Arithmetic operators on $GF(2^m)$ for cryptographic applications: performance – power consumption – security tradeoffs

Danuta Pamuła

17th December 2012

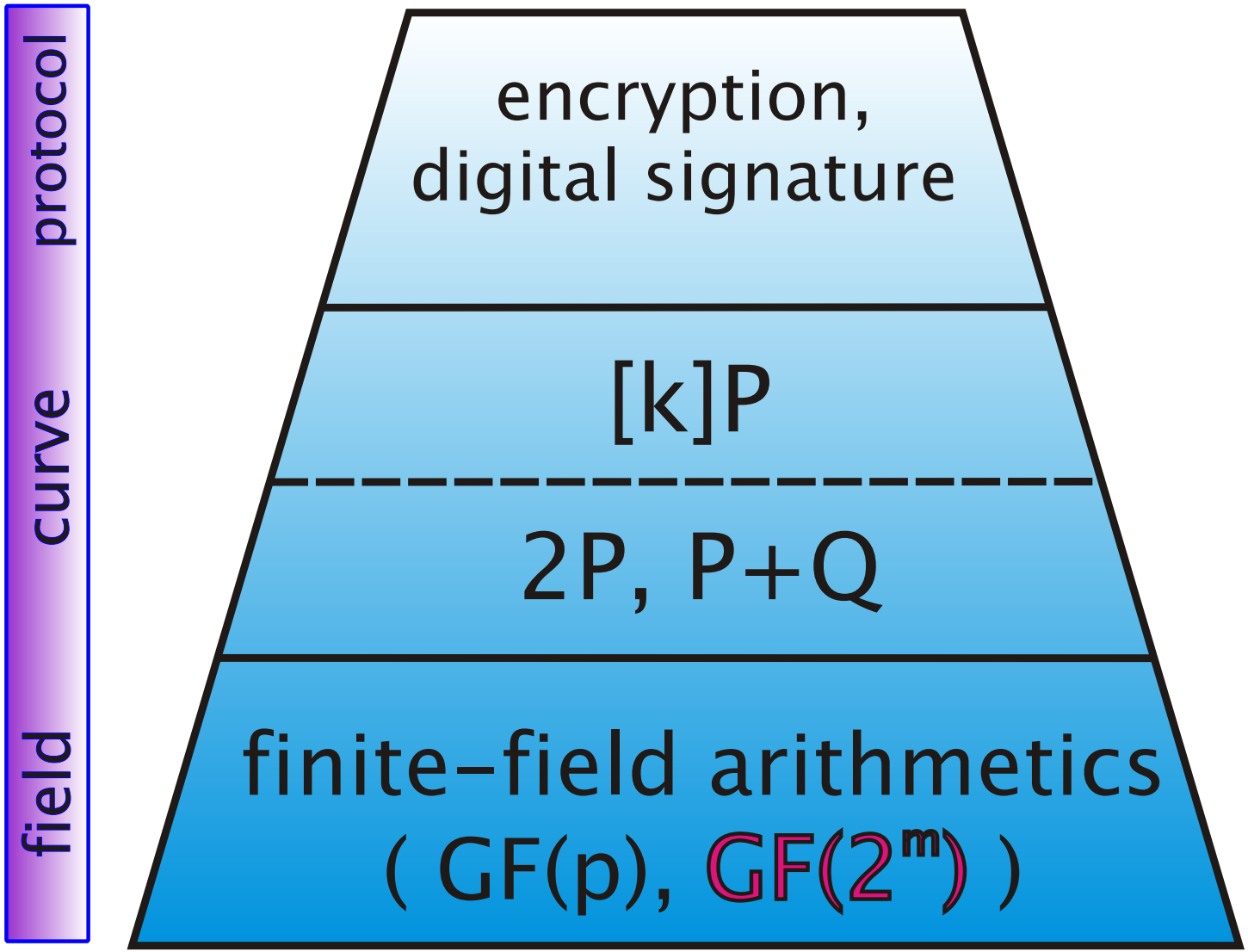
1. Introduction

- Arithmetic operators on $GF(2^m)$ - application, requirements
- Arithmetics in $GF(2^m)$ and elliptic curve cryptography
- Formulated thesis

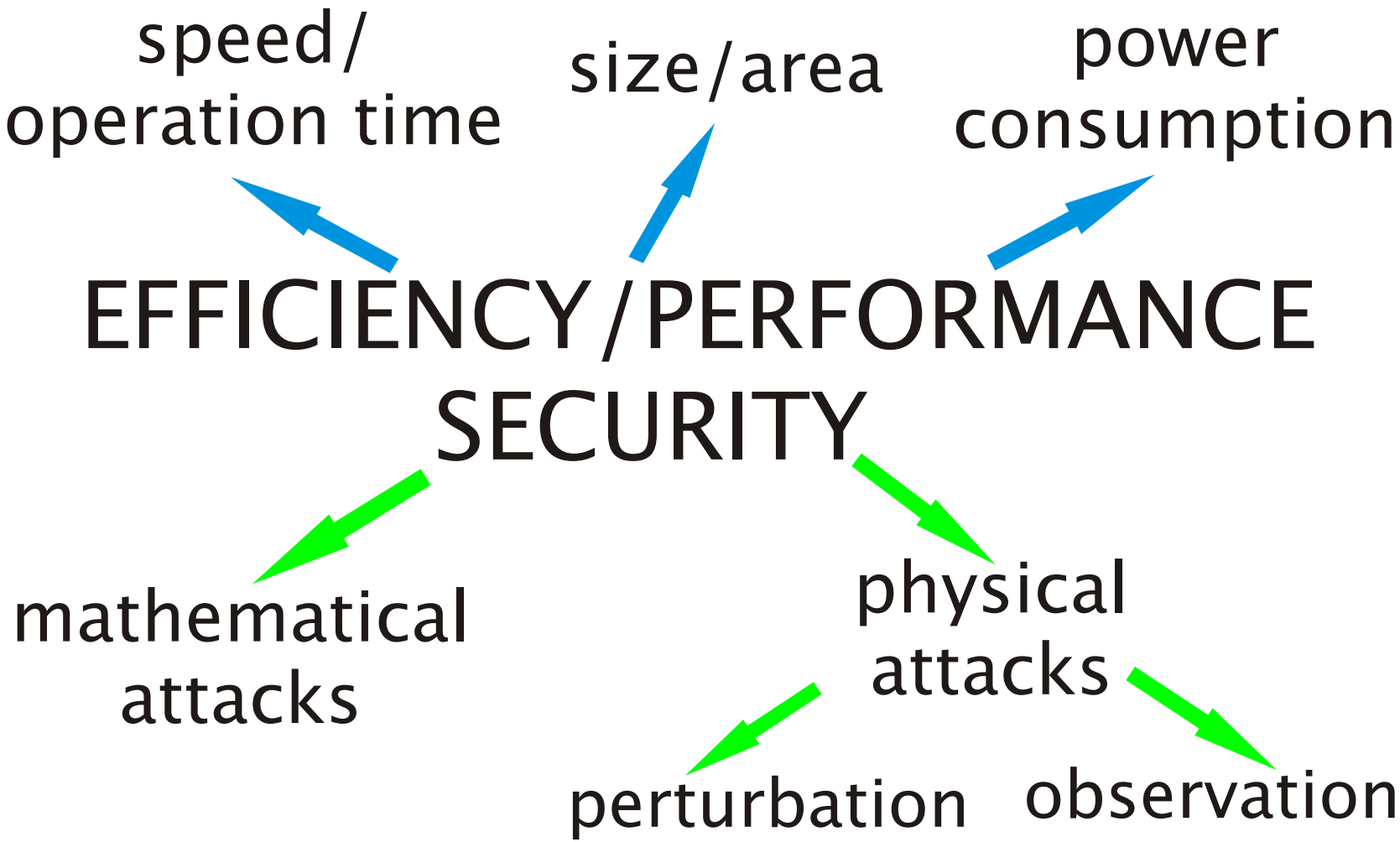
Arithmetic operators on $GF(2^m)$ - applications

