process (Q1)
begin
if Q1'event and Q1='1' then
Q2 <= D2;
end if;
D2 <= not Q2;
end process;
=====
-- output after divider on 4
DIV <= Q2 and enable;
=====
DIVIDER_2_3:
process (Q2)
begin
if Q2'event and Q2='1' then
Q3 <= D3;
end if;
D3 <= not Q3;

```



```

end process ;
=====
    — output after divider on 8
    DIV16 <= Q3 and enable ;
=====
end behavioral ;

```

Listing A.2: VHDL model of the sign detector

```

library IEEE ;
use IEEE.STD_LOGIC_1164.all ;
use IEEE.MATHREAL.all ;
use IEEE.NUMERIC_STD.all ;
use IEEE.STD_LOGIC_ARITH.all ;
use IEEE.STD_LOGIC_UNSIGNED.all ;

entity sign_detector is
    port (
        REF, CLK: in STD_LOGIC ;
        SIGN, nSIGN, TDCIN, TDCCLK: out STD_LOGIC
    );
end sign_detector ;

architecture behavioral of sign_detector is
    — generating of the constants and signals —————
    signal DELAY : TIME := 1 ps ;
    constant VCC.CONSTANT : STD_LOGIC := '1' ;
    signal Ao, Bo, D, Q1, A_S, Q2, B_R ;
    signal A_first, B_first, nQ3, Q3, Q1t, Q2t : STD_LOGIC ;
    signal NET1, NET2, OUT_C, RESET_LOCAL : STD_LOGIC ;
    signal TDCCLK_nondelayed, TDCIN_nondelayed : std_logic ;

    signal ERR : STD_LOGIC ;

begin
    D <= VCC.CONSTANT ;    — high level to the D-inputs of the latches
    =====
D_FLIP_FLOP_1 :
    Q1t <= '0' when RESET_LOCAL='1' else                — latch
    D when (REF'event and REF='1') else Q1 after DELAY*1 ;
    A_S <= not Q1 after DELAY*1 ;
    Q1 <= Q1t after DELAY*290 ;
    =====
D_FLIP_FLOP_2 :
    Q2t <= '0' when RESET_LOCAL='1' else                — latch
    D when (CLK'event and CLK='1') else Q2 after DELAY*1 ;
    B_R <= not Q2 after DELAY*1 ;
    Q2 <= Q2t after DELAY*290 ;
    =====
RS_FLIP_FLOP :
    — INPUT SR FLIP-FLOP on NOR gates
    process (A_S, B_R)

```

```

    — variables for pseudo-random number generator
    variable seed1, seed2: positive;
    variable rand: real;
    variable rnd: STD_LOGIC;

begin
RANDOM_NUMBER_GENERATOR:      — pseudo-random number generator
    UNIFORM(seed1, seed2, rand);
    if (rand <= 0.5) then
        rnd := '1';
    else
        rnd := '0';
    end if;
— Description of RS-latch
if (A_S = '0') and (B_R = '0') and (Bo = Ao) then
    Ao <= rnd after DELAY*10;      — assign of rnd nmb
    Bo <= not rnd after DELAY*10;  — in case of two '0'
elsif (A_S = '0') and (B_R = '0') then
    Ao <= Ao after DELAY;
    Bo <= Bo after DELAY;
elsif (A_S = '1') and (B_R = '1') then
    Ao <= '0' after DELAY;
    Bo <= '0' after DELAY;
elsif A_S = '1' then
    Ao <= '1' after DELAY;
    Bo <= '0' after DELAY;
elsif B_R = '1' then
    Bo <= '1' after DELAY;
    Ao <= '0' after DELAY;
end if ;
end process;
FILTER: — Description of metastability filter
process (Ao,Bo)
begin
if (Ao = '0') and (Bo = '0') then
    A_first <= '1' after DELAY;
    B_first <= '1' after DELAY;
elsif (Ao = '0') and (Bo = '1') then
    A_first <= '1' after DELAY;
    B_first <= '0' after DELAY;
elsif (Ao = '1') and (Bo = '0') then
    A_first <= '0' after DELAY;
    B_first <= '1' after DELAY;
elsif (Ao = '1') and (Bo = '1') then
    A_first <= '1' after DELAY;
    B_first <= '1' after DELAY;
end if;
end process;
=====
SR_FLIP_FLOP: — store flip-flop
nQ3 <= not(B_first and Q3) after DELAY*1;

```

```

Q3 <= not(A_first and nQ3) after DELAY*1;
=====
LOGIC_BLOCK:  — logic NOR and OR elements
NET1 <= (not(A_first or nQ3)) or (not(B_first or Q3));
=====
C.ELEMENT:    — C-element
OUT.C <= '1' when (NET1='1' and Q1='1' and Q2='1') else
'0' when (NET1='0' and Q1='0' and Q2='0') else OUT.C;
=====
RESET_LOGIC:
RESET.LOCAL <= OUT.C; — reset OR element
=====
TDCIN <= not(Q1 nor Q2);      — generating of the measure bit
TDCCLK <= not(Q1 nand Q2);
nSIGN <= Q3;
SIGN <= nQ3;
=====

end behavioral;

```

Listing A.3: VHDL model of the TDC

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.MATH_REAL.all;
use IEEE.NUMERIC_STD.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity TDC_LIP6 is
  port (
    TDCCLK, TDCIN, TDCRES: in STD_LOGIC;
    S : out STD_LOGIC_VECTOR (13 downto 0)
  );
end TDC_LIP6;

architecture behavioral of TDC_LIP6 is
  — generating of the constants and signals —————
  signal D.TDC: STD_LOGIC_VECTOR (14 downto 0);
  signal Q.TDC: STD_LOGIC_VECTOR (13 downto 0); — latch signal in TDC
begin
  =====

  Q.TDC(0) <= TDCIN after 55ps;
  Q.TDC(1) <= Q.TDC(0) after 28ps;
  Q.TDC(2) <= Q.TDC(1) after 38ps;
  Q.TDC(3) <= Q.TDC(2) after 30ps;
  Q.TDC(4) <= Q.TDC(3) after 32ps;
  Q.TDC(5) <= Q.TDC(4) after 30ps;
  Q.TDC(6) <= Q.TDC(5) after 30ps;

```

```

        Q.TDC(7) <= Q.TDC(6) after 30ps;
        Q.TDC(8) <= Q.TDC(7) after 30ps;
        Q.TDC(9) <= Q.TDC(8) after 30ps;
        Q.TDC(10) <= Q.TDC(9) after 30ps;
        Q.TDC(11) <= Q.TDC(10) after 30ps;
        Q.TDC(12) <= Q.TDC(11) after 30ps;
        Q.TDC(13) <= Q.TDC(12) after 30ps;

    process (TDCCLK, TDCRES)
    begin
        if TDCRES='0' then
            S <= "0000000000000000";
        elsif (TDCCLK'event and TDCCLK='1') then
            S(13 downto 0) <= Q.TDC(13 downto 0);
        end if;
    end process;
end behavioral;

```

Listing A.4: VHDL model of the digital processing block

```

=====
--
-- Title       : Error signal processing block
-- Design      : ADPLL
-- Author      : Eldar Zianbetov
-- Revision    : 1.2
--
=====
--
-- File        : filter.vhd
-- Generated   : Fri Nov 18 20:31:11 2010
--
=====
--
-- Description : Synthesizable model of the error processing block
-- 1- block of coefficients programming
-- 2- block of PI filter
-- 3- block of DCO B2T decoders
--
=====
--
-- Revision history:
--
=====

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.MATHREAL.all;
use IEEE.STD_LOGIC_ARITH.all;

entity FILTER_LIP6 is

```

```

port (
    CLKFLT, RESFLT, INFIN, CLKINFO, INFEN :in STD_LOGIC;
    ERRORIN1, ERRORIN2 : in std_logic_vector (4 downto 0);
    ERRORIN3, ERRORIN4 : in std_logic_vector (4 downto 0);
    INFOUT :out STD_LOGIC;
    PIOUT : out std_logic_vector (9 downto 0);
    A : out std_logic_vector (30 downto 0);
    B : out std_logic_vector (6 downto 0);
    C : out std_logic_vector (2 downto 0);
    KPout : out std_logic_vector (4 downto 0)
);
end FILTER_LIP6;

architecture rtl of FILTER_LIP6 is
    — Signals for block of coefficients programing
    signal COEFS0 : std_logic_vector (24 downto 0);
    — Kint (12 bit)+Kprop(5 bit)+KW1(2 bit)+KW2(2 bit)+KW3(2 bit)+KW4(2 bit)
    signal COEFS : std_logic_vector (24 downto 0);
    signal kintI: integer range 0 to 2**12-1;
    signal kpropI: integer range 0 to 2**5-1;
    signal ERROR1, ERROR2, ERROR3, ERROR4: integer range -2**4 to 2**4-1;
    — Signals for PI filter
    — KINT: from 1/2^12 to (2^12-1)/2^12 (0.000244141 to 0.999755859)
    — KPROP: from 1/2^5 to (2^5-1)/2^5 (0.03125 to 0.96875)
    signal adder_in_1 , adder_in_2 : integer range -2**6 to 2**6-1;
    signal adder_in_3 , adder_in_4 : integer range -2**6 to 2**6-1;
    signal TOTALERROR: integer range -2**8 to 2**8-1;
    signal TOTALERRORDELAYED: integer range -2**8 to 2**8-1;
    signal v_xi_K_prop: integer range -2**13 to 2**13-1;
    signal v_xi_K_int: integer range -2**20 to 2**20-1;
    signal s_v_xi , s_v_xi_m1: integer range -2**21 to 2**21-1 := 0;
    signal PIOUTI : integer range 0 to 2**10-1;
    signal PIOUT_BIN : std_logic_vector (9 downto 0);
    — Signals for DCO decoder
    signal COL1 : STD_LOGIC;
    signal COL2 : STD_LOGIC;
    signal COL3 : STD_LOGIC;
    signal ROW1 : STD_LOGIC;
    signal ROW2 : STD_LOGIC;
    signal ROW3 : STD_LOGIC;
    signal ROW4 : STD_LOGIC;
    signal ROW5 : STD_LOGIC;
    signal ROW6 : STD_LOGIC;
    signal ROW7 : STD_LOGIC;
    signal C_BIN: std_logic_vector (1 downto 0);
    signal B_BIN: std_logic_vector (2 downto 0);
    signal A_BIN: std_logic_vector (4 downto 0);
    signal A_NONDELAYED : std_logic_vector (30 downto 0);
    signal B_NONDELAYED : std_logic_vector (6 downto 0);

```

```

signal C_NONDELAYED : std_logic_vector (2 downto 0);

begin
=====
===== Coefficients programming =====
===== SHIFT REGISTER (COEFFICIENTS REGISTER (CR))=====
=====
-- The input (INFIN) come from the last programing block and the output
-- (INFOUT) is gone to the next programing block
-- Order COEFS(24-0): Kint(24-13), Kprop(12-8), KW1(7-6), KW2(5-4), KW3(3-2), KW4(1-0)
CR: -- Dynamic modification
    process (CLKINFO)
    begin
        if (CLKINFO'event and CLKINFO='1') then
            COEFS0 <= INFIN & COEFS0(COEFS0'left downto 1);
        end if;
    end process;
SR: -- technological delay on  $z^{-0.5}$ 
    process (CLKINFO)
    begin
        if (CLKINFO'event and CLKINFO='0') then
            INFOUT <= COEFS0(0);
        end if;
    end process;
OUTR: -- Output register
    process (INFEN)
    begin
        if (INFEN'event and INFEN='1') then
            COEFS <= COEFS0;
        end if;
    end process;
=====
-- Converting the std_logic input error to integer number

    ERROR1 <= conv_integer(signed(ERRORIN1));
    ERROR2 <= conv_integer(signed(ERRORIN2));
    ERROR3 <= conv_integer(signed(ERRORIN3));
    ERROR4 <= conv_integer(signed(ERRORIN4));

    kintI <= conv_integer(unsigned(COEFS(24 downto 13)));
    kpropI <= conv_integer(unsigned(COEFS(12 downto 8)));
    KPout(4 downto 0) <= COEFS(12 downto 8);
=====
===== ARITHMETIC BLOCK =====
=====
ARITH:
    -- input weighting and addition:

    adder_in_1 <= 0 when (COEFS(6)='0' and COEFS(7)='0') else

