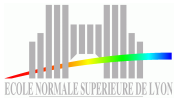
Optimized Hardware Arithmetic Operators

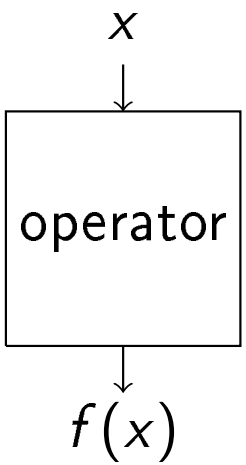
Romain Michard
Arénaire group

June 25^{th} 2008


Context

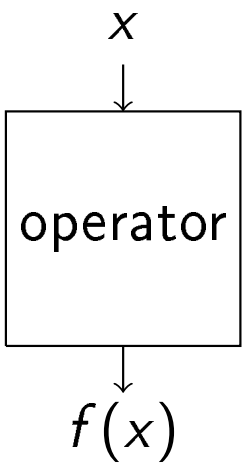
- Arithmetic operators



- Operations
  $+, -, \times, \div, \sqrt{}, \sin, \exp, \log, \ldots$
- Algorithms
  polynomial approximations, table-based methods, shift-and-add, ...
- Number representations
  Redundancy, point position, accuracy

Context

- Arithmetic operators

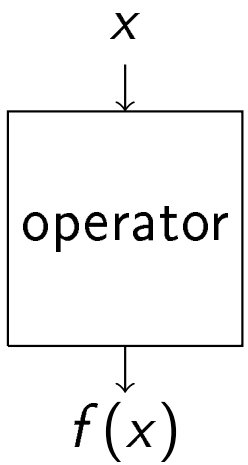


  - Operations
    $+, -, \times, \div, \sqrt{\ }, \sin, \exp, \log, \ldots$
  - Algorithms
    polynomial approximations, table-based methods, shift-and-add, ...
  - Number representations
    Redundancy, point position, accuracy

- Hardware or Software implementation

Context

- Arithmetic operators



  - Operations
    $+, -, \times, \div, \sqrt{}, \sin, \exp, \log, \ldots$
  - Algorithms
    polynomial approximations, table-based methods, shift-and-add, ...
  - Number representations
    Redundancy, point position, accuracy

- Hardware or Software implementation
- Optimization
  - Speed
  - Circuit area
  - Power consumption

Works

- Shift-and-add algorithms

`Divgen` and the division	SPIE 2005
E-method	SympA 2005
	FTFC 2005

- Function evaluation

Small operators	ASAP 2005 (Best Paper Award)
	SPIE 2006
	SiPS 2006
Generation of optimized evaluation operators	SympA 2006
	TSI 2008

Works

In this talk

- Shift-and-add algorithms
  - `Divgen` and the division — SPIE 2005
  - E-method — SympA 2005, FTFC 2005
- Function evaluation
  - Small operators — ASAP 2005 (Best Paper Award), SPIE 2006, SiPS 2006
  - Generation of optimized evaluation operators — SympA 2006, TSI 2008

Shift-and-Add Algorithms

Algorithms producing one result digit per iteration.
Most significant digit first

The same computation at each iteration:

- A current state (partial remainder)
- Choosing a new result digit (selection function)
- Calculating a new state (addition(s))