- Cryptography :
  - symmetric: AES, ...
  - assymetric: RSA, ... , **Elliptic Curve Cryptography (ECC)**
- error correcting codes
- computational biology (e.g. modelisation of genetic network)
- computational and algorithmic aspects of commutative algebra
- digital signal processing
- ...

Arithmetics in $GF(2^m)$ and ECC



Cryptosystem - requirements



Security of ECC systems

$GF(2^m)$ curve protocol

e.g. ElGamal encryption

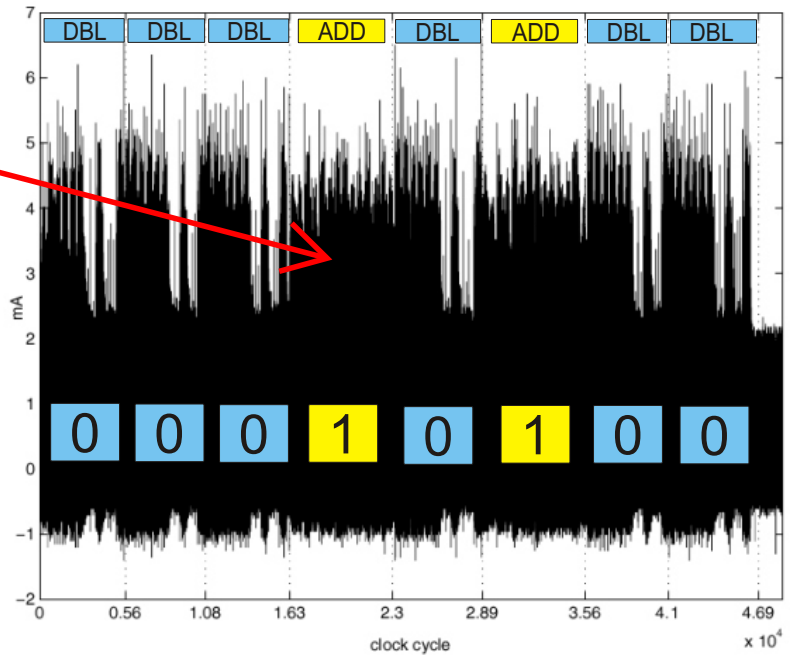
Double-and-Add

for i=0 to m-1
  if $k_i=1$ then Q=Q+P; //(ADD)
  P=2P; //(DBL)

$$R(x_R,y_R)=P(x_P,y_P)+Q(x_Q,y_Q)$$

$$x_R=\left(\frac{y_P+y_Q}{x_P+x_Q}\right)^2+\frac{y_P+y_Q}{x_P+x_Q}+x_P+x_Q+a;$$