```

```

ERROR1 when (COEFS(6)='1' and COEFS(7)='0') else
ERROR1*2 when (COEFS(6)='0' and COEFS(7)='1') else
ERROR1*4 when (COEFS(6)='1' and COEFS(7)='1') else
0;

```

```

adder_in_2 <= 0 when (COEFS(4)='0' and COEFS(5)='0') else
ERROR2 when (COEFS(4)='1' and COEFS(5)='0') else
ERROR2*2 when (COEFS(4)='0' and COEFS(5)='1') else
ERROR2*4 when (COEFS(4)='1' and COEFS(5)='1') else
0;

```

```

adder_in_3 <= 0 when (COEFS(2)='0' and COEFS(3)='0') else
ERROR3 when (COEFS(2)='1' and COEFS(3)='0') else
ERROR3*2 when (COEFS(2)='0' and COEFS(3)='1') else
ERROR3*4 when (COEFS(2)='1' and COEFS(3)='1') else
0;

```

```

adder_in_4 <= 0 when (COEFS(0)='0' and COEFS(1)='0') else
ERROR4 when (COEFS(0)='1' and COEFS(1)='0') else
ERROR4*2 when (COEFS(0)='0' and COEFS(1)='1') else
ERROR4*4 when (COEFS(0)='1' and COEFS(1)='1') else
0;

```

```

=====
===== FOUR INPUT ADDER AND DIVIDER =====
=====

```

— *Adding errors without averaging*

```
TOTAL_ERROR <=(adder_in_1+adder_in_2+adder_in_3+adder_in_4);
```

— *First register to eliminate the Encoder+Adder glitch*

```

process (RESFLT,CLKFLT)
begin
    if RESFLT='0' then
        TOTAL_ERROR_DELAYED <= 0;
    elsif (CLKFLT'event and CLKFLT='1') then
        TOTAL_ERROR_DELAYED <= TOTAL_ERROR;
    end if;
end process;

```

```

=====
===== PROPORTIONA-TEGRAL FILTER =====
=====

```

FILTER:

— *proportional path:*

```
v_xi_K_prop <= TOTAL_ERROR_DELAYED * kpropI;
```

— *integral path:*

— *adding and multiplying:*

```

v_xi_K_int <= TOTAL_ERROR_DELAYED * kintI;
s_v_xi <= v_xi_K_int + s_v_xi_m1;
-- first delay rising edge
-- latches:
  process (RESFLT,CLKFLT)
  begin
    if RESFLT='0' then
      s_v_xi_m1 <= 0;
    elsif (CLKFLT'event and CLKFLT='1') then
      s_v_xi_m1 <= s_v_xi;
    end if;
  end process;
-- combining and scaling:
-- integral path cover 10 bits output range
PIOUTI <= s_v_xi/2**12 + v_xi_K_prop/2**5 + 512;
PIOUT_BIN <= conv_std_logic_vector(PIOUTI,10);
  process (RESFLT,CLKFLT)
  begin
    if RESFLT='0' then
      PIOUT <= "1000000000";
    elsif (CLKFLT'event and CLKFLT='1') then
      PIOUT <= PIOUT_BIN;
    end if;
  end process;
=====
----- DCO A-, B-, C- channel DECODERS -----
=====
----- Component instantiations -----
C_NONDELAYED(0) <= PIOUT_BIN(1) or PIOUT_BIN(0);
C_NONDELAYED(1) <= PIOUT_BIN(1);
C_NONDELAYED(2) <= PIOUT_BIN(1) and PIOUT_BIN(0);
-----
B_NONDELAYED(0) <= PIOUT_BIN(4) or PIOUT_BIN(3) or PIOUT_BIN(2);
B_NONDELAYED(1) <= PIOUT_BIN(4) or PIOUT_BIN(3);
B_NONDELAYED(2) <= PIOUT_BIN(4) or (PIOUT_BIN(3) and PIOUT_BIN(2));
B_NONDELAYED(3) <= PIOUT_BIN(4);
B_NONDELAYED(4) <= PIOUT_BIN(4) and (PIOUT_BIN(3) or PIOUT_BIN(2));
B_NONDELAYED(5) <= PIOUT_BIN(4) and PIOUT_BIN(3);
B_NONDELAYED(6) <= PIOUT_BIN(4) and PIOUT_BIN(3) and PIOUT_BIN(2);
-----
ROW1 <= not(PIOUT_BIN(9) or PIOUT_BIN(8) or PIOUT_BIN(7));
ROW2 <= (PIOUT_BIN(8) nor PIOUT_BIN(9));
ROW3 <= (PIOUT_BIN(9) nor (PIOUT_BIN(8) and PIOUT_BIN(7)));
ROW4 <= not(PIOUT_BIN(9));
ROW5 <= (PIOUT_BIN(9) nand (PIOUT_BIN(8) or PIOUT_BIN(7)));
ROW6 <= (PIOUT_BIN(9) nand PIOUT_BIN(8));
ROW7 <= not(PIOUT_BIN(9) and PIOUT_BIN(8) and PIOUT_BIN(7));
-----
COL1 <= (PIOUT_BIN(6) nor PIOUT_BIN(5));
COL2 <= not(PIOUT_BIN(6));

```



```
COL3 <= (PIOUT_BIN(6) nand PIOUT_BIN(5));
```

```
A_NONDELAYED(0) <= (ROW1 nand COL1);
A_NONDELAYED(1) <= (ROW1 nand COL2);
A_NONDELAYED(2) <= (ROW1 nand COL3);
A_NONDELAYED(3) <= (ROW2 nand ROW1);
A_NONDELAYED(4) <= (ROW2 nand (COL1 or ROW1));
A_NONDELAYED(5) <= (ROW2 nand (COL2 or ROW1));
A_NONDELAYED(6) <= (ROW2 nand (COL3 or ROW1));
A_NONDELAYED(7) <= (ROW3 nand ROW2);
A_NONDELAYED(8) <= (ROW3 nand (COL1 or ROW2));
A_NONDELAYED(9) <= (ROW3 nand (COL2 or ROW2));
A_NONDELAYED(10) <= (ROW3 nand (COL3 or ROW2));
A_NONDELAYED(11) <= (ROW4 nand ROW3);
A_NONDELAYED(12) <= (ROW4 nand (COL1 or ROW3));
A_NONDELAYED(13) <= (ROW4 nand (COL2 or ROW3));
A_NONDELAYED(14) <= (ROW4 nand (COL3 or ROW3));
A_NONDELAYED(15) <= (ROW5 nand ROW4);
A_NONDELAYED(16) <= (ROW5 nand (COL1 or ROW4));
A_NONDELAYED(17) <= (ROW5 nand (COL2 or ROW4));
A_NONDELAYED(18) <= (ROW5 nand (COL3 or ROW4));
A_NONDELAYED(19) <= (ROW6 nand ROW5);
A_NONDELAYED(20) <= (ROW6 nand (COL1 or ROW5));
A_NONDELAYED(21) <= (ROW6 nand (COL2 or ROW5));
A_NONDELAYED(22) <= (ROW6 nand (COL3 or ROW5));
A_NONDELAYED(23) <= (ROW7 nand ROW6);
A_NONDELAYED(24) <= (ROW7 nand (COL1 or ROW6));
A_NONDELAYED(25) <= (ROW7 nand (COL2 or ROW6));
A_NONDELAYED(26) <= (ROW7 nand (COL3 or ROW6));
A_NONDELAYED(27) <= not (ROW7);
A_NONDELAYED(28) <= (COL1 nor ROW7);
A_NONDELAYED(29) <= (COL2 nor ROW7);
A_NONDELAYED(30) <= (COL3 nor ROW7);
```

```
process (RESFLT,CLKFLT)
```

```
begin
```

```
    if RESFLT='0' then
```

```
        A <= "00000000000000001111111111111111";
```

```
        B <= "0000000";
```

```
        C <= "000";
```

```
    elsif (CLKFLT'event and CLKFLT='1') then
```

```
        A <= A_NONDELAYED;
```

```
        B <= B_NONDELAYED;
```

```
        C <= C_NONDELAYED;
```

```
    end if;
```

```
end process;
```

```
end rtl;
```

Appendix B

Phase error sign detection theorem proof

Base. At this stage we'll proof that for any initial phases of the input signals t_{0r} and t_{0d} , one of the measurement cycles provides a '0' value of the SIGN signal.

If $t_a < t_b$, the first measurement cycle provides $SIGN = 1$, and the proof is done. Otherwise, the first measurement cycle(s) give(s) $SIGN = 0$, which is an incorrect value. In this case, we want to proof the following lemma:

It exist i and j such that at least two events of the fast sequence b lie between two neighboring events of the slow sequence a . This situation illustrated with Fig. B.1 produces a change on the SIGN value, and hence, in the present context, forces SIGN to take a '1' value.

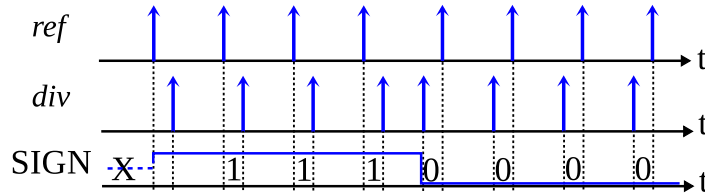


Figure B.1: Sign detection in BB circuit

Mathematically, the lemma is equivalent to a system of inequalities:

$$\begin{cases} T_r i + t_{0r} \leq T_d j + t_{0d} \\ T_d (j+1) + t_b < T_r (i+1) + t_{0r} \end{cases} \quad (\text{B.1})$$

These inequalities may be rewritten as :

$$\begin{cases} T_d j - T_r i \geq t_{0r} - t_{0d} \\ T_d j - T_r i \leq t_{0r} - t_{0d} + T_r - T_d \end{cases} \quad (\text{B.2})$$

To proof this inequality system, we consider two cases of relation between T_r and T_d : rational and irrational.

a) $T_d = m/n \cdot T_r$, $m < n$, $m, n \in \mathbb{N}$. We suppose that n and m have no common multipliers except 1.

In this case, the set of values $T_r i - T_d j$, $i, j \in N$ is a set of reel numbers starting from zero and spaced by $\Delta = T_r/n$. Hence, it exist i_1, j_1 such that :

$$T_d j - T_r i - \Delta \leq t_{0r} - t_{0d} < T_d j - T_r i \quad (\text{B.3})$$

Note that the first inequality of (B.2) is exactly the the second inequality of (B.3). Now, let us consider the first (left) inequality of (B.3):

$$T_d j - T_r i - \Delta \leq t_{0r} - t_{0d} \quad (\text{B.4})$$

We rewrite it as :

$$T_d j - T_r i \leq t_{0r} - t_{0d} + \Delta \quad (\text{B.5})$$

We note that $T_r - T_d = n\Delta - m\Delta \geq \Delta$ since $n > m$, hence from (B.5) follows :

$$T_d j - T_r i \leq t_{0r} - t_{0d} + (T_r - T_d) \quad (\text{B.6})$$

That proofs (B.2), so concluding the proof of the lemma. Hence, we proven that if the first values of the SIGN output are wrong, there will be at least one change of the SIGN value, and at least one output value represents a correct frequency sign. That concludes the base stage of the theorem proof.

b) if T_r/T_d is irrational, the set $T_r i - T_d j$, $i, j \in N$ is infinite countable set such that for any small Δ it exists i, j such that $T_r i - T_d j < \Delta$. That can be proven using the Van der Corput theorem on equidistribution modulo 1, from the number theory. Choosing $\Delta < T_r - T_d$ and repeating the last demonstration, we proof the base stage of the theorem proof for that second case.

It would be interesting to know after which number of cycles the correct sign of the frequency error is firstly detected in the worst case. The worst case is when $t_{0r} < t_{0d}$, and $t_{0d} - t_{0r} = T_d - \delta$, where δ is an infinitesimal value. In this case, the number of measurement cycles after which a correct result is always obtained is given by the formula (4.3). This formula gives also the maximal number of the quick sequence cycles necessary to let pass so to have a right frequency error sign measurement.

To prove this formula, let us consider two cases. If $T_r/T_d \geq 2$, it gives 2, which is evidently a correct result. Indeed, it correspond to the case in which one period of the slow sequence (r_i) includes several periods of the fast sequence (d_i). Evidently, since the second event of the fast sequence, the latter is leading and the detects a correct sign value. If $T_r/T_d < 2$, the fast sequence needs several cycles to compensate the initial one period delay. The fast sentence gains $T_r - T_d$ of delay by cycle, and when it comes to zero, the measured error frequency sign is correct again. It needs exactly the number of periods given by (4.3).