Example

The paper-and-pencil division

192	17
17	1
22	

Example

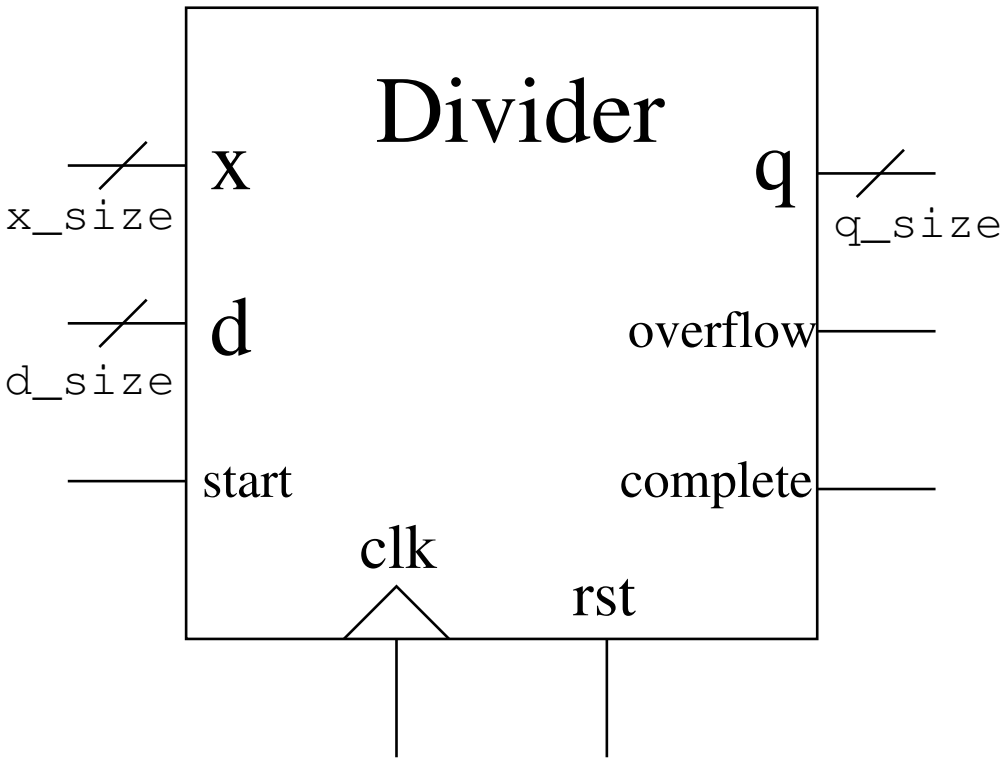
The paper-and-pencil division

192	17
17	11
22	
17	
5	

Divgen

Hardware Divider Generator

$$x = d \times q + rem$$



```
1  x_representation unsigned
2  d_representation unsigned
3  q_representation unsigned
4  x_size 9
5  d_size 6
6  q_size 7
7  algorithm SRT
8  q_radix 4
9  q_max_digit 3
10 partial_remainder_representation 2s_complement
11 step_adder RCA
12 #guard_bits 0
13 SRT_table_folding no
14 gray_encoding no
15 SRT_table_fig n&b
```

Divgen configuration file example.

- A C++ program (more than 5000 lines) under GPL license
- A divider hardware operator description generator
- From a simple specification to a VHDL optimized and synthesizable description

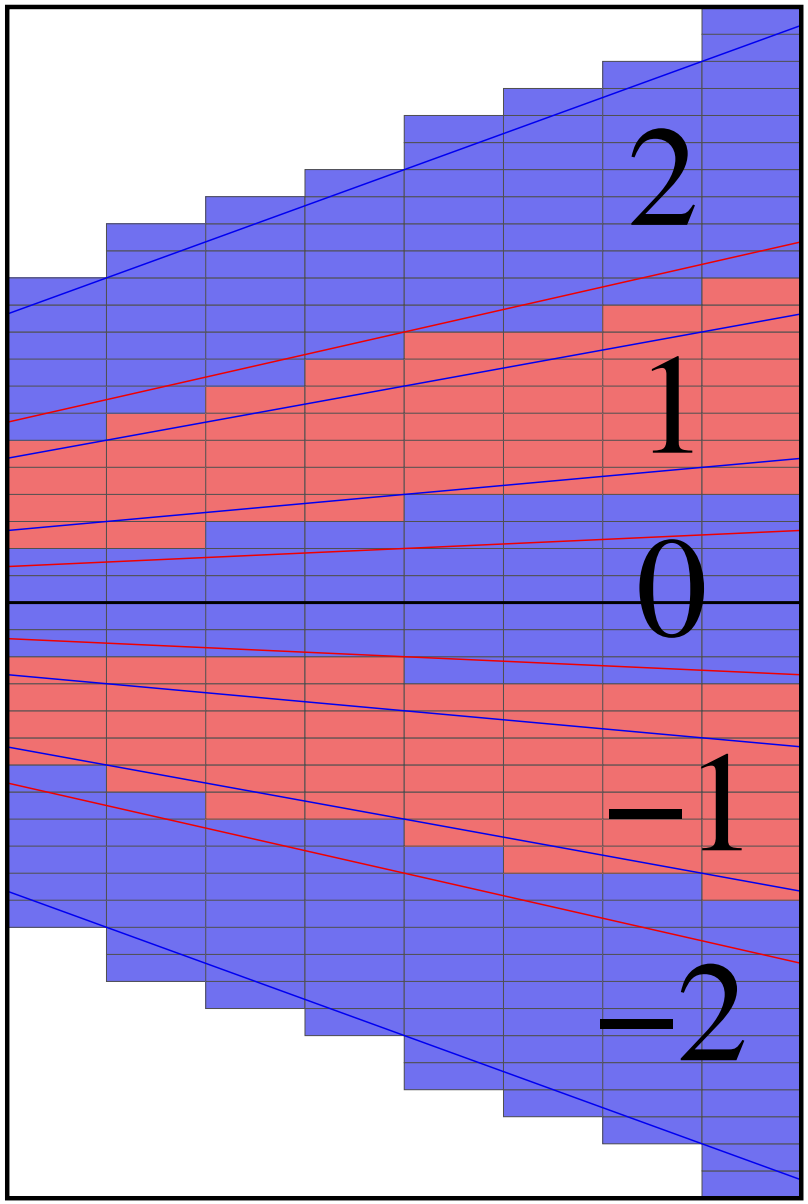
Supported algorithms:

- Restoring
- Nonrestoring
- SRT (a lot of options and parameters)

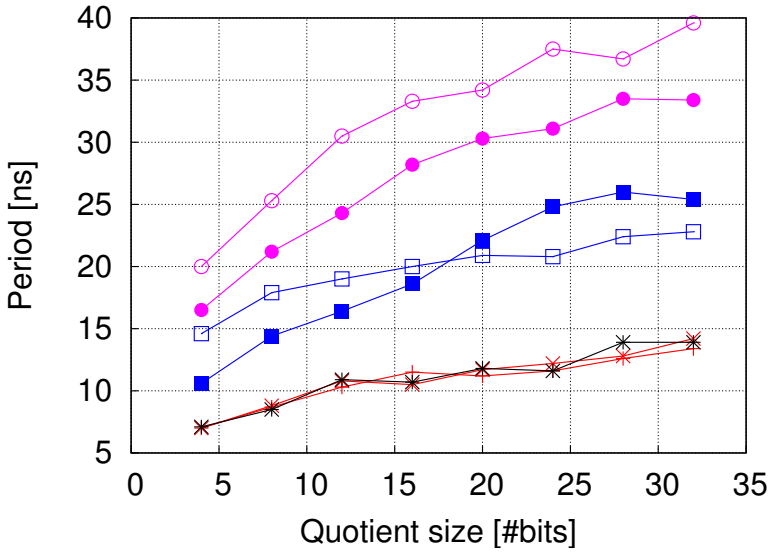
Validation of the operators

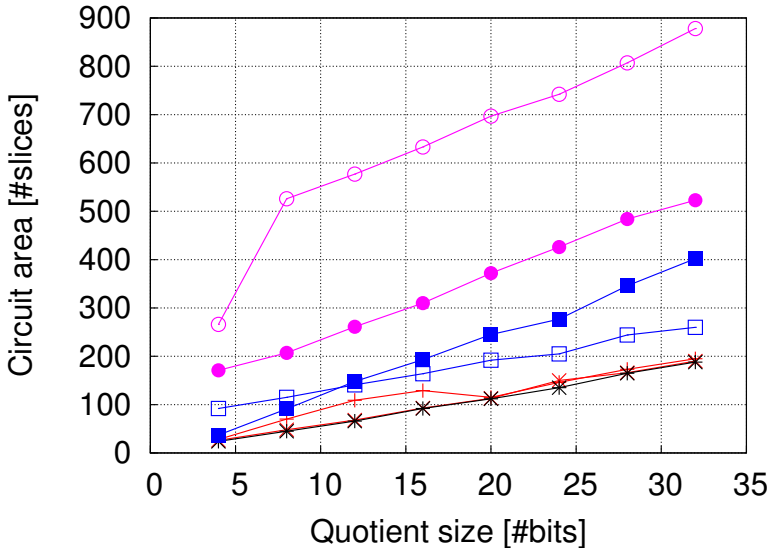
- Selection table verification
- Intermediate widths