$$y_R=\left(\frac{y_P+y_Q}{x_P+x_Q}\right)(x_R+x_P)+x_R+y_P;$$



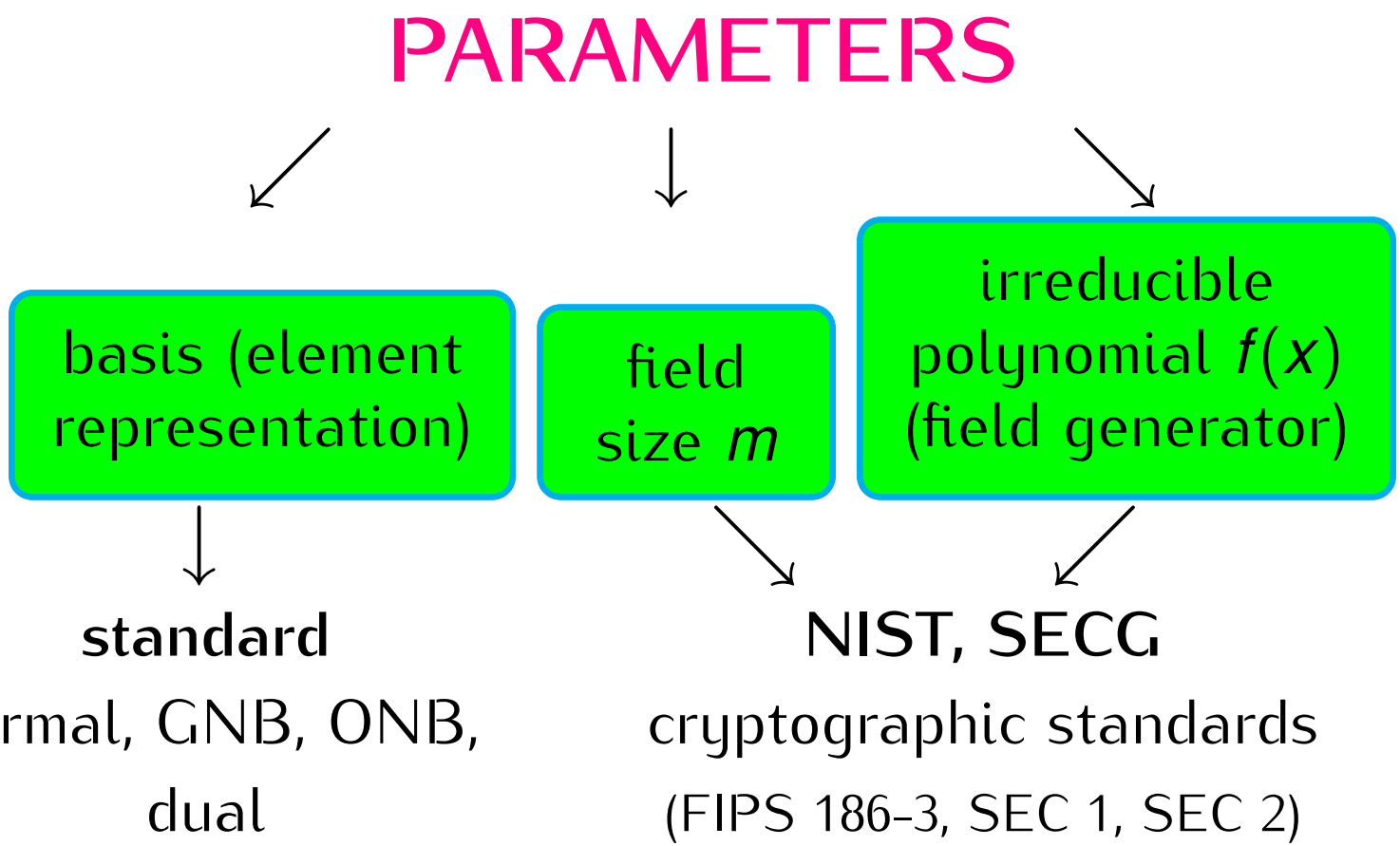
Thesis

It is possible to create ***efficient and secure*** against some side-channel ***power analysis*** attacks $GF(2^m)$ ***arithmetic operators*** dedicated to reconfigurable hardware.

2. Arithmetic in $GF(2^m)$ - efficient and secure hardware solutions

- Basics
- Addition
- Multiplication
- Proposed solutions

Arithmetics in $GF(2^m)$



GNB, ONB - Gaussian/Optimal Normal Basis, NIST - National Institute of Standards and Technology, SECG - Standards for Efficient Cryptography Group

Addition in $GF(2^m)$

Addition = XOR of binary polynomials

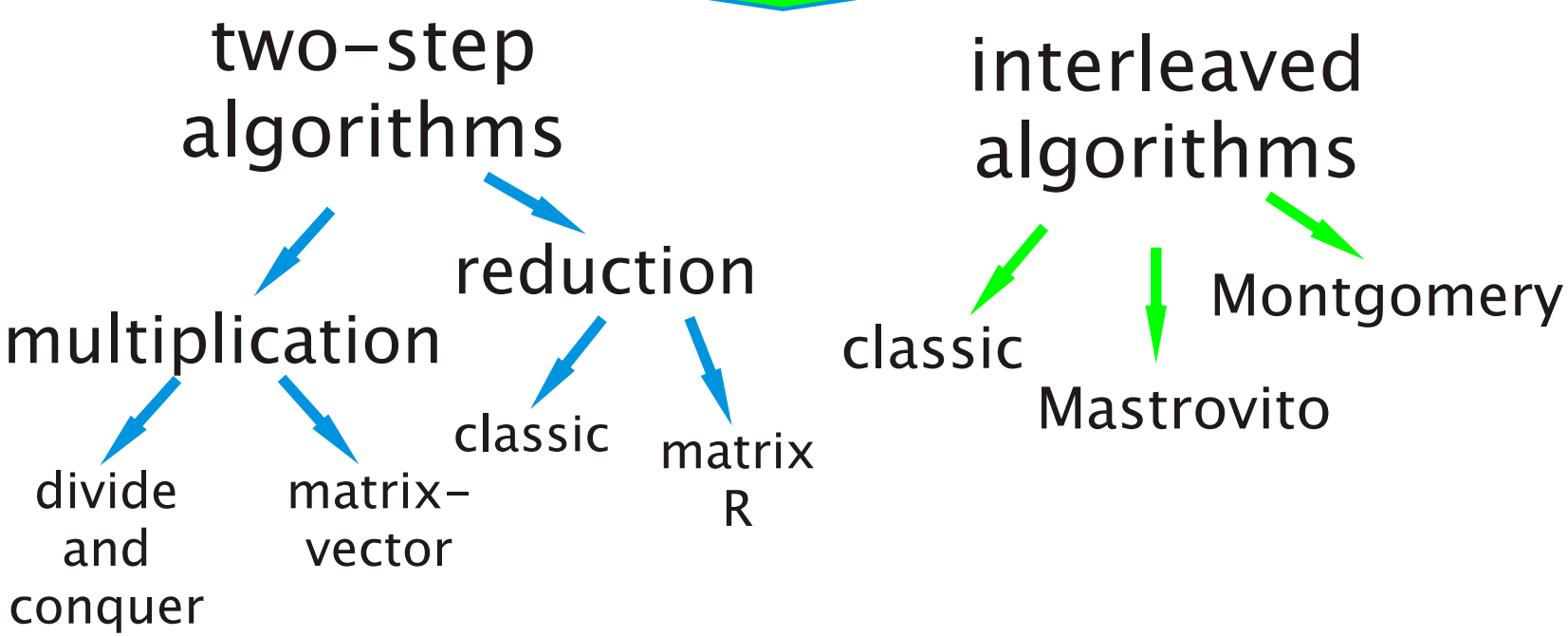
$$\mathbf{c} = \mathbf{a} \text{ XOR } \mathbf{b}$$