Induction. Let us prove that as far as one correct measurement of the frequency error is done, the next measurement is necessary correct as well.

It is the same as saying that if for some i, j $r_i - d_j > 0$, then $r_{i+1} - d_{j+1} > 0$. This validity of this statement is evident from the formula (4.2) and from the fact that $T_r > T_d$.

The theorem is proven.

Appendix C

Matlab scripts

Matlab control sequence generators

Here you can find the Matlab scripts used during the step of verification of the ADPLL and clock network. Each script returns data files, which have been included during the VHDL and Eldo simulations of the circuits.

Listing C.1: Script generates a programming sequence for the single ADPLL

```
clc
clear
tr = 0.05;
vdd = 1.1;           % vdd voltage , V
Fref = 280;         % reference frequency , MHz
Fsck = 100;        % data send rate , MHz
Ki=0.0243; % 1/4096:1/4096:4095/4096 or 2.4414e-04:2.4414e-04:0.9998
Kp=1; % 1/8:1/8:31/8 or 0.125:0.125:3.875

BBen=0; %% bang-bang mode bit , 0 - disable , 1 - enable

Kint=round(Ki*(2^12)) % 100 ;% 4095 max
Kprop=round(Kp*(2^5)) % 24; % 31 max

KW1=1; KW2=0; KW3=0; KW4=0;

%% file to write binary data
fid_vhdl=fopen('/users/cao/zianbeto/cadence/include/param_adpll.dat', 'w');

send_to_file(KW4, 2, 0, fid_vhdl);
send_to_file(KW3, 2, 0, fid_vhdl);
send_to_file(KW2, 2, 0, fid_vhdl);
send_to_file(KW1, 2, 0, fid_vhdl);
send_to_file(Kprop, 5, 0, fid_vhdl);
send_to_file(Kint, 12, 0, fid_vhdl);
send_to_file(BBen, 1, 0, fid_vhdl);

fprintf(1, 'Ding_VHDL!\n');
```

```

%% loading data from file
Z=importdata('/users/cao/zianbeto/cadence/include/param_adpll.dat');
%% file to write timing data
fid_eldo=fopen('/users/cao/zianbeto/cadence/include/param_adpll_eldo', 'w');
N = length(Z);

tref = 1000/Fref;
tsda = 1000/Fsck;

bit0 = Z(1);
fprintf(fid_eldo, 'V10_sda_0_PWL_(0_1.2g_2.4gn_1.2g\n',
    bit0*vdd, tsda-tr, bit0*vdd);
d=0;
for p = 2:N
    if p < N
        bit = Z(p);
        if bit == 0
            fprintf(fid_eldo, '+4.6gn_1.2g_4.6gn_1.2g\n', (p-1)*tsda,
                bit*vdd, p*tsda-tr, bit*vdd);
        elseif bit == 1
            fprintf(fid_eldo, '+4.6gn_1.2g_4.6gn_1.2g\n', (p-1)*tsda,
                bit*vdd, p*tsda-tr, bit*vdd);
        end
    elseif p == N
        fprintf(fid_eldo, ')\n');
        fprintf(fid_eldo, 'V11_reset_0_PWL_(0_1.1_1n_1.1_1.03n_0_3n_0_3.03n
_1.1_4.6gn_1.1_4.6gn_0_4.6gn_0_4.6gn_1.1)\n', 2+(p+d)*tsda,
            2+(p+d)*tsda+tr, 2+(p+d)*tsda+tr+1, 2+(p+d)*tsda+tr+1+tr);
        fprintf(fid_eldo, 'V12_adpll_en_0_PWL_(0_0_4.6gn_0_4.6gn_0_4.6gn
_0_4.6gn_1.1)\n', (p+d)*tsda, (p+d)*tsda+tr, (p+d)*tsda+tr+1,
            (p+d)*tsda+tr+1+tr);
        end
    end
end

fprintf(fid_eldo, 'V13_sck_0_PULSE_(0_1.2g_0_0_2.4gn_2.4gn_3.4gn_3.4gn_
)\n', vdd, tr, tr, tsda/2, tsda);
fprintf(fid_eldo, 'V14_ref_0_PULSE_(0_1.2g_0_0_2.4gn_2.4gn_3.4gn_3.4gn_
', vdd, tr, tr, tref/2, tref);
fprintf(1, 'Ding_ELDO!\n');

```

Listing C.2: Script generates a programming sequence for the clock network

```

clc
clear

%Ki=0.0028; % 1/4096:1/4096:4095/4096 or 2.4414e-04:2.4414e-04:0.9998
Ki=0.0028;
%Kp=2.2; % 1/8:1/8:31/8 or 0.125:0.125:3.875
Kp=1;
BBen=0; %% bang-bang mode bit, 0 - disable, 1 - enable

```

```

k=1; %% network mode, 0 – unidirectional, 1 – bidirectional
l=0;

Ki_int=round(Ki*(2^12));
Kp_int=round(Kp*(2^5));

Kprop_num_flag=Kp_int/2

if Kprop_num_flag < 2^5
    KW_coef =1;
elseif ( Kprop_num_flag >= 2^5 & Kprop_num_flag < (2^6))
    KW_coef =2;
elseif ( Kprop_num_flag >= 2^6 & Kprop_num_flag <= (2^7))
    KW_coef =4;
end
KW_coef
Kprop_num_4in=round(Kp_int/4/KW_coef)
Kprop_num_3in=round(Kp_int/3/KW_coef)
Kprop_num_2in=round(Kp_int/2/KW_coef)

Kint_num_4in=round(Ki_int/4/KW_coef)
Kint_num_3in=round(Ki_int/3/KW_coef)
Kint_num_2in=round(Ki_int/2/KW_coef)

KW1(1)=KW_coef*k; KW2(1)=KW_coef*1; KW3(1)=KW_coef*1; KW4(1)=0;
Kint_num(1)=Kint_num_3in; Kprop_num(1)=Kprop_num_3in;
KW1(2)=KW_coef*k; KW2(2)=KW_coef*1; KW3(2)=KW_coef*1; KW4(2)=0;
Kint_num(2)=Kint_num_3in; Kprop_num(2)=Kprop_num_3in;
KW1(3)=KW_coef*k; KW2(3)=KW_coef*1; KW3(3)=KW_coef*1; KW4(3)=0;
Kint_num(3)=Kint_num_3in; Kprop_num(3)=Kprop_num_3in;
KW1(4)=KW_coef*k; KW2(4)=0; KW3(4)=KW_coef*1; KW4(3)=0;
Kint_num(4)=Kint_num_2in; Kprop_num(4)=Kprop_num_2in;

KW1(5)=0; KW2(5)=KW_coef*1; KW3(5)=KW_coef*1; KW4(5)=KW_coef*k;
Kint_num(5)=Kint_num_3in; Kprop_num(5)=Kprop_num_3in;
KW1(6)=KW_coef*k; KW2(6)=KW_coef*1; KW3(6)=KW_coef*1; KW4(6)=KW_coef*k;
Kint_num(6)=Kint_num_4in; Kprop_num(6)=Kprop_num_4in;
KW1(7)=KW_coef*k; KW2(7)=KW_coef*1; KW3(7)=KW_coef*1; KW4(7)=KW_coef*k;
Kint_num(7)=Kint_num_4in; Kprop_num(7)=Kprop_num_4in;
KW1(8)=KW_coef*k; KW2(8)=0; KW3(8)=KW_coef*1; KW4(8)=KW_coef*k;
Kint_num(8)=Kint_num_3in; Kprop_num(8)=Kprop_num_3in;

KW1(9 )=0; KW2(9 )=KW_coef*1; KW3(9 )=KW_coef*1; KW4(9 )=KW_coef*k;
Kint_num(9 )=Kint_num_3in; Kprop_num(9 )=Kprop_num_3in;
KW1(10)=KW_coef*k; KW2(10)=KW_coef*1; KW3(10)=KW_coef*1; KW4(10)=KW_coef*k;
Kint_num(10)=Kint_num_4in; Kprop_num(10)=Kprop_num_4in;
KW1(11)=KW_coef*k; KW2(11)=KW_coef*1; KW3(11)=KW_coef*1; KW4(11)=KW_coef*k;
Kint_num(11)=Kint_num_4in; Kprop_num(11)=Kprop_num_4in;
KW1(12)=KW_coef*k; KW2(12)=0; KW3(12)=KW_coef*1; KW4(12)=KW_coef*k;
Kint_num(12)=Kint_num_3in; Kprop_num(12)=Kprop_num_3in;

```

```

KW1(13)=0; KW2(13)=KW_coef*1; KW3(13)=0; KW4(13)=KW_coef*k;
Kint_num(13)=Kint_num_2in; Kprop_num(13)=Kprop_num_2in;
KW1(14)=KW_coef*k; KW2(14)=KW_coef*1; KW3(14)=0; KW4(14)=KW_coef*k;
Kint_num(14)=Kint_num_3in; Kprop_num(14)=Kprop_num_3in;
KW1(15)=KW_coef*k; KW2(15)=KW_coef*1; KW3(15)=0; KW4(15)=KW_coef*k;
Kint_num(15)=Kint_num_3in; Kprop_num(15)=Kprop_num_3in;
KW1(16)=KW_coef*k; KW2(16)=0; KW3(16)=0; KW4(16)=KW_coef*k;
Kint_num(16)=Kint_num_2in; Kprop_num(16)=Kprop_num_2in;

Kp_1=integer_to_binary(Kprop_num(1),5,0);
Kp_4=integer_to_binary(Kprop_num(4),5,0);
Kp_13=integer_to_binary(Kprop_num(13),5,0);
Kp_16=integer_to_binary(Kprop_num(16),5,0);
Kp_out(1) = Kp_1(2); Kp_out(2) = Kp_4(2);
Kp_out(3) = Kp_13(2); Kp_out(4) = Kp_16(2);
Kp_out = Kp_out

%% file to write binary data
fid_vhdl=fopen('/users/cao/zianbeto/cadence/include/param.dat', 'w');

order_of_loading=[13 14 15 16 41 40 39 38 37 36 35 9 10 11 12 34 33
 32 31 30 29 28 5 6 7 8 27 26 25 24 23 22 21 1 2 3 4 20 19 18 17];
M = length(order_of_loading);
for count=1:M
i=order_of_loading(count);
    if i <= 16
        send_to_file(KW4(i), 2, 0, fid_vhdl);
        send_to_file(KW3(i), 2, 0, fid_vhdl);
        send_to_file(KW2(i), 2, 0, fid_vhdl);
        send_to_file(KW1(i), 2, 0, fid_vhdl);
        send_to_file(Kprop_num(i), 5, 0, fid_vhdl);
        send_to_file(Kint_num(i), 12, 0, fid_vhdl);
    elseif i > 16
        send_to_file(BBen, 1, 0, fid_vhdl);
    end
end
fprintf(1, 'Ding_VHDL!\n');

% loading data from file
Z=importdata('/users/cao/zianbeto/cadence/include/param.dat');
%% file to write timing data
fid_eldo=fopen('/users/cao/zianbeto/cadence/include/param_eldo', 'w');
N = length(Z);

Fsck = 100;           % data send rate , MHz
tsda = 1000/Fsck;
tr = 0.03;
vdd = 1.1;           % vdd voltage , V
bit0 = Z(1);

```

```

fprintf(fid_eldo , 'V10_sda_0_PWL_(0_0_1.1g_2.4gn_1.1g\n' , bit0*vdd ,
    tsda-tr , bit0*vdd);

for p = 2:N
    if p < N
        bit = Z(p);
        if bit == 0
            fprintf(fid_eldo , '+%4.6gn_1.2g_4.6gn_1.2g\n' , (p-1)*tsda ,
                bit*vdd , p*tsda-tr , bit*vdd);
        elseif bit == 1
            fprintf(fid_eldo , '+%4.6gn_1.2g_4.6gn_1.2g\n' , (p-1)*tsda ,
                bit*vdd , p*tsda-tr , bit*vdd);
        end
    elseif p == N
        fprintf(fid_eldo , ')\n');
        fprintf(fid_eldo , 'V11_reset_0_PWL_(0_1.1_1n_1.1_1.03n_0_3n_0_3.03n
            _1.1_4.6gn_1.1_4.6gn_0_4.6gn_0_4.6gn_1.1)\n' , 2+p*tsda , 2+p*tsda+tr ,
            2+p*tsda+tr+1 , 2+p*tsda+tr+1+tr);
        fprintf(fid_eldo , 'V12_ntwrk_en_0_PWL_(0_0_4.6gn_0_4.6gn_0_4.6gn
            _0_4.6gn_1.1)\n' , p*tsda , p*tsda+tr , p*tsda+tr+1 , p*tsda+tr+1+tr);
        end
    end

fprintf(fid_eldo , 'V13_sck_0_PULSE_(0_0_1.2g_0_0_2.4gn_2.4gn_3.4gn_3.4gn
    _)\n' , vdd , tr , tr , tsda/2 , tsda);
fprintf(fid_eldo , 'V14_vdda_gnda_PWL_(0_0_4.6gn_0_4.6gn_0_4.6gn_0_4.6gn
    _1.1)\n' , p*tsda , p*tsda+tr , p*tsda+tr+1 , p*tsda+tr+1+tr);
fprintf(1 , 'Ding_ELDO!\n');

```

Listing C.3: Integer to binary subscript

```

function y=integer_to_binary(x, n, s)
x_fi=fi(x,s,n,0);
for i=1:n
    y(i)=int16(bitget(x_fi ,n-i+1));
end
end

```

Listing C.4: Script writes the sequence to file

```

function y=send_to_file(value , n , s , fid_vhdl)
value_bin=integer_to_binary(value ,n ,s);
for i=1:n
fprintf(fid_vhdl , '%i\n' , value_bin(n-i+1));
end
y=1;
end

```


Matlab test circuit control script

Here are presented the M-files with set of instructions to command the test board. The script creates and opens the virtual COM port over the USB interface. The test sequences submitted to ATMEGA μ -controller and decoded to the parallel 10-bit code. Successful submission followed by the procedure of port closing and deleting.