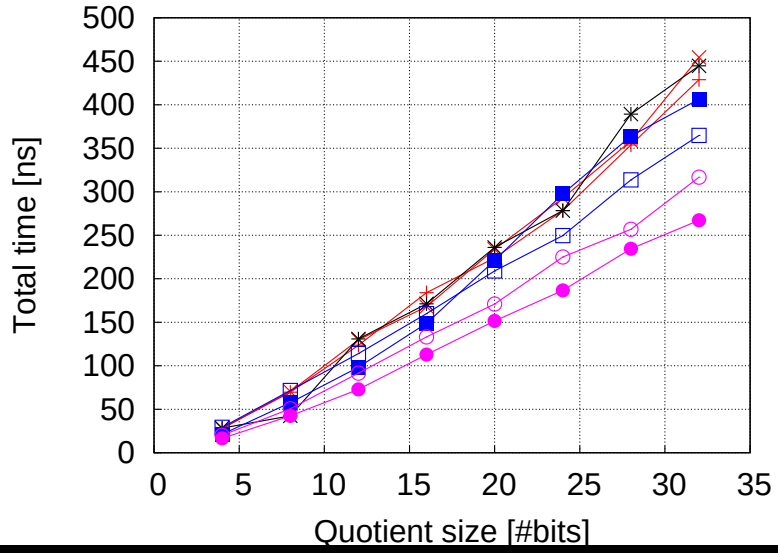
Selection table example

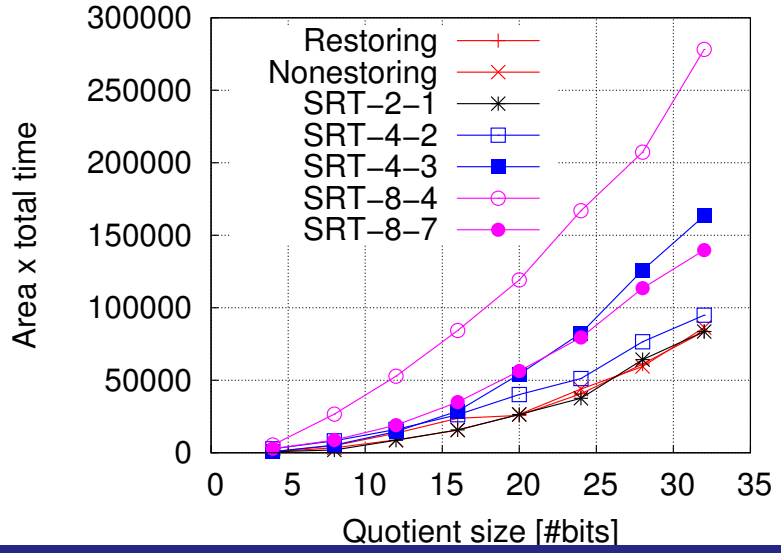


Some comparisons









E-Method

by M.D. Ercegovac in the 70's

Goal: evaluating rational fractions (polynomials) using a shift-and-add algorithm

$$\begin{pmatrix} 1 & -x & 0 & & \cdots & & & 0 \\ q_1 & 1 & -x & 0 & & \cdots & & 0 \\ q_2 & 0 & 1 & -x & 0 & & \cdots & 0 \\ & & \ddots & \ddots & \ddots & \ddots & & \vdots \\ & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & & & \ddots & \ddots & \ddots & 0 \\ & & & & & \ddots & 1 & -x \\ q_n & 0 & & \cdots & & & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ \\ \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ \\ \\ \vdots \\ p_{n-1} \\ p_n \end{pmatrix}$$

$$y_0 = \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^{n} p_i \cdot x^i}{1+\sum_{i=1}^{n} q_i \cdot x^i} \simeq f(x)$$

High Radix

Line i at iteration j:

$$w_i[j] = \beta \times \left(w_i[j-1] - d_0[j-1] \cdot q_i - d_i[j-1] + d_{i+1}[j-1] \cdot x\right)$$
$$d_i[j] = Sel(w_i[j])$$

High Radix

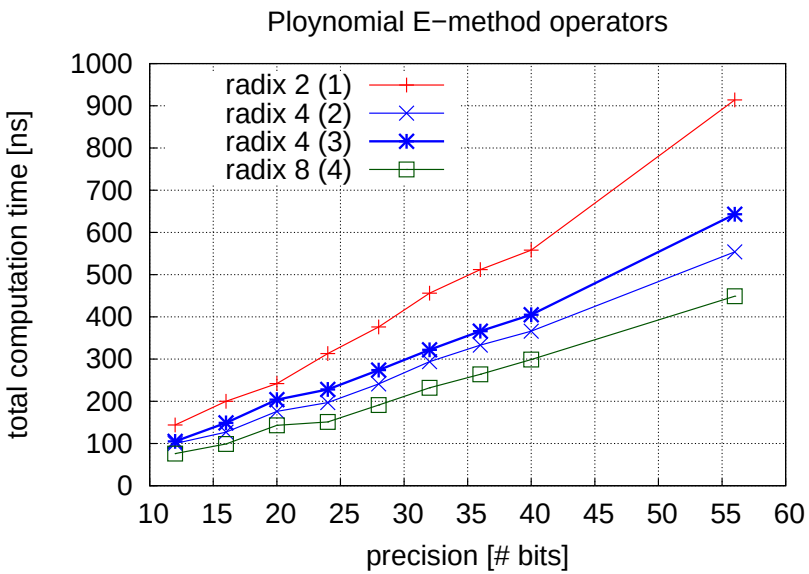
Line i at iteration j:

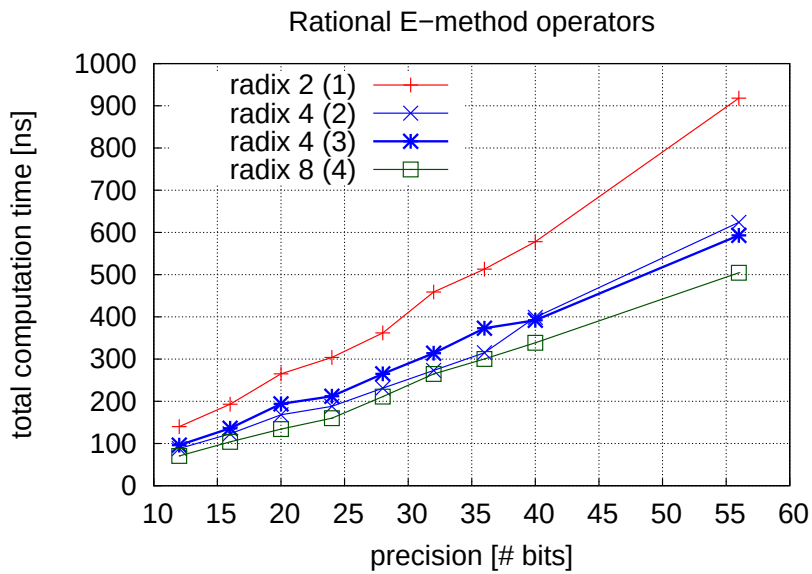
$$w_i[j] = \beta \times \left(w_i[j-1] - d_0[j-1] \cdot q_i - d_i[j-1] + d_{i+1}[j-1] \cdot x\right)$$
$$d_i[j] = Sel(w_i[j])$$

If β is high then there is less iterations but choosing d_i is more difficult.
Computing the E-method with a high radix is interesting anyway.

High Radix

Results





High radices give better total computation times.

Activity Reduction

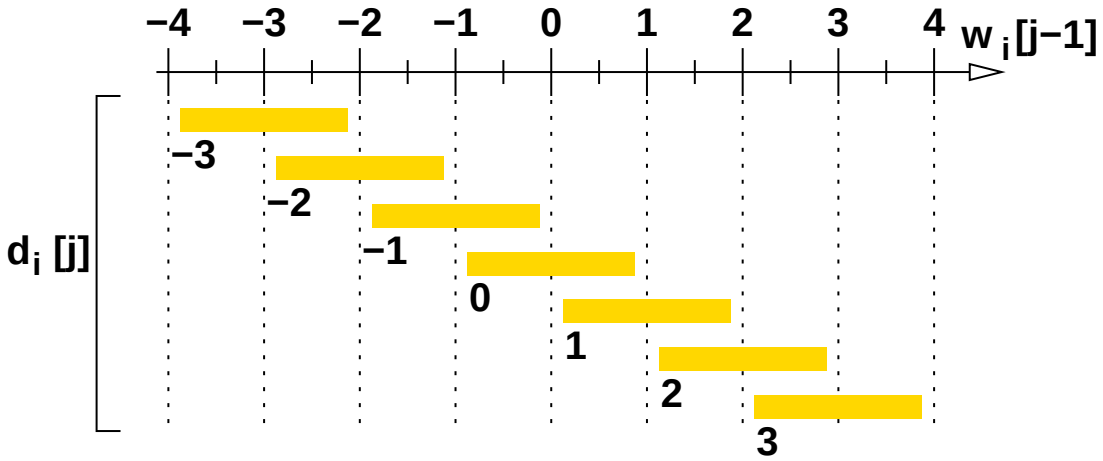
$$w_i[j] = \beta \times \left(w_i[j-1] - d_0[j-1] \cdot q_i - d_i[j-1] + d_{i+1}[j-1] \cdot x \right)$$

We have to choose a digit (d_i).

Activity Reduction

$$w_i[j] = \beta \times \left(w_i[j-1] - d_0[j-1] \cdot q_i - d_i[j-1] + d_{i+1}[j-1] \cdot x \right)$$

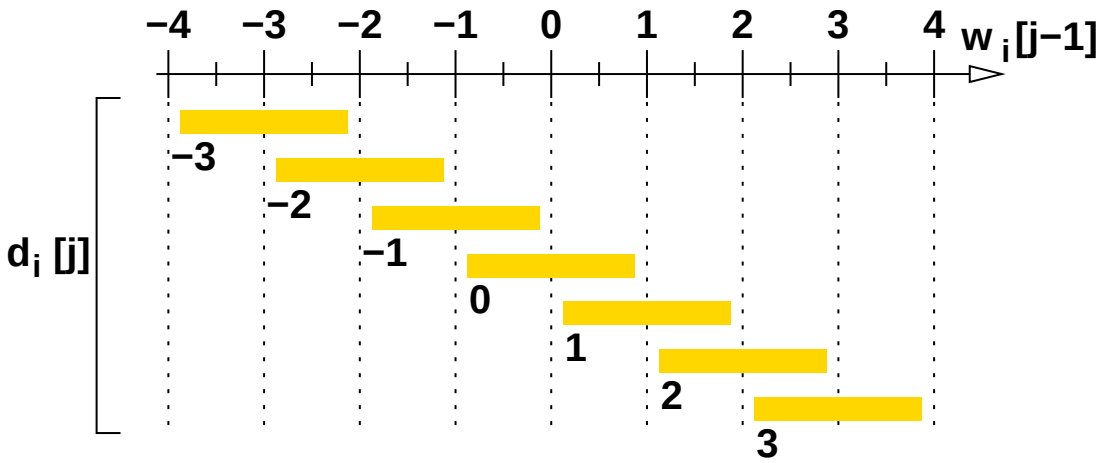
We have to choose a digit (d_i).
When using a redundant representation it is possible to choose several values.