Propositions (data in processor are passed in words (16, 32-bit):
[1/2] Add every two incoming words of a, b, accumulate partial results in register c (1) or in BlockRAM (2);
[3] Wait for all words of a, b, add a and b;

field size m	(1)(Virtex-6)		(2)(Virtex-6)	
	[LUT]	[MHz]	[LUT]	[MHz]
163	21	771	26	562
233	**21**	**771**	**26**	**562**
283	22	767	28	560
409	22	767	28	560
571	**24**	**578**	**31**	**558**

Multiplication in $GF(2^m)$

$$c(x) = a(x)b(x) \bmod f(x)$$



Multiplication - Mastrovito matrix approach

Idea:

$$\mathbf{c = Mb},$$

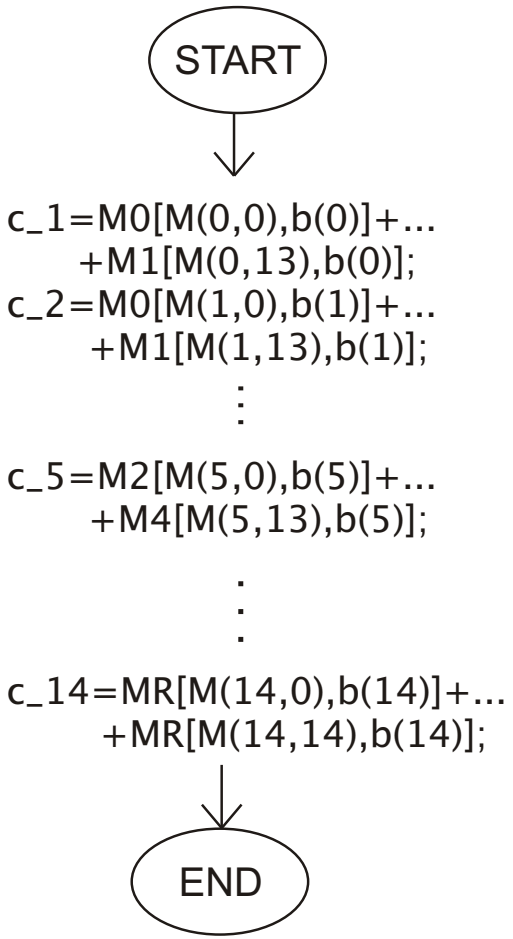
where **M** is a $m \times m$ **Mastrovito matrix**

Problems:

1. **Size** of matrix M ($m = 163,\ 233,\ 283,\ 409,\ 571$)
2. **Construction** of matrix M (iterative algorithm, combination of matrices A and R)
3. **Storing** matrix M
4. **Multiplication** of matrix M by vector b

M	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	M1	M1	Mc
1	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	M1	Mc
2	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	Mc
3	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	Mc
4	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	M4	M4	Mc
5	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	M4	Mc
6	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	Mc
7	M0	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	Mc
8	M0	M0	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	Mc
9	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	M3	M3	Mc
10	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	M3	Mc
11	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	Mc
12	M0	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	Mc
13	M0	M0	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	Mc
14	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR

b
b0
b1
b2
b3
b4
b5
b6
b7
b8
b9
b10
b11
b12
b13
b14



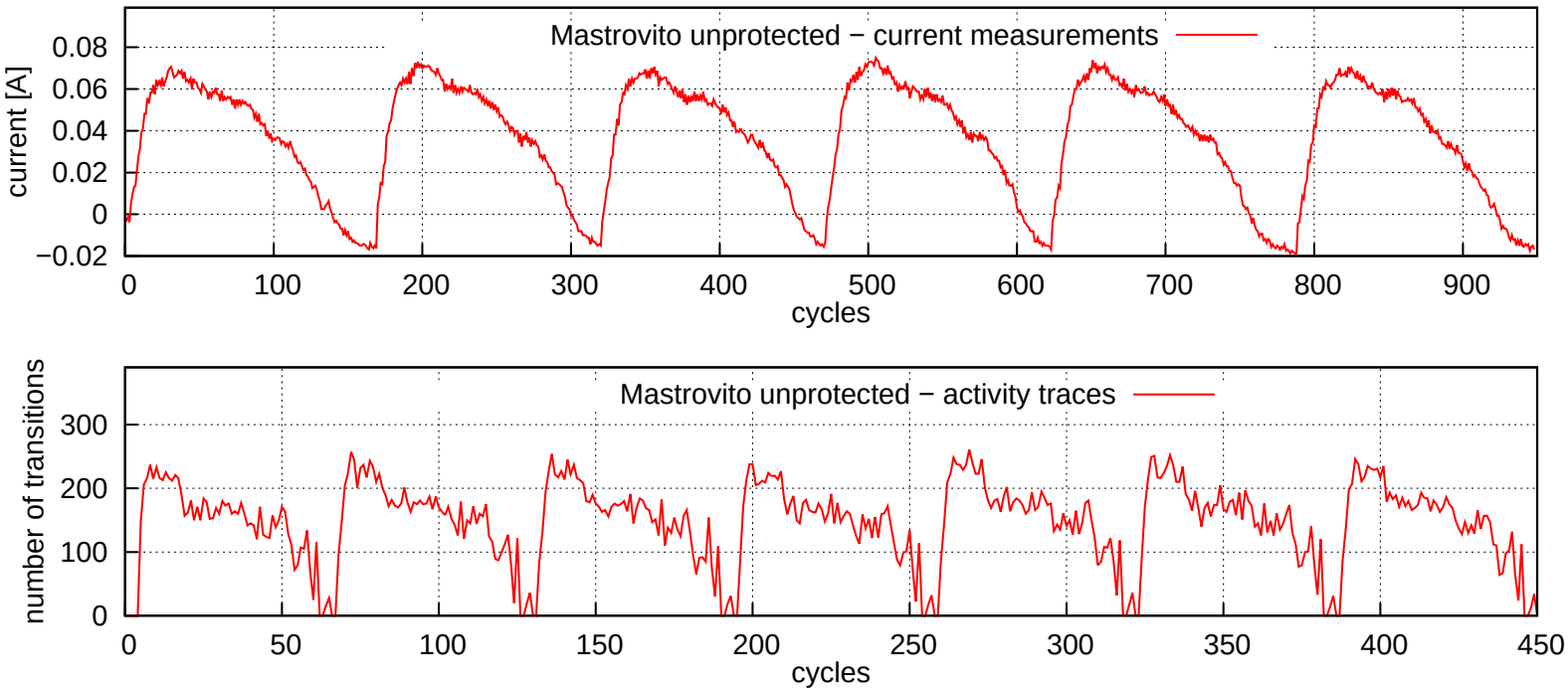
1. Partition of M into **submatrices 16×16 bit**
2. **Construction** of submatrices **"on-the-fly"** during multiplication, determiniation of submatrices with similar structures
3. Specialised **submultipliers** for each submatrix type - submultiplier constructs required submatrix during multiplication
4. The schedule of multiplication $M(i,j)b(i)$ is controlled by **Finite State Machine (FSM)**