Listing C.5: Control script transfers the data via a virtual COM port to the Arduino NANO board

```
create_and_open_vcm;
status = get(USB, 'Status')

for k=0:50:1000
    k
    fprintf(USB, '%i', k)
    %fprintf(USB, '%i', 50)
    %pause(0.02);
    %fprintf(USB, '%i', 60)
    pause(4.00);
end

%close_and_delete_vcm;
```

Listing C.6: Script opens a communication port

```
clc
clear
BaudRate=9600;
USB = serial('COM3');
set(USB, 'BaudRate', BaudRate, 'StopBits', 1);
%set(USB, 'Terminator', 'LF', 'Parity', 'none');
%set(USB, 'FlowControl', 'none');
if isempty(USB)
    USB = serial(Port, 'baudrate', BaudRate);
else
    fclose(USB);
    USB = USB(1);
end
fopen(USB);
```

Listing C.7: Script closes a communication port

```
fclose(USB);
delete(USB);
clear USB;
clc
clear
```

Matlab post-processing script

The M-file with postprocessing procedures are presented in this section. The captured by the digital oscilloscope waveforms are converted to the timing data array. The following results could be obtained:

- Time Interval Error (TIE), *ps*
- Cycle-to-cycle period deviation, *ps*
- Period variation histogram
- Ideal period of oscillation, *ns*
- Ideal frequency of oscillation, *MHz*
- Jitter rms value, *ps*
- Jitter peak-to-peak value, *ps*
- Spectrum of the signal

Listing C.8: Post-processing of the captured by oscilloscope waveforms

```
%clc;
%clear all;
%close all;
%warning off all
format long e
th = 0.5;           % level of threshold 0...1
ts = 5e-11;        % digital oscilloscope sampling period
import_wavform('C:/Users/LeCroyUser/Desktop/C1Trace-ofdm00002.dat',ts,th);
%% Loading data
load('Treal.mat','treal','U','Nt','Tideal');
%% Absolute period deviation
TIE = zeros(Nt-2,1);
for i = 1:1:Nt-2
    TIE(i) = (treal(i+1) - treal(i))-Tideal;
end
figure(1), plot(TIE,'LineWidth',2.0)
title('Absolute_period_variation');
grid on
%% Cycle-to-cycle deviation
TIEcc = zeros(Nt-2,1);
for i = 1:1:Nt-2
    TIEcc(i) = (treal(i+2) - treal(i+1))-(treal(i+1) - treal(i));
end
figure(2), plot(TIEcc,'LineWidth',2.0)
title('Cycle-to-cycle_period_variation');
grid on
%% Period variation histogram
figure(3), hist(TIE,100)
title('Period_variation_histogram');
grid on
%% Ideal period and frequency
fprintf(1, 'Ideal_period:_%2.4f_ns\n', Tideal*1e9)
```

```

fprintf(1, 'Ideal_frequency:_%3.4f_MHz\n', 1e-6/Tideal)
%% RMS Jitter
% The RMS value of dT for infinite number of cycles is called as
% "RMS jitter"
Tjrms = std(TIE);
fprintf(1, 'Jitter_rms:_%1.4f_ps\n', Tjrms*1e12)
%% Peak-to-peak Jitter
% The difference between maximum and minimum value of dT is called as
% "peak-to-peak jitter".
%alpha = 14.1;
alpha = 6.581;
BER = 0.5*erfc(alpha/(8^0.5));
Tjpp = Tjrms * alpha;
% Tjpp = max(dT)-min(dT); % standard oscilloscope method
fprintf(1, 'Jitter_pp:_%1.4f_ps\n', Tjpp*1e12)
%% FFT of the input signal
NFFT = 2^nextpow2(Nt); %%% Next power of 2 from length of y
Y = fft(U,NFFT)/Nt;
Fs = 1/ts; %%% sampling frequency
f = Fs/2*linspace(0,1,NFFT/2+1);
figure(4), semilogy(f,2*abs(Y(1:NFFT/2+1)), 'LineWidth',1.5)
title('Spectrum_of_signal');
grid on
zoom xon
% zoom(zz)
xlabel('Frequency_(Hz)')
ylabel('|Y(f)|')
%figure(5), plot(t,U)

```

Appendix D

DCO test chip characterization flow

This appendix presents the details of the lab bring-up and measurement procedures. In order to parametrize the fabricated test circuit of DCO the testing platform was developed (Fig. D.2). It consists of PCB and control module. The schematic of the developed PCB is depicted in a Fig. D.1. It consists of power supply unit, level shifters, USB interface board, DCO chip.

Supply unit

The circuit is supplied by the +5V taken from USB interface of the personal computer. It is adjusted and filtered by:

- LDO regulators *IC5* and *IC6* with fixed output voltage: 1.1V, 1.8V
- LDO regulator *IC4* with variable output voltage for *v_{dda}*: 0.99...1.21V
- multiple tantalum and ceramic caps to prevent supply ripple and coupling

The adjustable supply is used for the core of the oscillator (rail *v_{dda}* in DCO). The fixed 1.1V is for digital part of the circuit and 1.8V for the IO pads.

Level shifters

Two IO/level shifting drivers *IC2* and *IC3* used to interface the control board high 5V signals with low 1.8V inputs of DCO. 10 LEDs are used to indicate and verify the control code.

USB interface board

The used USB interface board is *ARDUINO Nano* with ATMEGA16 μ controller and *FTDI232* interface integrated circuit on-board. It receives the information from the virtual COM port of the computer, store the package, decode it from HEX to binary format and outputs it on pins *D3* – *D12*. Complete documentation and support of this platform are available on official project [cite](#).

DCO chip

Designed oscillator *IC1* is in MLP24 package. It is surrounded by the decoupling capacitors and interface passive components. The output high frequency signals are connected to the oscilloscope by two SMA connectors.

Figure D.1: Schematic of the developed PCB

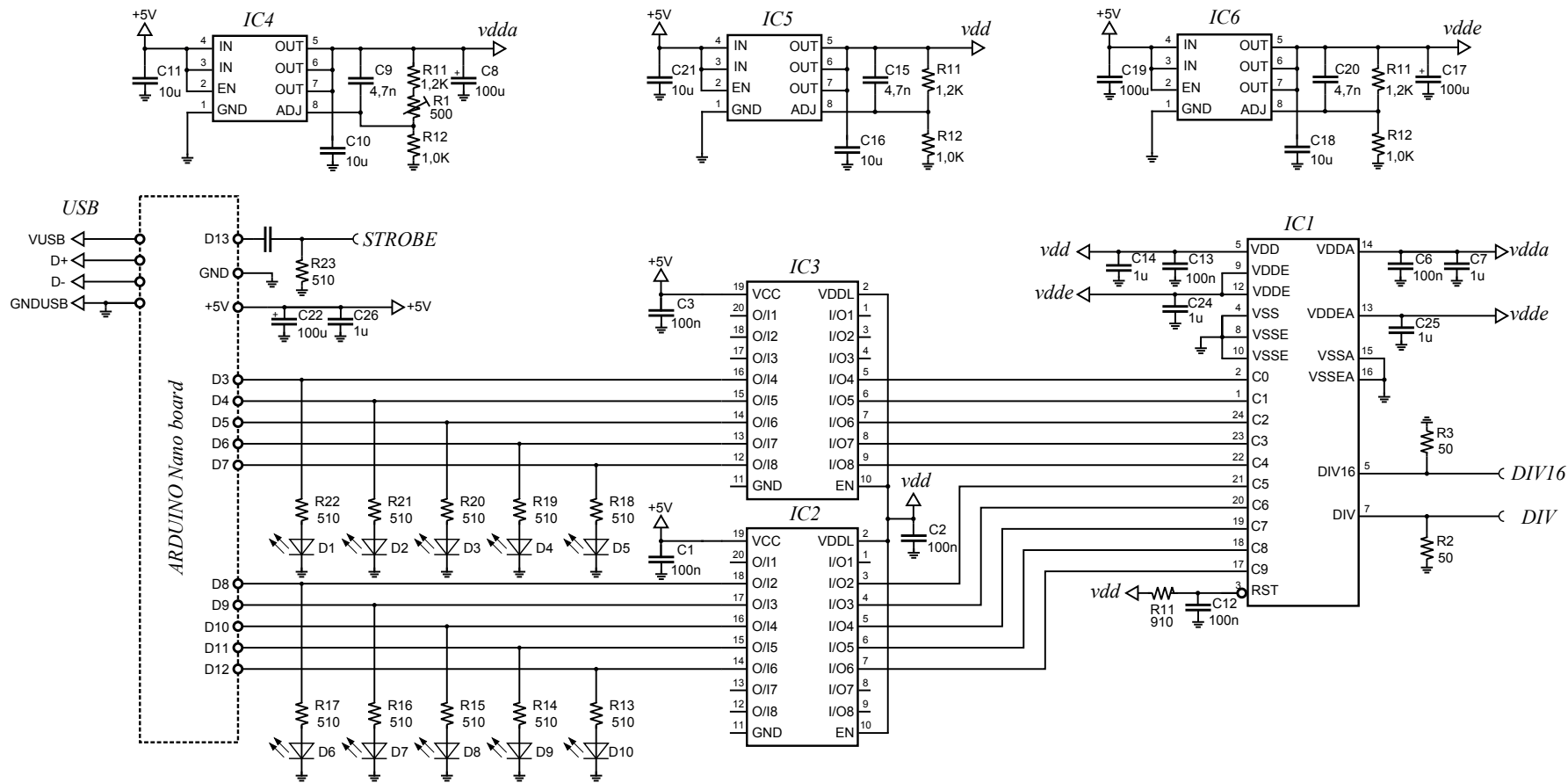




Figure D.2: Photo of the developed testing platform

Appendix E

FPGA prototyping of the clocking network

Environment and measurement set

The Mentor Graphics AdvaceMS mixed simulator is used to perform behavioral mixed simulations of the developed code before porting it to Altera Quartus II environment, which has been extensively used in the design of the clocking network prototype.

The clocking network has been implemented onto EP2C70F672C6 chip. The clocking network core has been designed keeping in mind the spatial location criterion. This means that nodes of the network were spatially distributed over the FPGA area so to form an equivalent to the specified 2D mesh network.

The reference clock has been generated by external **FLUKE PM 5136 Synthesized Function Generator**. The output signals were captured and processed by **LeCroy Waverunner** oscilloscope with **Mixed Signal Oscilloscope Option**.

The FPGA chip is installed on *Cyclone II DSP Development Board*. Fig. E.1 depicts the functional diagram of this platform and Fig. E.2 top view of the board with description.

The measurement set is depicted in a Fig. E.3.

VHDL models

This section introduces the complete synthesizable VHDL models of the proposed DCO and TDC for FPGA implementation.