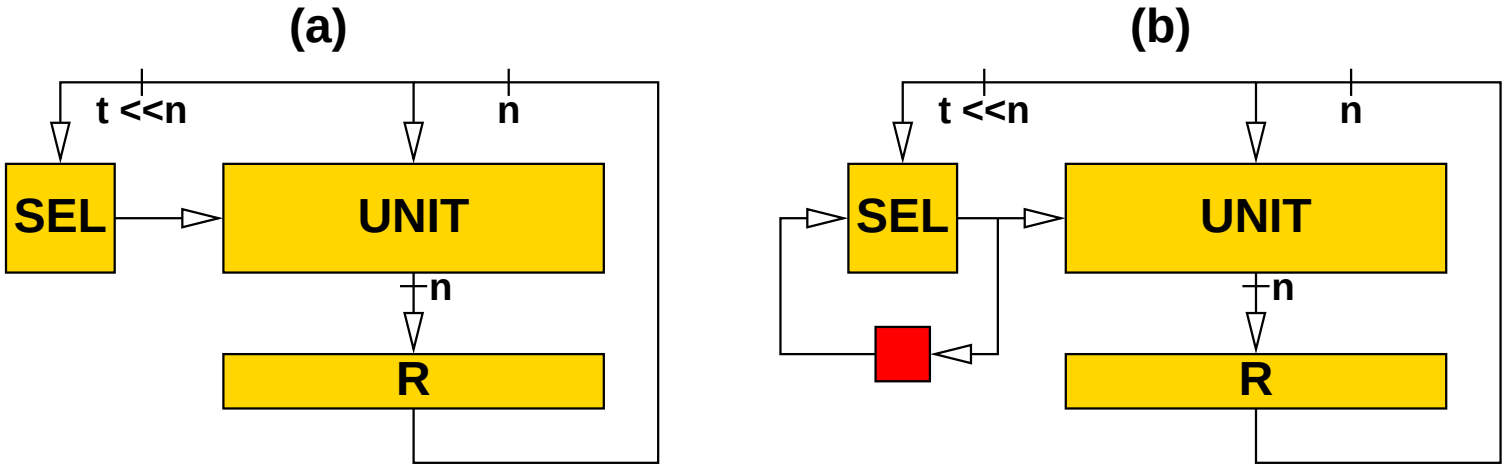
Activity Reduction

$$w_i[j] = \beta \times \left(w_i[j-1] - d_0[j-1] \cdot q_i - d_i[j-1] + d_{i+1}[j-1] \cdot x \right)$$

We have to choose a digit (d_i).
When using a redundant representation it is possible to choose several values.



Trying to choose the precedent value when possible
$\longrightarrow$ Modifying the selection tables



Selection function architecture

$$w_i[j] = \beta \times \left(w_i[j-1] - d_0[j-1] \cdot q_i - d_i[j-1] + d_{i+1}[j-1] \cdot x \right)$$

Functional bit-level simulations with Maple (degree 4, 16 bits)

		$d_i[j-1]$						
		-3	-2	-1	0	1	2	3
$d_i[j]$ Without memory	-3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	-2	0.01	0.02	0.01	0.02	0.03	0.03	0.01
	-1	0.00	0.01	0.10	0.34	0.25	0.02	0.00
	0	0.00	0.01	0.02	2.60	0.45	0.01	0.00
	1	0.03	0.02	0.26	0.58	0.25	0.00	0.01
	2	0.00	0.01	0.04	0.04	0.02	0.00	0.00
	3	0.00	0.01	0.00	0.00	0.00	0.01	0.00
$d_i[j]$ With memory	-3	0.00	0.00	0.00	0.02	0.01	0.00	0.00
	-2	0.00	0.01	0.00	0.01	0.07	0.00	0.00
	-1	0.00	0.00	0.02	0.03	0.02	0.02	0.02
	0	0.02	0.00	0.02	3.00	0.31	0.01	0.07
	1	0.00	0.00	0.00	0.53	0.49	0.01	0.03
	2	0.00	0.00	0.03	0.17	0.03	0.14	0.01
	3	0.00	0.00	0.02	0.08	0.01	0.02	0.01