$\hookrightarrow$

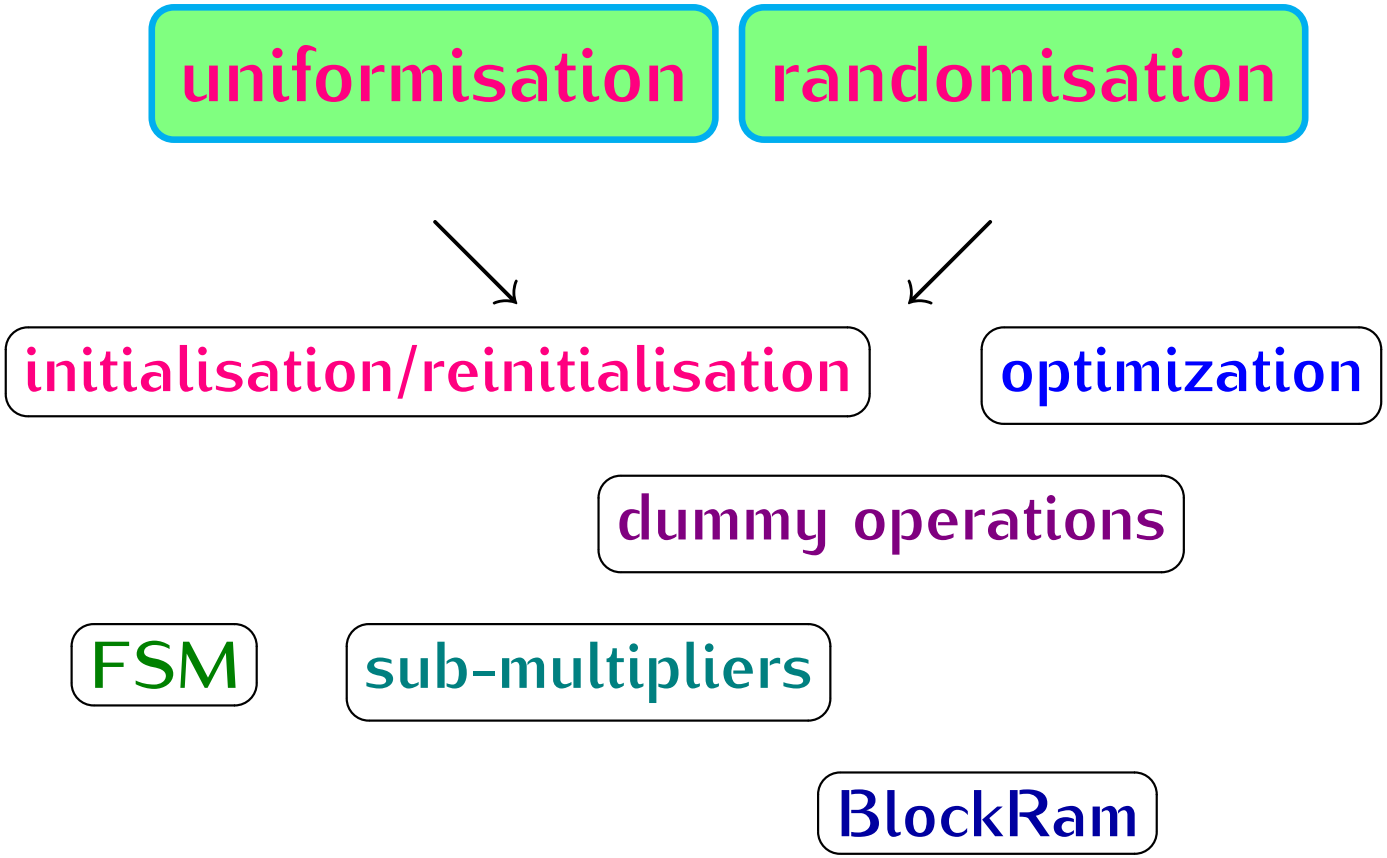
Security of the operator - power (activity) analysis

current consumption

activity monitor



Increasing security against power cryptanalysis



Note/Constraint: mainly algorithmic modifications, strictly hardware modifications were not considered (portability of the solution)