Listing E.1: VHDL model of the synthesizable DCO

```
library IEEE;  
use IEEE.STD_LOGIC_1164.all;  
use IEEE.MATH_REAL.all;  
use IEEE.NUMERIC_STD.all;  
use IEEE.STD_LOGIC_ARITH.all;  
use IEEE.STD_LOGIC_UNSIGNED.all;
```

```
ENTITY nco_fpga IS
```



```

PORT (
    CLK_fpga: in STD.LOGIC;
    FLT_OUT: in STD.LOGIC_VECTOR(9 downto 0);
    CLK_OUT: out std_logic
);
END;

ARCHITECTURE rtl OF nco_fpga IS

    ----- generating of the constants and signals -----

    constant N: INTEGER:=1789;    — max value of counter
    signal M: INTEGER:=1155;    — medium value of counter
    signal sum:INTEGER range 0 to N:=512 ;
    signal FLT_OUT_integer : integer range 0 to 1023;

BEGIN

    FLT_OUT_integer <= conv_integer(FLT_OUT);
    — The period is equal to (N-FLT_OUT+1)*T-CLK_fpga
    process(CLK_fpga,FLT_OUT) is
        begin
            if (CLK_fpga' event and CLK_fpga='1') then

                if (sum < M) then
                    sum <= sum + 1;
                    CLK_OUT<='1';
                elsif (sum < N) then
                    sum <= sum + 1;
                    CLK_OUT<='0';
                else
                    sum <= FLT_OUT_integer;
                    M <= (FLT_OUT_integer+N)/2;
                    CLK_OUT<='1';
                end if;

            end if;
        end process;
END rtl;

```

Listing E.2: VHDL model of the synthesizable TDC

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.MATH_REAL.all;    use IEEE.NUMERIC_STD.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
use IEEE.numeric_std.all;

entity tdc_fpga is
    port (
        ERR,SIGNE,CLK_fpga,RESET: in STD.LOGIC;
        ERREUR, NERREUR: out std_logic_vector (4 downto 0)
    );

```

```

    );
end tdc_fpga;

architecture rtl of tdc_fpga is
    ----- generating of the constants and signals -----
    signal cpt_int: INTEGER range 0 to 15;
    signal CODER_OUT: integer range -16 to 15 ;
    signal count_reset: integer range 0 to 7;
begin
process (RESET, CLK_fpga, ERR, cpt_int) is
begin
    if (RESET = '0') then
        cpt_int <= 1;
        count_reset <= 0;
    else
        if (CLK_fpga'event and CLK_fpga='1') then
            if (ERR = '0') then
                if (count_reset > 2) then
                    cpt_int <= 1;
                    count_reset <= 0;
                else
                    cpt_int <= cpt_int;
                    count_reset <= count_reset + 1;
                end if;
            else
                count_reset <= 0;
                if (cpt_int < 15) then
                    cpt_int <= cpt_int + 1;
                else cpt_int <= cpt_int;
                end if;
            end if;
        end if;
    end if;
end process;

process (ERR, RESET, SIGNE)
begin
    if (RESET='0') then
        CODER_OUT <= 1;
    elsif (ERR='0' and ERR'event) then
        if (SIGNE='0') then
            CODER_OUT <= -1 * cpt_int;
        else
            CODER_OUT <= cpt_int;
        end if;
    end if;
end process;