$\Sigma_{OD} = 2.25$

$\Sigma_{OD} = 1.56$

-30%

Function Evaluation

- Many applications: computer science, image processing, signal processing, ...
- Different ways: table-based methods, polynomial approximations
- Different points of view: max error, average error

- Many applications: computer science, image processing, signal processing, ...
- Different ways: table-based methods, polynomial approximations
- Different points of view: max error, average error

Operators for signal processing and multimedia
$\Rightarrow$ Polynomial approximations

$$f(x) \simeq p_n \cdot x^n + ... + p_2 \cdot x^2 + p_1 \cdot x + p_0$$

Multiplications are expensive.
Low-degree polynomials
How to do with few multiplications?

Partial products array (PPA) for a 5-bit numbers multiplication:

$$
\begin{array}{cccccccccc}
 & & & & x_4 & x_3 & x_2 & x_1 & x_0 \\
 & & & \times & y_4 & y_3 & y_2 & y_1 & y_0 \\
\hline
 & & & & x_4y_0 & x_3y_0 & x_2y_0 & x_1y_0 & x_0y_0 \\
 & & & x_4y_1 & x_3y_1 & x_2y_1 & x_1y_1 & x_0y_1 & \\
 & & x_4y_2 & x_3y_2 & x_2y_2 & x_1y_2 & x_0y_2 & & \\
 & x_4y_3 & x_3y_3 & x_2y_3 & x_1y_3 & x_0y_3 & & & \\
x_4y_4 & x_3y_4 & x_2y_4 & x_1y_4 & x_0y_4 & & & &
\end{array}
$$

Partial products array (PPA) for a 5-bit numbers multiplication:

$$
\begin{array}{cccccccccc}
 & & & & x_4 & x_3 & x_2 & x_1 & x_0 & \\
 & & & \times & y_4 & \cancel{y_3} & y_2 & \cancel{y_1} & y_0 & \\
\hline
 & & & & x_4y_0 & x_3y_0 & x_2y_0 & x_1y_0 & x_0y_0 & \\
 & & & \cancel{x_4y_1} & x_3y_1 & x_2y_1 & x_1y_1 & x_0y_1 & & \\
 & & x_4y_2 & x_3y_2 & x_2y_2 & x_1y_2 & x_0y_2 & & & \\
 & \cancel{x_4y_3} & x_3y_3 & \cancel{x_2y_3} & \cancel{x_1y_3} & \cancel{x_0y_3} & & & & \\
x_4y_4 & x_3y_4 & x_2y_4 & x_1y_4 & x_0y_4 & & & & &
\end{array}
$$

Partial products array (PPA) for a 5-bit numbers multiplication:

$$
\begin{array}{cccccccccc}
 & & & & x_4 & x_3 & x_2 & x_1 & x_0 & \\
 & & & \times & y_4 & \cancel{y_3} & y_2 & \cancel{y_1} & y_0 & \\
\hline
 & & & & x_4y_0 & x_3y_0 & x_2y_0 & x_1y_0 & x_0y_0 & \\
 & & & \cancel{x_4y_1} & x_3y_1 & x_2y_1 & x_1y_1 & x_0y_1 & & \\
 & & x_4y_2 & x_3y_2 & x_2y_2 & x_1y_2 & x_0y_2 & & & \\
 & \cancel{x_4y_3} & x_3y_3 & \cancel{x_2y_3} & x_1y_3 & \cancel{x_0y_3} & & & & \\
x_4y_4 & x_3y_4 & x_2y_4 & x_1y_4 & x_0y_4 & & & & &
\end{array}
$$

$\Rightarrow y_i$ are the polynomial coefficients bits

$\Rightarrow$ coefficients with few nonzero digits

Constraint: p_i have no more than 3 nonzero digits.
Problem: the `minimax` algorithm doesn't work anymore.