Optimization

most optimizations/decomposition left to synthesis tool

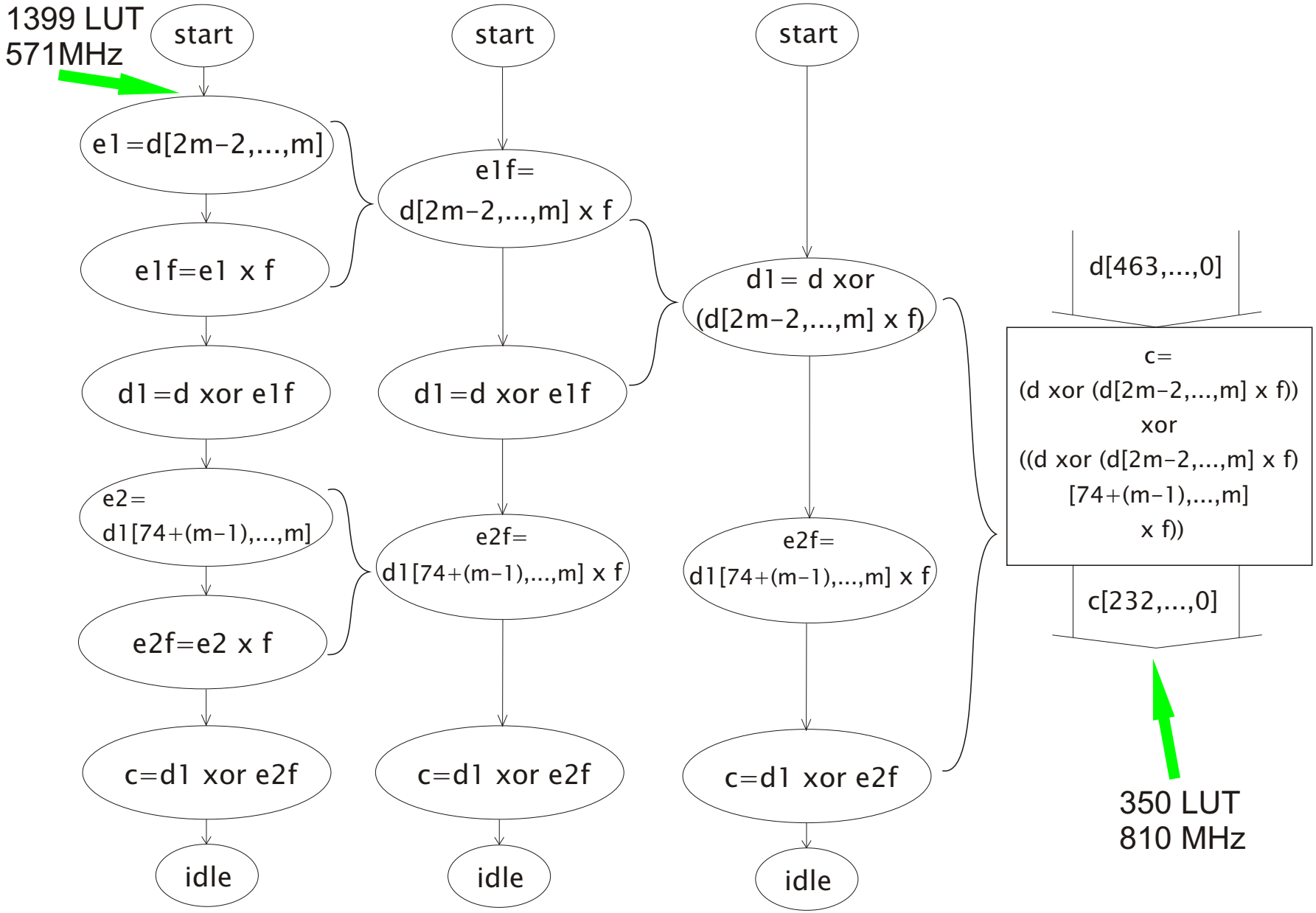
Proposition: (optimization/decomposition (if possible) "by hand")

- removal of auxiliary/unnecessary registers;
- partitioning of very large registers and complex, sequential operations into smaller/easier(simpler) ones;
- merging sequential operations

BlockRam

only LUT blocks were used to implement solutions

Proposition: units were partially implemented in BlockRams - according to some sources it dimnishes power consumption;


1399 LUT
571MHz
start
e1=d[2m-2,...,m]
e1f=e1 x f
d1=d xor e1f
e2=
d1[74+(m-1),...,m]
e2f=e2 x f
c=d1 xor e2f
idle
start
e1f=
d[2m-2,...,m] x f
d1=d xor e1f
e2f=
d1[74+(m-1),...,m] x f
c=d1 xor e2f
idle
start
d1= d xor
(d[2m-2,...,m] x f)
e2f=
d1[74+(m-1),...,m] x f
c=d1 xor e2f
idle
d[463,...,0]
c=
(d xor (d[2m-2,...,m] x f))
xor
((d xor (d[2m-2,...,m] x f)
[74+(m-1),...,m]
x f))
c[232,...,0]
350 LUT
810 MHz

FSM

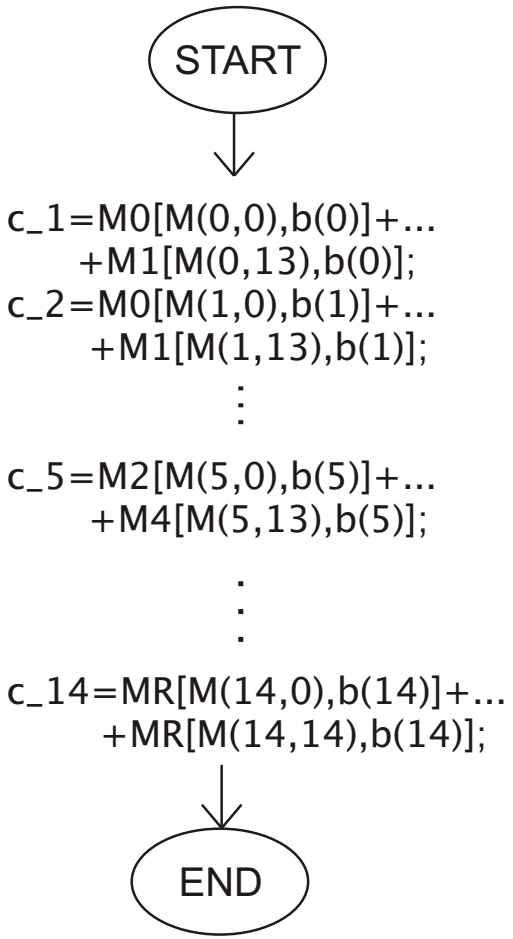
one FSM controling all submultipliers, many states (necessity of re-utilisation of submultipliers)

Proposition:

- **uniformisation:** same number of states, unification of number of registers/bit switching in each state, changed order of submultiplications;
- **randomisation:** each instance/type of submultiplier is controlled by different FSM (additional FSMs), each FSM is started at different moment of multiplication process;
- less states, more instances of submultipliers used, more activity in one state (no submultiplier is idle in any state);
- avoiding idle states between consecutive multiplications;

