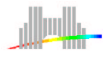


# Automatic Generation of Hardware Arithmetic Operators: a Small Example

Arnaud Tisserand  
INRIA Arénaire LIP

LIRMM  
March 31, 2005



- 1 Introduction
- 2 Hardware Evaluation of Functions
- 3 Example: Small Polynomial Approximations of Functions

## Part 1

### Introduction

## Research Field

### Computer arithmetic:

- 1 **algorithms** for the evaluation of various operations  
*addition, multiplication, division, operations involving constants, square root, power, sine, cosine, exponential, logarithm, filters, cryptography. . .*
- 2 **number systems**  
*integer, fixed/floating point, logarithmic system, residue number system, finite fields. . .*
- 3 **low-level implementations: hardware or/and software**  
*ASIC, FPGA, reconfigurable architectures, low-power, asynchronous circuits, general/special purpose processors, libraries (e.g. libm). . .*

**Problem** : find a good **adequacy** between an arithmetic (1, 2) and its implementation (3).

## Target Applications

Examples :

- scientific computing  
*high performances operations, multiple precision, linear algebra, high-level primitives, libraries. . .*
- digital signal processing  
*filters, analog-to-digital conversion, signal analysis, modem, codec, compression, noise attenuation. . .*
- image processing  
*norms, rotations, curve approximation, pattern computation, 3D → 2D projection. . .*
- digital control  
*change of coordinates system, non-linear functions, computation stability. . .*

## Generation of Hardware Operators

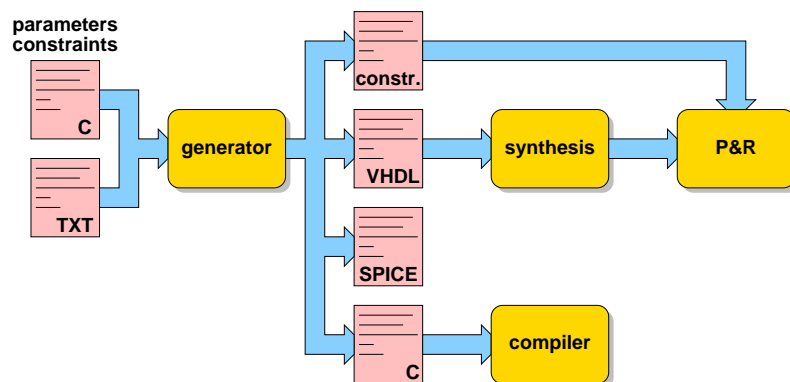
Problem: in current CAD tools, there is no support for advanced arithmetic (current limits: ALU,  $\pm$ ,  $\times$  on integers)

Solutions :

- **dedicated operator**  
*high performances, not portable, not extensible*
- **libraries**  
*complex development and maintenance, limited languages (VHDL, Verilog), limited algorithmic level, moderate portability, not extensible*
- automatic **generator** of operators  
*high performances, portable, extensible, exploration of the design space, but high initial development effort*

▶▶▶ our solution ◀◀◀

## A Generator in the Design Flow



Parameters and constraints examples: operands and result sizes, number system, algorithm, radix, target (ASIC, FPGA), loop unrolling, internal memory types. . .

## Developed Generators

- Small polynomial approximations of functions (**example**)
- Division units: Divgen project (shift-and-add algorithms)
- Machine efficient polynomials library (MEPLib project)
- Linear applications (multiplication of vector by a constant matrix)
- Multipartite tables
- Modular multipliers for FPGAs
- Multipliers and dividers for FPGAs with embedded hardwired small multipliers (e.g. Virtex II)

Links to some generators: <http://www.ens-lyon.fr/LIP/Arenaire/>

## A Possible Classification of Operations

From the point of view of computer arithmetic, we classify the operations in 3 main groups (order: growing complexity):

- arithmetic operations:  $x + y$ ,  $x \times y$ ,  $x \times y + z$  (and  $=, >, \geq, \dots$ )
- algebraic functions:  $x \div y$ ,  $\sqrt{x}$ ,  $\frac{1}{x}$ ,  $\frac{1}{\sqrt{x}}$ ,  $\frac{1}{\sqrt{x^2+y^2}}$ ...
- elementary functions:  $\sin(x)$ ,  $\cos(x)$ ,  $\exp(x)$ ,  $\log(x)$ ,  $\arctan(x)$ ...