Constraint: p_i have no more than 3 nonzero digits.
Problem: the `minimax` algorithm doesn't work anymore.
Trying to get a *good* polynomial requiring few hardware resources in 4 steps:

Constraint: p_i have no more than 3 nonzero digits.
Problem: the `minimax` algorithm doesn't work anymore.
Trying to get a *good* polynomial requiring few hardware resources in 4 steps:

- Starting from the `minimax` polynomial

Constraint: p_i have no more than 3 nonzero digits.
Problem: the `minimax` algorithm doesn't work anymore.
Trying to get a *good* polynomial requiring few hardware resources in 4 steps:

- Starting from the `minimax` polynomial
- Quantifying the theoretical coefficients

Constraint: p_i have no more than 3 nonzero digits.
Problem: the `minimax` algorithm doesn't work anymore.
Trying to get a *good* polynomial requiring few hardware resources in 4 steps:

- Starting from the `minimax` polynomial
- Quantifying the theoretical coefficients
- Approximated powers

Constraint: p_i have no more than 3 nonzero digits.
Problem: the `minimax` algorithm doesn't work anymore.
Trying to get a *good* polynomial requiring few hardware resources in 4 steps:

- Starting from the `minimax` polynomial
- Quantifying the theoretical coefficients
- Approximated powers
- Fine tuning of the coefficients

Results

Example

sine function on $[0, \pi/4[$, 8-bit degree-2 approximation

Results

Example

sine function on $[0, \pi/4[$, 8-bit degree-2 approximation

- `Minimax`
  $p_{th}(x) = -0.0023098047 + 1.0540785973 \cdot x - 0.1882871881 \cdot x^2$

Error	Cost
$\epsilon_{th} = 0.23 \times 10^{-2}$	$3\times$, $2\pm$

Results

Example

sine function on $[0, \pi/4[$, 8-bit degree-2 approximation

- `Minimax`
  $p_{th}(x) = -0.0023098047 + 1.0540785973 \cdot x - 0.1882871881 \cdot x^2$

Error	Cost
$\epsilon_{th} = 0.23 \times 10^{-2}$	$3\times$, $2\pm$

- Constraints on the coefficients
  $p(x) = -(0.000000001)_2 + (1.0001001\overline{1})_2 \cdot x - (0.0011000001)_2 \cdot x^2$

Avg err.	Max err.	Cost
0.16×10^{-2}	0.43×10^{-2}	$1\times$, $2\pm$

Results

Example

sine function on $[0, \pi/4[$, 8-bit degree-2 approximation

- `Minimax`
  $p_{th}(x) = -0.0023098047 + 1.0540785973 \cdot x - 0.1882871881 \cdot x^2$

Error	Cost
$\epsilon_{th} = 0.23 \times 10^{-2}$	$3\times$, $2\pm$

- Constraints on the coefficients
  $p(x) = -(0.000000001)_2 + (1.0001001\overline{1})_2 \cdot x - (0.0011000001)_2 \cdot x^2$

Avg err.	Max err.	Cost
0.16×10^{-2}	0.43×10^{-2}	$1\times$, $2\pm$

- Powers approximation

Avg err.	Max err.	Cost
0.69×10^{-2}	0.23×10^{-1}	$7\pm$

Results

Example

sine function on $[0, \pi/4[$, 8-bit degree-2 approximation

- `Minimax`
  $p_{th}(x) = -0.0023098047 + 1.0540785973 \cdot x - 0.1882871881 \cdot x^2$

Error	Cost
$\epsilon_{th} = 0.23 \times 10^{-2}$	$3\times$, $2\pm$

- Constraints on the coefficients
  $p(x) = -(0.000000001)_2 + (1.0001001\overline{1})_2 \cdot x - (0.0011000001)_2 \cdot x^2$

Avg err.	Max err.	Cost
0.16×10^{-2}	0.43×10^{-2}	$1\times$, $2\pm$

- Powers approximation

Avg err.	Max err.	Cost
0.69×10^{-2}	0.23×10^{-1}	$7\pm$

- Fine tuning

Avg err.	Max err.	Cost
0.41×10^{-2}	0.18×10^{-1}	$6\pm$

Synthesis results

Synthesis on Xilinx Virtex-E FPGAs:

Size (bits)	Multipartite		SMSO		Our method	
	Area	Period	Area	Period	Area	Period
8	19	16.6	21	8	27	14.9
12	76	18.0	63	14	50	20.5
16	280	24.8	123	19	71	23.5

Comparison with Multipartite and SMSO methods for $\sin(x)$ on $[0, \pi/4[$.

Conclusion and Future Prospects

- Conclusion
  - Additions to `Divgen`: new algorithms, other shift-and-add operators,...
  - Power consumption estimation for the E-method
  - Function evaluation: new method to be included in the generators

Conclusion and Future Prospects

- Conclusion
  - Additions to `Divgen`: new algorithms, other shift-and-add operators,...
  - Power consumption estimation for the E-method
  - Function evaluation: new method to be included in the generators
- Future prospects
  - Hardware architectures: optimizations (more than arithmetic only, low power,...)
  - Postdoctoral research in Ares group at INSA Lyon
    Hardware architectures for networks