M	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	M1	M1	Mc
1	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	M1	Mc
2	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	Mc
3	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	Mc
4	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	M4	M4	Mc
5	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	M4	Mc
6	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	Mc
7	M0	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	Mc
8	M0	M0	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	Mc
9	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	M3	M3	Mc
10	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	M3	Mc
11	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	Mc
12	M0	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	Mc
13	M0	M0	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	Mc
14	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR

b
b0
b1
b2
b3
b4
b5
b6
b7
b8
b9
b10
b11
b12
b13
b14



Submultipliers

One instance for each type of submatrix/submultiplier

Proposition:

- using more than one instance of the same submultiplier;
- *note: submultipliers were optimised during efficiency analysis (by hand), these are combinational circuits*;

Dummy operation

In some states some submultipliers are idle, some registers are unused

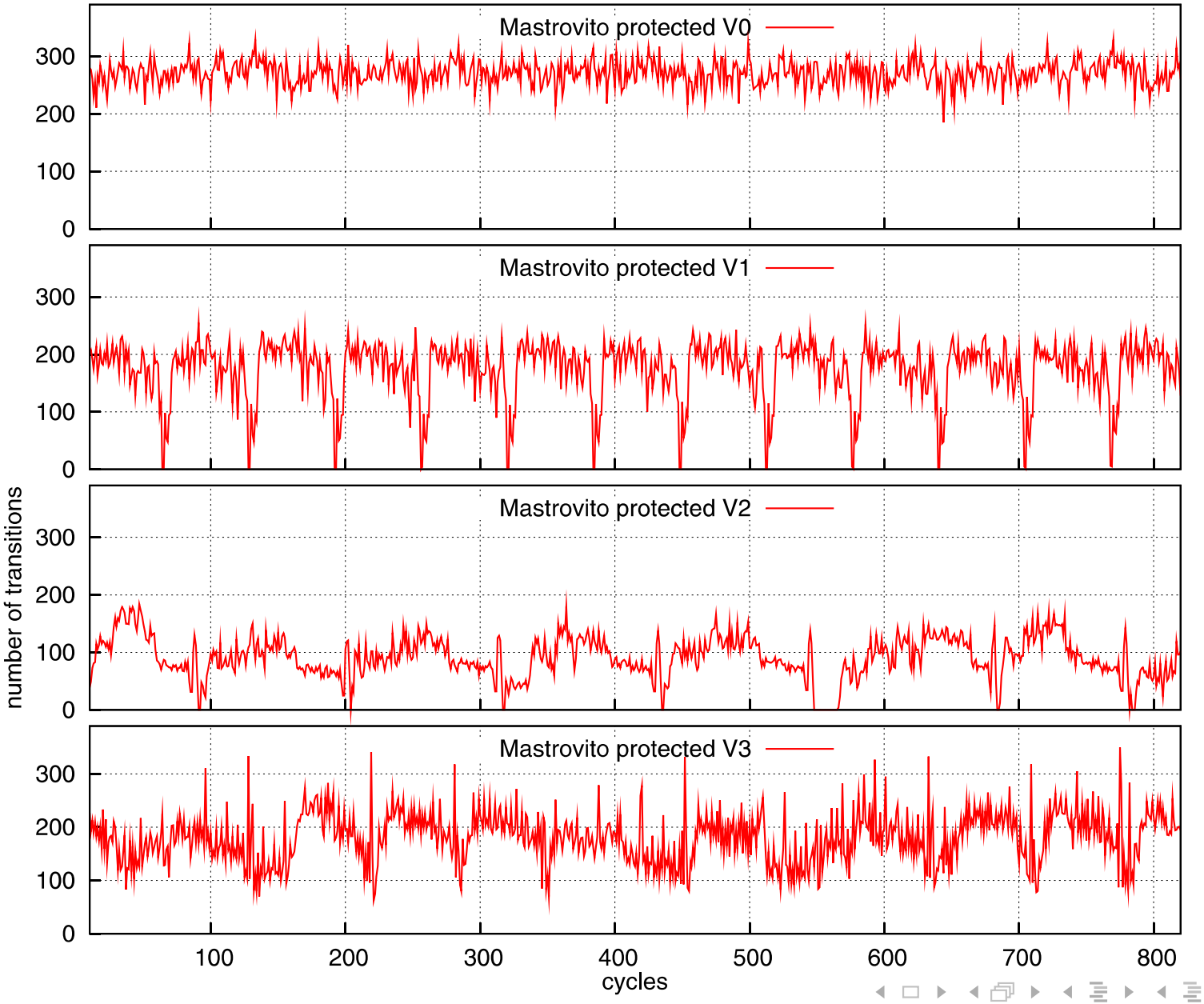
Proposition: dummy operations on unused registers;

Initialisation/reinitialisation

registers are resetted/reloaded at the beginning of multiplication

Proposition:

- resetting/reloading just before use;
- filling with random values (not constant), instead of zeroes;


Mastrovito protected V0
Mastrovito protected V1
Mastrovito protected V2
Mastrovito protected V3
number of transitions
cycles
0
100
200
300
400
500
600
700
800

Algorithm (Virtex 6)	area LUT ($\times\alpha$)	f MHz ($\times\alpha$)	clock cycles ($\times\alpha$)	AT
Mastrovito	**3760**	**297**	**75**	**0.95**
Mastrovito v0 (uniformisation)	3889 ($\times$1.03)	225 ($\times$0.75)	48 ($\times$0.64)	0.83
Mastrovito v1 (uniformisation)	3463 ($\times$0.92)	414 ($\times$1.39)	75 ($\times$1.00)	0.63
Mastrovito v2 (randomisation)	3700 ($\times$0.98)	306 ($\times$1.03)	avg.116 ($\times$1.55)	1.35
Mastrovito v3 (randomisation)	3903 ($\times$1.04)	319 ($\times$1.07)	avg.80 ($\times$1.07)	0.97