-----
ERREUR <= std_logic_vector (to_signed (CODER_OUT, ERREUR'length));

```

```
NERREUR<=std_logic_vector(to_signed(-1*CODER_OUT, NERREUR'length));  
end rtl;
```

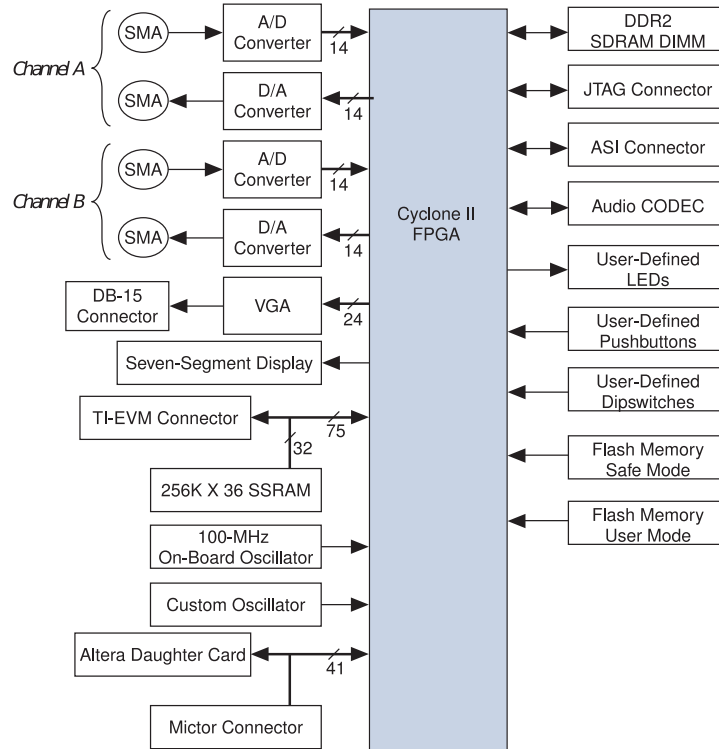


Figure E.1: Functional diagram of Cyclone II DSP Development Board

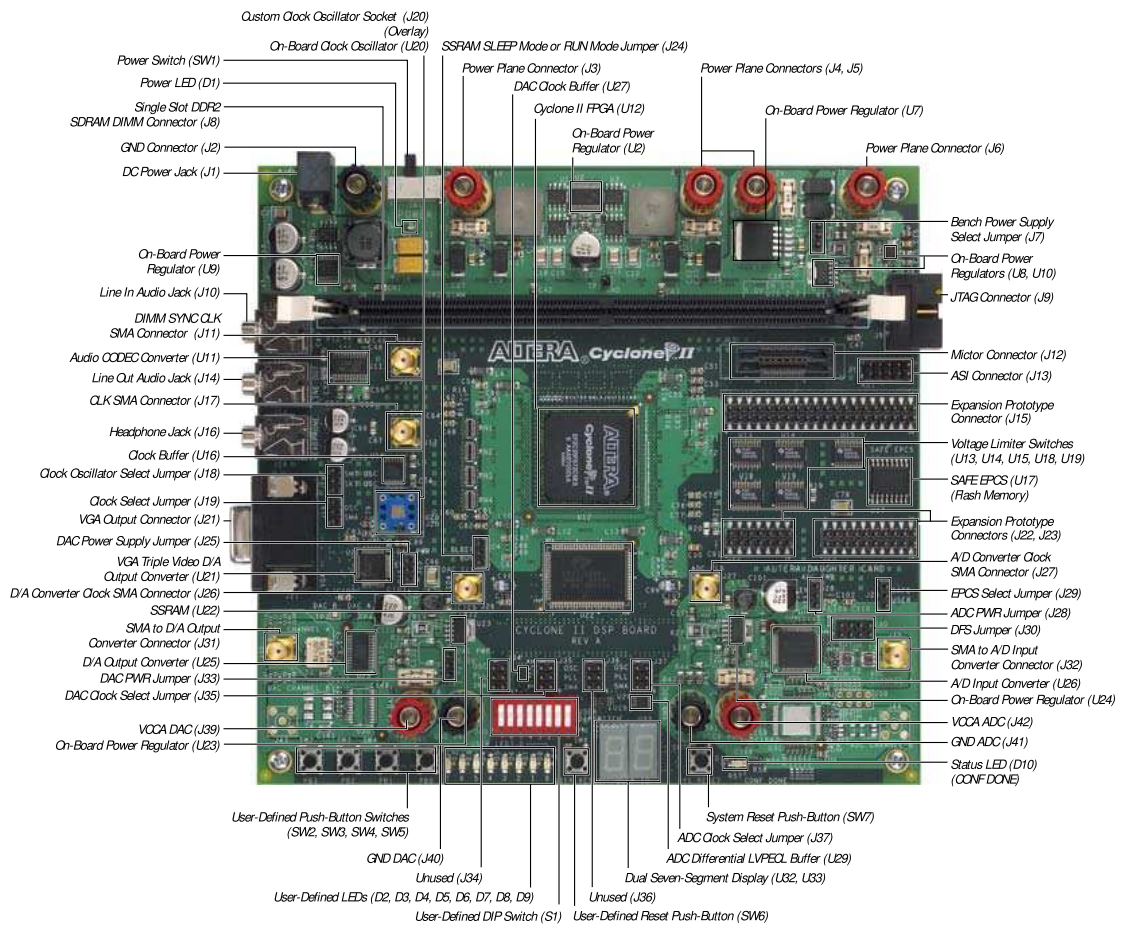


Figure E.2: Top view of Cyclone II DSP Development Board

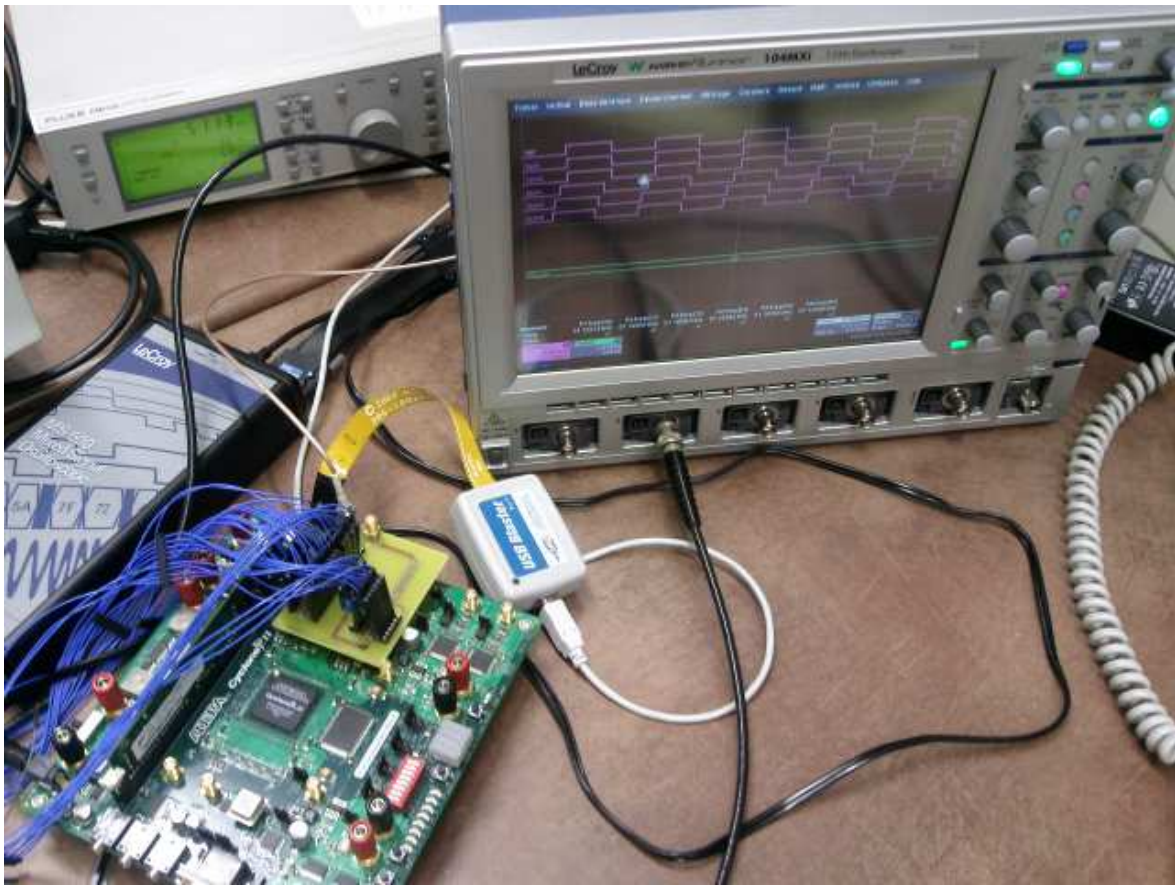


Figure E.3: **FPGA prototyping platform:** Cyclone II EP2C70 DSP Development Board and LeCroy Waverunner Digital Sampling Oscilloscope

Appendix F

VLSI implementation of the network

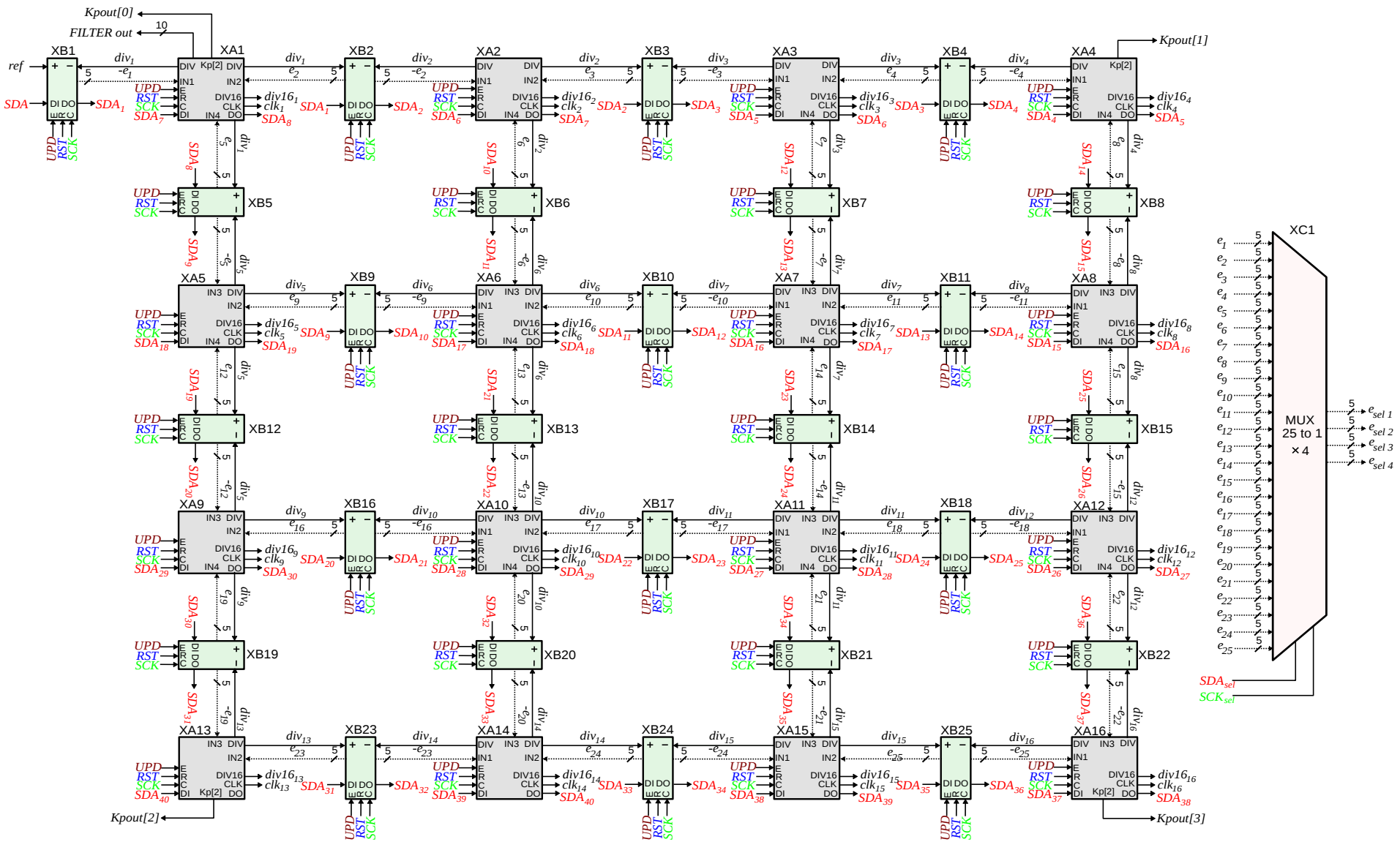


Figure F.1: Complete block diagram of the clock network and DFT blocks

Appendix G

Clocking network test chip characterization flow

This appendix presents the details of the lab bring-up and measurement procedures. In order to parametrize the fabricated test circuit of DCO the testing platform was developed (Fig. G.2). It consists of PCB and control module. The schematic of the developed PCB is depicted in a Fig. G.1. It consists of power supply unit, level shifters, USB interface board, test chip.

Supply unit

The circuit is supplied by the external stabilized low noise source $+5.5V$. In addition, it is adjusted and filtered by:

- LDO regulators $IC4$ and $IC5$ with fixed output voltage: $1.1V$, $1.8V$
- LDO regulator $IC3$ with variable output voltage for v_{dda} : $1.0 \dots 1.3V$
- multiple tantalum and ceramic caps to prevent supply ripple and coupling

The adjustable supply is used for the cores of the oscillators. The fixed $1.1V$ is for digital part of the circuit and $1.8V$ for the I/O pads.

Level shifters

Two IO/level shifting drivers $IC2$ and $IC3$ used to interface the control board high $5V$ signals with low $1.8V$ inputs of DCO. 10 LEDs are used to indicate and verify the control code.

USB interface board

The used USB interface board is *ARDUINO Nano* with ATMEGA16 μ controller and *FTDI232* interface integrated circuit on-board. It receives the information from the virtual COM port of the computer, store the package, and translate it to serial interface (signals *SDA*, *SCK*, *SDE* and *RST*). The serial control of DFT multiplexer is done in a same way but using signals *MSCK* and *MSDA*.

Network test chip

Designed clocking network *IC1* is in CQFP120 package. It is surrounded by the decoupling capacitors and interface passive components. The output clock signals are connected to the oscilloscope by 16 coaxial SMA connectors. The reference clock and PFD clock come on PCB via two coaxial SMA connectors. The multibit signals took out via multiple connectors.

Measurement tools

The reference clock has been generated by **ANRITSU Signal Quality Analyzer MP1800A**. The output clock signals were captured and processed by **LeCroy WaveRunner 625Zi** oscilloscope (2.5 GHz, 20 GS/s, 4ch, 16 Mpts/Ch DSO). For the capturing of multibit signals we have used **Agilent 16902B Modular Logic Analysis System**. The power supply unit is **Agilent E3630A 35 W Triple Output power supply**. Voltage supply measurements have been performed by **Metrix MX53C** digital multimeter.

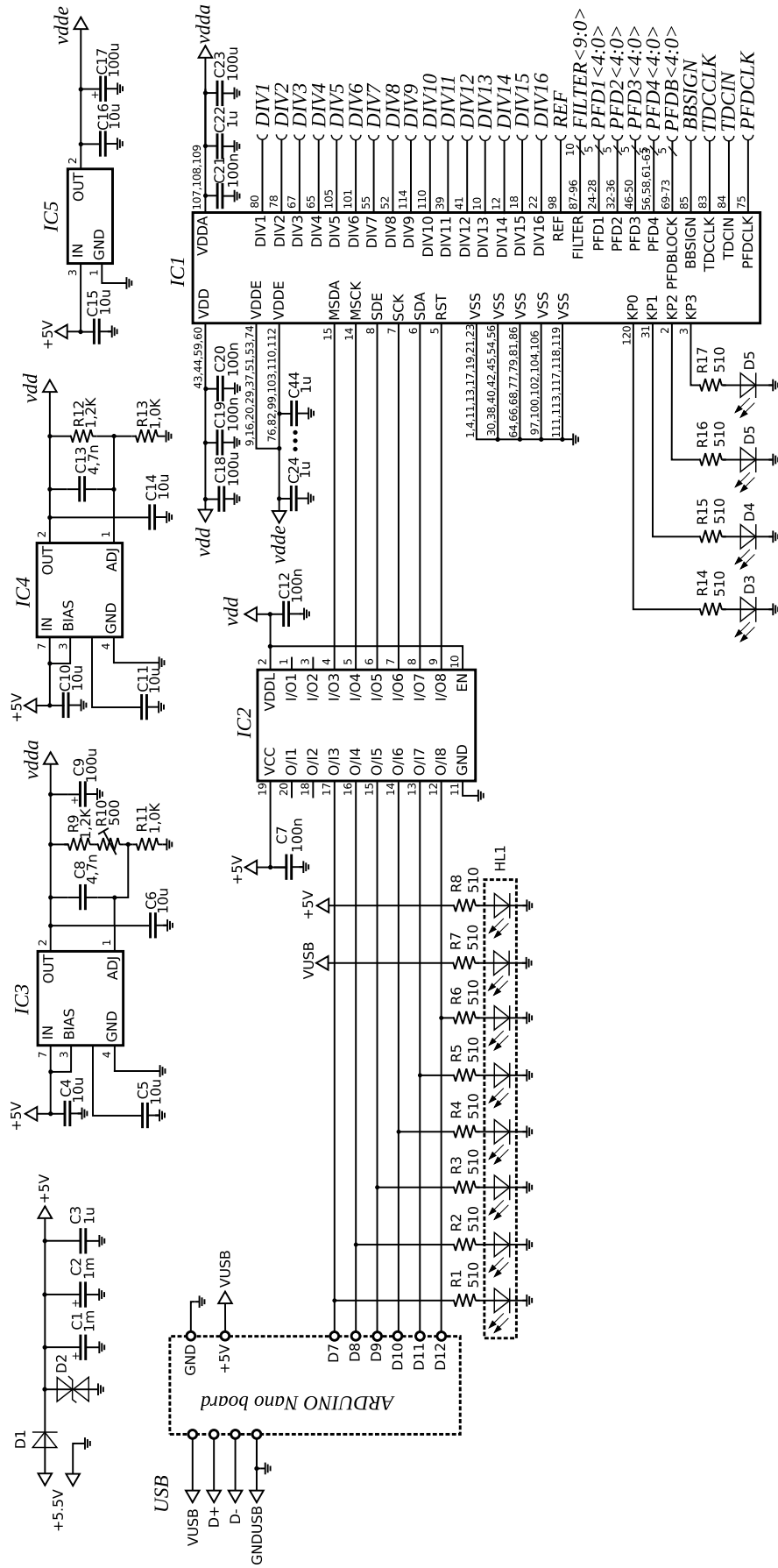


Figure G.1: Schematic of the developed PCB for network chip testing

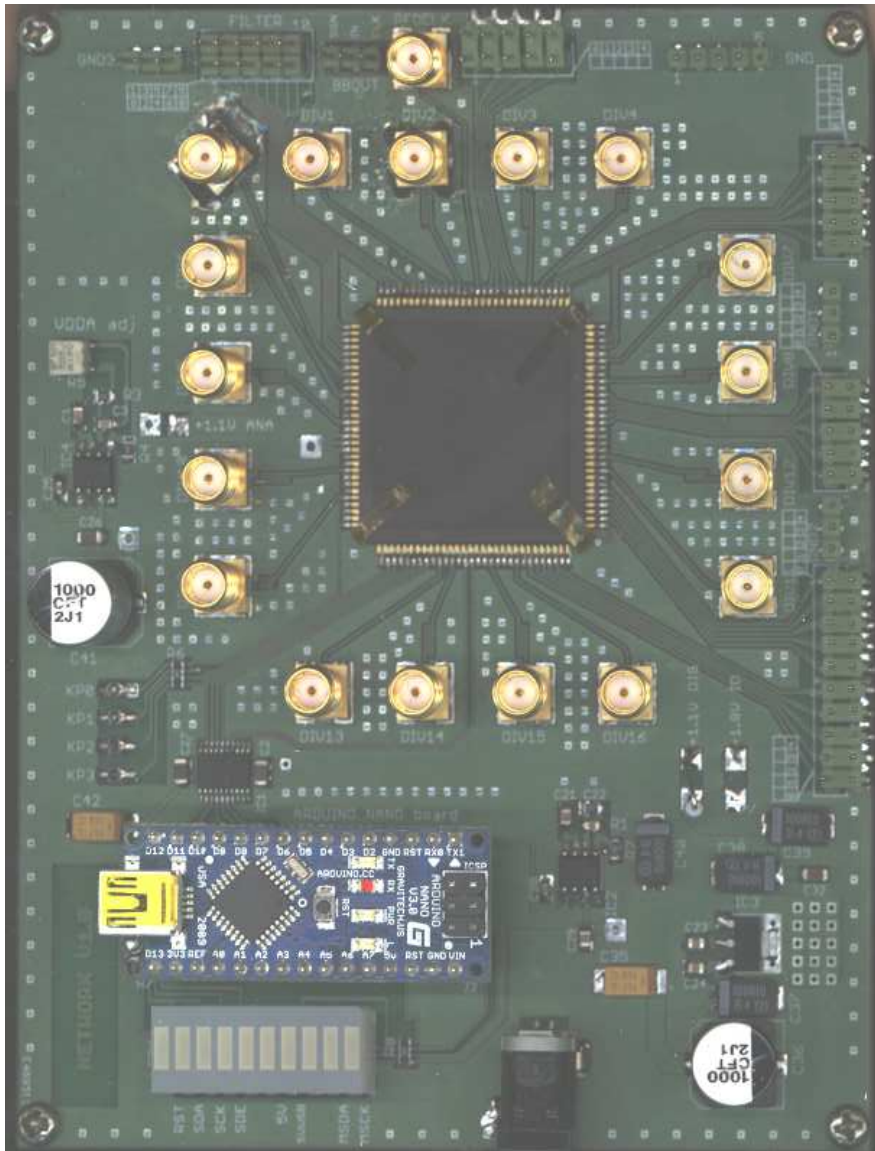


Figure G.2: Photo of the developed clocking network testing platform

List of Abbreviations and Symbols

Abbreviation	Description	Definition
SoC	System-on-Chip	page 1
MPSoC	Multi Processor System-on-Chip	page 2
SCA	Synchronous Clocking Area	page 3
GALS	Globally Asynchronous Locally Synchronous	page 1
GSLs	Globally Synchronous Locally Synchronous	page 3
PC	Phase Comparator	page 8
VDL	Variable Delay Line	page 8
PVT	Process, supply Voltage and Temperature	page 8
VCO	Voltage Controlled Oscillator	page 10
PLL	Phase-Locked Loop	page 11
ADPLL	All-digital phase-locked loop	page 18
DPC	Digital Phase Comparator	page 18
DLF	Digital Loop Filter	page 18
DAC	Digital-to-Analog Converter	page 18
DCO	Digitally-Controlled Oscillator	page 18
CAD	Computer-Aided Design	page 19
PI	Proportional-Integral filter	page 20
DFT	Design-For-Test	page 19
CTI	Coarse-Tuning Inverter	page 52
CTIA	Coarse-Tuning Inverter Additional	page 53
FTI	Fine-Tuning Inverter	page 53
B2T	Binary-to-Thermometer	page 53
LUT	Look-Up Table	page 58
DNL	Differential Nonlinearity	page 78
BB	Bang-Bang	page 81
PWM	Pulse Width Modulation	page 82
TDC	Time-to-Digital Converter	page 83
PFD	Phase-Frequency Detector	page 83
ADC	Analog-to-Digital Converter	page 85
CLA	Carry Look-Ahead adder	page 100
SPI	Serial Programming Interface	page 105
DDFS	Direct Digital Frequency Synthesis	page 112
NoC	Network-on-Chip	page 144

Symbol	Description	Definition
ϕ	Phase of the periodic signal	page 16
$\Delta\phi, e_{a,b}$	Phase error	page 16
F_{osc}	Oscillation frequency	page 45
F_0	Minimal oscillation frequency of DCO	page 48
K	DCO control code	page 47
ΔF	Oscillator tuning step	page 48
F_{ref}	Reference frequency	page 24
α	Proportional coefficient of PI filter	page 21
β	Integral coefficient of PI filter	page 21
Δ_{PFD}	Digital phase/frequency detector resolution	page 21
$e_{ri}[n]$	Quantified phase error	page 30
τ_d	Gate/stage delay	page 42
C_{int}	Interconnection capacitance	page 42
C_l	Load capacitance	page 42
$I_{s,p}/I_{s,n}$	Source current of P and N MOS transistors	page 42
C_{cti}	Output capacitance of CTI cell	page 45
C_m	Output capacitance of main inverter	page 45
I_{cti}	Current contributed by CTI inverter	page 45
I_m	Current contributed by main inverter	page 45
ΔF_{fine}	Fine frequency tuning step	page 57
ΔF_{coarse}	Coarse frequency tuning step	page 57
V_{th}	CMOS transistor threshold voltage	page 92
τ_{TDC}	TDC resolution	page 94
ΔT	Timing error	page 94
K_w	Filter input weighting coefficients	page 100
$T_{DCO, fpga}$	DCO clock period in FPGA	page 112
$F_{n, vlsi}$	Nominal divided frequency of ASIC DCO	page 113
$F_{n, fpga}$	Nominal divided frequency of FPGA DCO	page 114
$\Delta F_{n, fpga}$	Frequency tuning step of FPGA DCO at nominal frequency	page 113
τ_{fpga}	TDC resolution in FPGA	page 115

Bibliography

- [1] A. Abdelhadi, R. Ginosar, A. Kolodny, and E.G. Friedman. Timing-driven variation-aware nonuniform clock mesh synthesis. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*, pages 15–20, 2010. [cited at p. 7]
- [2] A.A. Abidi. Phase noise and jitter in CMOS ring oscillators. *Solid-State Circuits, IEEE Journal of*, 41(8):1803–1816, 2006. [cited at p. 44]
- [3] J.M. Akre, J. Juillard, D. Galayko, and E. Colinet. Synchronization Analysis of Networks of Self-Sampled All-Digital Phase-Locked Loops. 59(4):708–720, 2012. [cited at p. 19, 21, 29, 30, 100, 109]
- [4] J.M. Akre, J. Juillard, M. Javidan, E. Zianbetov, D. Galayko, A. Korniienko, and E. Colinet. A Design Approach for Networks of Self-Sampled All-Digital Phase-Locked Loops. In *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, pages 725–728, 2011. [cited at p. 29, 30, 100, 109]
- [5] F. Anceau. Une technique de réduction de la puissance dissipée par l’horlogerie des circuits complexes rapides Zones isochrones. *Evolution*. [cited at p. 2, 11]
- [6] F. Anceau and Y. Bonnassieux. *Conception des circuits VLSI: du composant au systeme*. 2007. [cited at p. 63, 70, 72]
- [7] C.J. Anderson, J.G. Petrovick, J.M. Keaty, J. Warnock, G. Nussbaum, J.M. Tendier, C. Carter, S. Chu, J. Clabes, and J. DiLullo. Physical design of a fourth-generation POWER GHz microprocessor. In *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International*, volume 13, pages 232–233. IEEE, 2001. [cited at p. 6]
- [8] E. Beigne, F. Clermidy, S. Miermont, Y. Thonnart, A. Valentain, and P. Vivet. A localized power control mixing hopping and super cut-off techniques within a GALS NoC. ... *Circuit Design and ...*, pages 0–4, 2008. [cited at p. 24]