α: $\texttt{secured} = \alpha \times \texttt{original}$

$AT = area \times execution\_time$

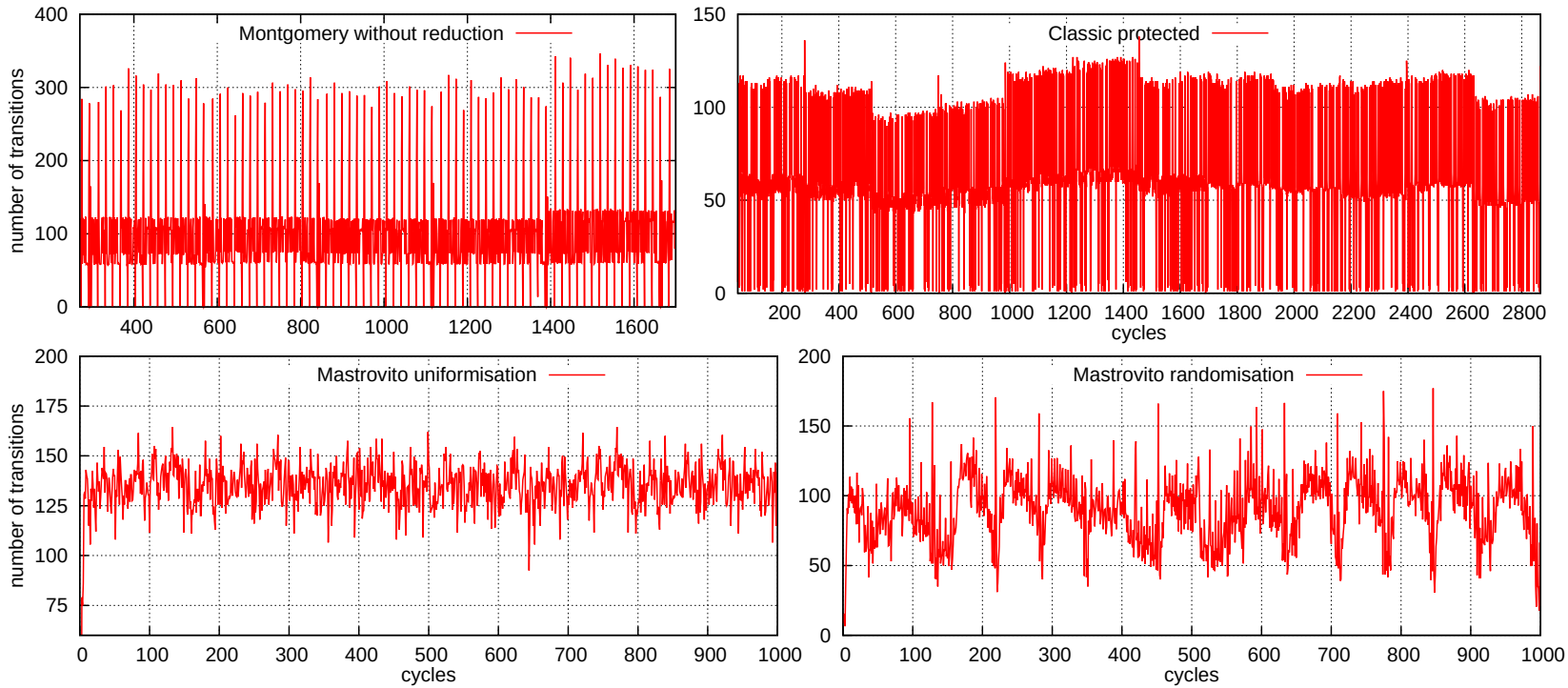
5. Summary - results, comments, future prospects

Summarising, as a result of conducted researches the following original results were obtained:

- **efficient in terms of speed and area $GF(2^m)$ hardware arithmetic operators dedicated to ECC** applications were proposed:

Algorithm (Virtex 6)	Area [LUT]	f [MHz]	clock cycles	AT
Classic 1	3638	302	264	3.18
Classic 2	2862	302	238	2.25
Mastrovito	3760	297	75	0.95
Montgomery (full)	3197	338	270	2.55

- **successful protections** against some power analysis side channel attacks for $GF(2^m)$ hardware arithmetic operators were developed;



- the **tradeoff** between efficiency and security of $GF(2^m)$ hardware arithmetic operators was found.;

Algorithm (Virtex 6)	area [LUT]	freq. [MHZ]	# cycles
Classical	2868	270	260
$\times\alpha$ factor	$\times$0.79	$\times$0.89	$\times$0.98
Montgomery	2099	323	264
$\times\alpha$ factor	$\times$0.96	$\times$1.00	$\times$0.98
Mastrovito v0	3889	225	48
$\times\alpha$ factor	$\times$1.03	$\times$0.75	$\times$0.64
Mastrovito v1	3463	414	75
$\times\alpha$ factor	$\times$0.92	$\times$1.39	$\times$1.00
Mastrovito v2	3700	306	avg.116
$\times\alpha$ factor	$\times$0.98	$\times$1.03	$\times$1.55
Mastrovito v3	3903	319	avg.80
$\times\alpha$ factor	$\times$1.04	$\times$1.07	$\times$1.07

Solution	m	FPGA	Area	max.f	T
Crowe	256	Virtex II	5267 LUT	44.91 MHz	5.75 us
Grabbe	233	XC2V6000 FF1517-4	37296 LUT	77 MHz	-
			11746 LUT	90.33 MHz	-
			36857 LUT	62.85 MHz	-
			45435 LUT	93.20 MHz	-
Rodriguez-Henriquez	191	XCV2600E	8721 CLB	-	82.4 us

Algorithm	m	FPGA	Area [LUT]	max.f [MHz]	T [us]
Classical mod	233	XC2V6000	4498	115	2.26
Montgomery			2099	129	2.04
Mastrovito v0			6387	183	0.26
Mastrovito v1			5154	107	0.7
Mastrovito v2			6364	113	1.02
Mastrovito v3			6387	100	0.8

Future prospects

- investigation of **inversion, division** in the field,
- investigation of **other** representations of elements of the field (**basis**) and its impact on the architecture,
- **integration with ECC processor** and further security evaluation
- **hardware countermeasures**: bus coding, clocks, special structures
- countermeasures against **other types of side-channel attacks**
- ...


Thank you for your attention
Dziękuję za uwagę
Merci pour votre attention
Gliwice 2012


Thank you for your attention
Dziękuję za uwagę
Merci pour votre attention
Gliwice 2012

point doubling $2P$