- [9] A. Bhatnagar and D. Miller. Optical Interconnection and Clocking for Electronic Chips. In *8th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2004)*, number Sci, pages 1–5, 2004. [cited at p. 8]
- [10] M. Cabanas-Holmen, E. Cannon, A. Kleinosowski, J. Ballast, J. Killens, and J. Socha. Clock and Reset Transients in a 90 nm RHBD Single-Core Tilera Processor. *IEEE Transactions on Nuclear Science*, 56(6):3505–3510, December 2009. [cited at p. 6]

- [11] K. Cadien, M. Reshotko, B. Block, A. Bowen, D. Kencke, and P. Davids. Challenges for on-chip optical interconnects. *Proceedings of SPIE*, 5730:133–143, 2005. [cited at p. 8, 9]
- [12] C. Cao. A 50-GHz Phase-Locked Loop in 0.13- μ m CMOS. *Solid-State Circuits, IEEE Journal of*, 2007. [cited at p. 82]
- [13] A. Chakraborty, K. Duraisami, A. Sathanur, P. Sithambaram, L. Benini, A. Macii, E. Macii, and M. Poncino. Dynamic Thermal Clock Skew Compensation Using Tunable Delay Buffers, 2008. [cited at p. 7]
- [14] G. Chen and H. Chen. On-chip copper-based vs. optical interconnects: delay uncertainty, latency, power, and bandwidth density comparative predictions. *Interconnect . . .*, 40(4):434–446, 2006. [cited at p. 8, 9]
- [15] K.H. Cheng and W.B. Yang. A dual-slope phase frequency detector and charge pump architecture to achieve fast locking of phase-locked loop. In *Circuits and Systems II:*, volume 50, pages 892–896, 2003. [cited at p. 82]
- [16] Y. M. Chung and C. L. Wei. An all-digital phase-locked loop for digital power management integrated chips. *2009 IEEE International Symposium on Circuits and Systems*, pages 2413–2416, May 2009. [cited at p. 20]
- [17] N. Da Dalt. A design-oriented study of the nonlinear dynamics of digital bang-bang PLLs. *Circuits and Systems I: Regular Papers, IEEE*, 52(1):21–31, 2005. [cited at p. 101]
- [18] S.E. Esmaeili, A.J. Al-Khalili, and G.E.R. Cowan. A novel approach for skew compensation in energy recovery clock distribution networks, 2008. [cited at p. 7]
- [19] L. Fanori, A. Liscidini, and R. Castello. 3.3 GHz DCO with a frequency resolution of 150Hz for All-digital PLL. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 48–49. IEEE, 2010. [cited at p. 43]
- [20] J.S. Foresi, D.R. Lim, and L. Liao. Small radius bends and large angle splitters in SOI waveguides. *Photonics West . . .*, 1997. [cited at p. 8]
- [21] E.G. Friedman. Clock distribution networks in synchronous digital integrated circuits. *Proceedings of the IEEE*, 89(5):665–692, May 2001. [cited at p. 2]
- [22] W. Grollitsch, R. Nonis, and N. Da Dalt. A 1.4 psrms-period-jitter TDC-less fractional-N digital PLL with digitally controlled ring oscillator in 65nm CMOS. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 478–479. IEEE, 2010. [cited at p. 20]
- [23] V. Gutnik and A.P. Chandrakasan. Active GHz clock network using distributed PLLs. *Solid-State Circuits Conference, 2000. Digest of Technical Papers. 2000 IEEE International*, 35(11):1553–1560, 2000. [cited at p. 11, 13]
- [24] V. Gutnik and A.P. Chandrakasan. Active GHz clock network using distributed PLLs. *IEEE Journal of Solid-State Circuit*, 35:1553–1560, 2000. [cited at p. 17, 38, 110]

- [25] S. Hauck. Asynchronous design methodologies: An overview. *Proceedings of the IEEE*, 83(1):69–93, 1995. [cited at p. 1]
- [26] J.U. Horstmann, H.W. Eichel, and R.L. Coates. Metastability behavior of CMOS ASIC flip-flops in theory and test. *IEEE Journal of Solid State Circuits*, 24(1):146–157, 1989. [cited at p. 87]
- [27] H.Y. Hsieh, W. Liu, M. Clements, and P. Franzon. Self-calibrating clock distribution with scheduled skews, 1998. [cited at p. 7, 8]
- [28] S.P. Iyer and O. Oliaei. Phase-frequency synthesis using PLL-networks. *2008 Joint 6th International IEEE Northeast Workshop on Circuits and Systems and TAISA Conference*, pages 5–8, June 2008. [cited at p. 11]
- [29] M. Javidan, E. Zianbetov, and F. Anceau. A novel technique to reduce the metastability of Bang-Bang Phase Frequency Detectors. In *(ISCAS), 2011 IEEE*, pages 2577–2580, 2011. [cited at p. 107]
- [30] M. Javidan, E. Zianbetov, F. Anceau, D. Galayko, A. Korniienko, E. Colinet, G. Scorletti, J.M. Akre, and J. Juillard. All-digital PLL array provides reliable distributed clock for SOCs. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 2589–2592. IEEE, 2011. [cited at p. 3, 27, 38]
- [31] B. Jeon, Y. Moon, and T. Ahn. A study on 11 MHz ~ 1537 MHz DCO using tri-state inverter for DAB application. *TENCON 2009 - 2009 IEEE Region 10 Conference*, pages 1–5, November 2009. [cited at p. 40, 43]
- [32] R. Ji, L. Chen, G. Luo, X.X Zeng, J. Zhang, and Y. Feng. A Novel Low-Power Clock Skew Compensation Circuit, 2008. [cited at p. 7]
- [33] C. Johnson, D.H. Allen, J. Brown, S. Vanderwiel, R. Hoover, H. Achilles, C.Y. Cher, G.A. May, H. Franke, and J. Xenedis. A wire-speed powerTM processor: 2.3 GHz 45nm SOI with 16 cores and 64 threads. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, volume 44, pages 104–105. IEEE, 2010. [cited at p. 2]
- [34] A. Korniienko, E. Colinet, G. Scorletti, and E. Blanco. H loop shaping control for distributed PLL network. In *2009 PhD Research in Microelectronics and Electronics*, pages 336–339, 2009. [cited at p. 21, 29, 30, 32, 100, 109]
- [35] A. Korniienko, E. Colinet, G. Scorletti, E. Blanco, D. Galayko, and J. Juillard. A clock network of distributed ADPLLs using an asymmetric comparison strategy, 2010. [cited at p. 21, 29, 30, 32, 100, 109]
- [36] A. Korniienko, G. Scorletti, E. Colinet, E. Blanco, J. Juillard, and D. Galayko. Control law synthesis for distributed multi-agent systems: Application to active clock distribution networks. In *Proceedings of the 2011 American Control Conference*, pages 4691–4696. Laboratoire Ampere, UMR CNRS 5005, Ecole Centrale de Lyon, 36 Av. Guy de Collongue, 69134 Ecully cedex, France, IEEE, 2011. [cited at p. 21, 29, 30, 32, 33, 109]

- [37] V. Kratyuk, P.K. Hanumolu, K. Ok, K. Mayaram, and U.K. Moon. A Digital PLL with a Stochastic Time-to-Digital Converter. *2006 Symposium on VLSI Circuits, 2006. Digest of Technical Papers.*, pages 31–32, 2006. [cited at p. 44]
- [38] T. Kuendiger, L. MacEachern, and S. Mahmoud. A novel digitally controlled low noise ring oscillator. *2008 IEEE International Symposium on Circuits and Systems*, pages 1016–1019, May 2008. [cited at p. 43]
- [39] P.M. Levine and G.W. Roberts. A high-resolution flash time-to-digital converter and calibration scheme. In *Test Conference, 2004. Proceedings. ITC 2004. International*, pages 1148–1157, 2004. [cited at p. 94]
- [40] W. Li, B. Mbouombouo, and J. Leyrer. Low leakage PMOS on-chip decoupling capacitor cells compatible with standard CMOS cells, United States Patent 6608365, 2003. [cited at p. 63]
- [41] B. Lunitz and J. Jahns. Tolerant design of a planar-optical clock distribution system. *Optics communications*, 1997. [cited at p. 8]
- [42] H. Mizuno and K. Ishibashi. A noise-immune GHz-clock distribution scheme using synchronous distributed oscillators. In *Solid-State Circuits Conference, 1998. Digest of Technical Papers. 1998 IEEE International*, number 11, pages 404–405. IEEE, 1998. [cited at p. 10, 145]
- [43] L.H.A. Monteiro, R.V. dos Santos, and J.R.C. Piqueira. Estimating the critical number of slave nodes in a single-chain PLL network. *IEEE Communications Letters*, 7(9):449–450, September 2003. [cited at p. 27]
- [44] B. Moon, Y. Park, and D.K. Jeong. Monotonic Wide-Range Digitally Controlled Oscillator Compensated for Supply Voltage Variation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(10):1036–1040, October 2008. [cited at p. 44]
- [45] T. Olsson and P. Nilsson. A digitally controlled PLL for SoC applications. *IEEE Journal of Solid-State Circuits*, 39:751–760, 2004. [cited at p. 40, 44, 46]
- [46] F. O’Mahony, C.P. Yue, M.A. Horowitz, and S.S. Wong. A 10-GHz global clock distribution using coupled standing-wave oscillators. *Solid-State Circuits, IEEE Journal of*, 38(11):1813–1820, 2003. [cited at p. 10, 11]
- [47] VF Pavlidis. Clock distribution networks in 3-D integrated systems. ... *Integration (VLSI) Systems*, ... , 2011. [cited at p. 143]
- [48] R.K. Pokharel, A. Tomar, H. Kanaya, and K. Yoshida. Design of Highly Linear, 1GHz 8-bit Digitally Controlled Ring Oscillator with Wide Tuning Range in 0.18 um CMOS Process. In *Microwave Conference, 2008 China-Japan Joint*, number 2, pages 623–626. IEEE, 2008. [cited at p. 44, 46]
- [49] G.A. Pratt and J. Nguyen. Distributed synchronous clocking. *IEEE Transactions on Parallel and Distributed Systems*, 6:314–328, 1995. [cited at p. 11, 12, 13, 16, 25, 26, 27, 144]
- [50] P. E. Ross. Why cpu frequency stalled. *Spectrum, IEEE*, (April 2008):2010, 2008. [cited at p. 1]

- [51] S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, B. Cherkauer, J. Stinson, J. Benoit, R. Varada, and J. Leung. A 65-nm dual-core multithreaded Xeon processor with 16-MB L3 cache. *Solid-State Circuits, IEEE Journal of*, 42(1):17–25, 2007. [cited at p. 2]
- [52] S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, S. Kottapalli, and S. Vora. A 45 nm 8-core enterprise Xeon processor. *Solid-State Circuits, IEEE Journal of*, 45(1):7–14, 2010. [cited at p. 2]
- [53] M. Saint-Laurent and M. Swaminathan. A multi-PLL clock distribution architecture for gigascale integration. *Proceedings IEEE Computer Society Workshop on VLSI 2001. Emerging Technologies for VLSI Systems*, pages 30–35, 2001. [cited at p. 11, 27]
- [54] M. Saint-Laurent, P. Zarkesh-Ha, M. Swaminathan, and J.D. Meindl. Optimal clock distribution with an array of phase-locked loops for multiprocessor chips. *Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems. MWSCAS 2001 (Cat. No.01CH37257)*, 1:454–457, 2001. [cited at p. 11, 27]
- [55] S. L. Sam, A.P. Chandrakasan, and D. Boning. Variation issues in on-chip optical clock distribution. In *Statistical Methodology, IEEE International Workshop on, 2001 6th.*, pages 64–67. IEEE, 2001. [cited at p. 8, 9]
- [56] M. Sasaki. A 9.5 GHz 6ps-skew space-filling-curve clock distribution with 1.8 V full-swing standing-wave oscillators. In *Solid-State Circuits Conference, 2008. ISSCC 2008*, pages 518–520, 2008. [cited at p. 10]
- [57] M. Sasaki, M. Shiozaki, A. Mori, A. Iwata, and H. Ikeda. 12GHz Low-Area-Overhead Standing-Wave Clock Distribution with Inductively-Loaded and Coupled Technique. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 180–595. IEEE, 2007. [cited at p. 10]
- [58] J.P. Schoellkopf. Circuit indicating the phase relation between several signals having the same frequency, 1996. [cited at p. 7]
- [59] G. Scorletti and G. Duc. An LMI approach to decentralized H control. *International Journal of Control*, 74(3):211–224, 2001. [cited at p. xvii, 29]
- [60] R. Senthinathan, S. Fischer, H. Rangchi, and H. Yazdanmehr. A 600 MHz IA-32 microprocessor with enhanced data streaming for graphics and video, 1999. [cited at p. 7, 8, 9]
- [61] C. Shan, E. Zianbetov, M. Javidan, F. Anceau, M. Terosiet, S. Feruglio, D. Galayko, O. Romain, E. Colinet, and J. Juillard. FPGA implementation of reconfigurable ADPLL network for distributed clock generation. In *2011 International Conference on Field-Programmable Technology*, pages 1–4. Ieee, December 2011. [cited at p. 141]
- [62] L.K. Soh and Y.F.K. Edwin. An adjustable reset pulse phase frequency detector for phase locked loop. In *Design, 2009. ASQED 2009. 1st Asia, 2009*. [cited at p. 82]
- [63] R.B. Staszewski, J.L. Wallberg, S. Rezeq, C.M. Hung, O. Eliezer, S. Vemulapalli, C. Fernando, K. Maggio, R. Staszewski, N. Barton, M.C. Lee, P. Cruise, M. Entezari, K. Muhammad, and

- D. Leipold. All-Digital PLL and Transmitter for Mobile Phones. *Technology*, 40(12):2469–2482, 2005. [cited at p. 18]
- [64] S.K. Tewksbury and L.A. Hornak. Optical clock distribution in electronic systems. *The Journal of VLSI Signal Processing*, 1997. [cited at p. 8]
- [65] C. Thangaraj, R. Pownall, P. Nikkel, G. Yuan, K.L. Lear, and T Chen. Fully CMOS-Compatible On-Chip Optical Clock Distribution and Recovery. *Ieee Transactions On Very Large Scale Integration Vlsi Systems*, 18(10):1385–1398, 2010. [cited at p. 9]
- [66] C. Ti, Y. Liu, and T. Lin. A 2.4-GHz fractional-N PLL with a PFD/CP linearization and an improved CP circuit. *2008 IEEE International Symposium on Circuits and Systems*, pages 1728–1731, May 2008. [cited at p. 82]
- [67] J.A. Tierno, A.V. Rylyakov, and D.J. Friedman. A wide power supply range, wide tuning range, all static CMOS all digital PLL in 65 nm SOI. *IEEE Journal of Solid-State Circuits*, 43:42–51, 2008. [cited at p. xix, 40, 87, 89]
- [68] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez. Reducing power in high-performance microprocessors. In *Proceedings of the 35th annual Design Automation Conference*, pages 732–737. ACM, 1998. [cited at p. 6]
- [69] G. Tosik, F. Abramowicz, Z. Lisik, and F. Gaffiot. Clock Skew Analysis in Optical Clock Distribution Network. In *Proc 9th International Conference The Experience of Designing and Applications of CAD Systems in Microelectronics CADSM 07*, pages 422–425. Ieee, 2007. [cited at p. 8]
- [70] Stavros Volos, Ciprian Seiculescu, Boris Grot, Naser Khosro Pour, Babak Falsafi, and Giovanni De Micheli. CCNoC: Specializing On-Chip Interconnects for Energy Efficiency in Cache-Coherent Servers. *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip, (Nocs):67–74*, May 2012. [cited at p. 144]
- [71] K. Waheed and R.B. Staszewski. Time-domain behavioral modeling of a multigigahertz digital RF oscillator using VHDL. In *Circuits and Systems, 2005. 48th ...*, pages 1669–1672, 2005. [cited at p. 59]
- [72] Y. Wang, J. Yu, Y. Surya, and C. Huang. A compact delay-recycled clock skew-compensation and/or duty-cycle-correction circuit, 2011. [cited at p. 7]
- [73] R.B. Watson and R.B. Iknaian. Clock-buffer-chip with multiple-target automatic skew compensation, 1995. [cited at p. 7]
- [74] L. Xiu. *VLSI circuit design methodology demystified: a conceptual taxonomy*. Wiley-IEEE Press, 2008. [cited at p. 124]
- [75] L. Xu and S. Lindfors. A Digitally Controlled 2.4-GHz Oscillator in 65-nm. In *Circuit Design*, volume 00, pages 2007–2010, 2007. [cited at p. 43]
- [76] L. Xu, K. Stadius, and J. Ryyna nen. A wide-band digitally controlled ring oscillator. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 1983–1986. IEEE, 2010. [cited at p. 43]

- [77] T. Yamashita, T. Fujimoto, and K. Ishibashi. A dynamic clock skew compensation circuit technique for low power clock distribution, 2005. [cited at p. 7]
- [78] S.Y. Yang, W.Z. Chen, and T.Y. Lu. A 7.1 mW, 10 GHz all digital frequency synthesizer with dynamically reconfigured digital loop filter in 90 nm CMOS technology. *Solid-State Circuits, IEEE Journal of*, 45(3):578–586, 2010. [cited at p. 20]
- [79] M. Yuffe, M. Mehalel, E. Knoll, J. Shor, T. Kurts, E. Altshuler, E. Fayneh, K. Luria, and M. Zelikson. A Fully Integrated Multi-CPU, Processor Graphics, and Memory Controller 32-nm Processor. *Solid-State Circuits, IEEE Journal of*, (99):1–1, 2011. [cited at p. 2]
- [80] J. Zhuang, Q. Du, and T. Kwasniewski. A 3.3 GHz LC-based digitally controlled oscillator with 5kHz frequency resolution. *2007 IEEE Asian Solid-State Circuits Conference*, pages 428–431, November 2007. [cited at p. 43]
- [81] E. Zianbetov, F. Anceau, and M. Javidan. A Digitally Controlled Oscillator in a 65-nm CMOS process for SoC clock generation. In *(ISCAS), 2011 IEEE*, pages 2845–2848, 2011. [cited at p. 44, 80]
- [82] E. Zianbetov, M. Javidan, and F. Anceau. Design and VHDL modeling of all-digital PLLs. *NEWCAS Conference (NEWCAS), 2010 8th IEEE International*, pages 293–296, 2010. [cited at p. 30, 38, 80]