## Part 2

### Hardware Evaluation of Functions

### Accuracy vs. speed

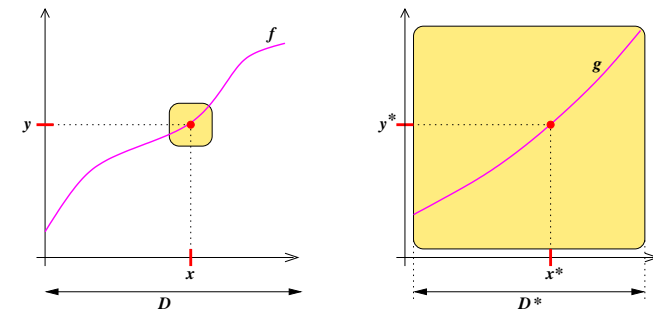
An evaluation function **cannot** be fast and accurate on a wide domain

Theoretical accuracy (# bits) for  $\sin(x)$  (minimax polynomial<sup>1</sup>):

$d$	$x \in$			
	$[0, \pi/4]$	$[0, \pi/2]$	$[0, \pi]$	$[0, 2\pi]$
1	6	3	0	0
2	8	6	5	0
3	14	9	5	3
4	17	13	10	3
5	23	17	10	7

<sup>1</sup>Evaluation of a degree- $d$  polynomial using  $d+$  and  $d\times$

### Standard scheme for the evaluation of a function



Step ❶: range reduction

Step ❷: evaluation

Step ❸: rounding and post-treatment

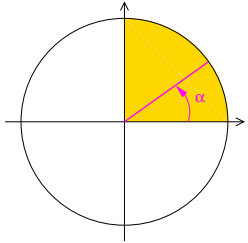
$$(f, x) \xrightarrow{RR} (g, x^*)$$

$$y^* = g(x^*)$$

$$y^* \xrightarrow{RPT} y$$

## Range Reduction Example

Evaluation of the sine and cosine functions on  $[0, 2\pi]$  based on the sine function on  $[0, \pi/2]$  :



$$\alpha \in [0, \pi/2[ \rightarrow \begin{cases} \alpha' = \alpha \\ \sin(\alpha) = \sin(\alpha') \\ \cos(\alpha) = \cos(\alpha') = \sin(\frac{\pi}{2} - \alpha') \end{cases}$$

$$\alpha \in [\pi/2, \pi[ \rightarrow \begin{cases} \alpha' = \alpha - \pi/2 \\ \sin(\alpha) = \sin(\frac{\pi}{2} - \alpha') \\ \cos(\alpha) = -\sin(\alpha') \end{cases}$$

$$\alpha \in [\pi, 3\pi/2[ \rightarrow \begin{cases} \alpha' = \alpha - \pi \\ \sin(\alpha) = -\sin(\alpha') \\ \cos(\alpha) = -\cos(\alpha') = -\sin(\frac{\pi}{2} - \alpha') \end{cases}$$

$$\alpha \in [3\pi/2, 2\pi[ \rightarrow \begin{cases} \alpha' = \alpha - 3\pi/2 \\ \sin(\alpha) = -\cos(\alpha') = -\sin(\frac{\pi}{2} - \alpha') \\ \cos(\alpha) = \sin(\alpha') \end{cases}$$

## Standard Range Reduction Methods

The two main range reduction methods:

- additive reduction:  $x^* = x - kC$
- multiplicative reduction:  $x^* = x/C^k$

The internal precision during the computation is critical!

Example for the logarithm function:  $x \in \mathbb{D} \xrightarrow{RR} x^* \in [1/2, 1]$

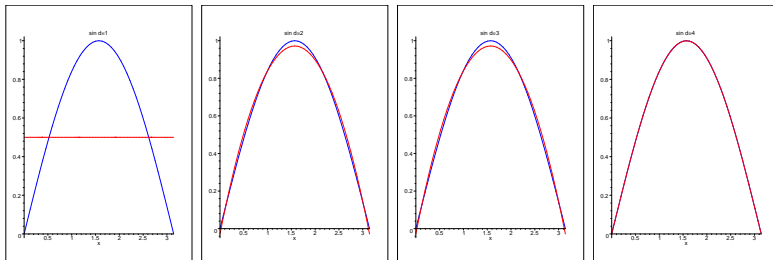
$$x^* = \frac{x}{2^k}$$

$$f(x) = \ln(x^*) + k \ln(2)$$

## Error Sources: the Method Error

Error source related to the **accuracy of the approximation**

Example : approximation of  $\sin(x)$  using a minimax polynomial of degree 1,2,3,4:



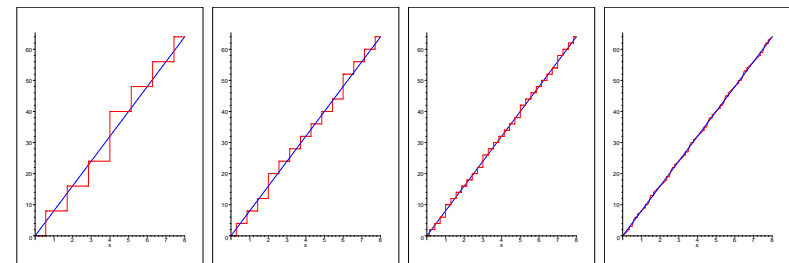
$d$	1	2	3	4
accuracy (# bits)	0.98	5.14	5.14	10.7

**Solution** : use the *best possible* approximation

## Error Sources: Computation Error

Error source related to the **finite precision** of the computations (also known as **rounding error**)

Example : approximation of  $f(x) = 8x$  for several precisions:



**Solution** : use *appropriate* precision and *control* the rounding errors

## Function Evaluation Methods

There are four main types of methods for the evaluation of functions:

- 1 polynomial approximations (or rational approximations)  
*fast and generic operator but requires multiplications (and a division)*
- 2 shift-and-add algorithms (e.g. CORDIC)  
*small operator but provides only one result digit at each iteration (improvement using high radix iterations), limited genericity*
- 3 tables and add algorithms  
*very fast operators but dedicated to only one function and limited to small accuracy*
- 4 function iteration algorithms (e.g. Newton)  
*fast operator, limited genericity, requires multiplications and limited to algebraic functions*

## Part 3

### Example: Small polynomial approximations of functions

joint work with Romain Michard and Nicolas Veyrat-Charvillon

## References

### Digital Arithmetic

Milos Ercegovac et Tomas Lang  
2003  
Morgan Kaufmann  
ISBN: 1–55860–798–6



### Elementary functions: Algorithms and Implementation

Jean-Michel Muller  
1997  
Birkhauser, Boston  
ISBN: 0–8176–3990–X

## Notations

- function  $f$  with inputs and outputs in **fixed-point** format
- argument  $x$  is in the domain  $[a, b]$  and the result  $f(x)$  is in the range  $[a', b']$
- polynomial approximation of degree  $d$
- input  $x$  is a  $w_I$ -bit number, output  $f(x)$  is a  $w_O$ -bit number
- $()_2$  denotes the binary representation ( $3.125 = (11.001)_2$ )
- Polynomial coefficients in **borrow-save** representation ( $-1 = \bar{1}$ )
- $g$  **guards bits** (intermediate computations on  $w_O + g$  bits)
- $P(x) = p_0 + p_1x + p_2x^2 + p_3x^3$

## Minimax Polynomial Approximations

Degree- $d$  **minimax polynomial approximation** to  $f$  on  $[a, b]$  is  $P^*$  :

$$\|f - P^*\|_\infty = \min_{P \in \mathcal{P}_d} \|f - P\|_\infty$$

$\mathcal{P}_d$  is the set of polynomials with real coefficients and degree at most  $d$

Minimax approximations computed using an algorithm due to Remes

Theoretical approximation error  $\epsilon_{th}$  :

$$\epsilon_{th} = \|f - P\|_\infty = \max_{a \leq x \leq b} |f(x) - P(x)|$$

**Example 1** Degree-3 minimax approximation of the sine function on  $[0, \pi/4]$  (results from Maple and truncated to 10 decimals),  $\epsilon_{th} = 0.0000474552$  (i.e. 14.3 bits of accuracy):

$$P(x) = -0.0000474552 + 1.0017332478x - 0.0095826177x^2 - 0.1522099691x^3$$

## Average and Maximum Error

- **output**( $P(x)$ ) effective result (computations on  $w_O + g$  bits)
- effective **total error**  $\epsilon = f(x) - \text{output}(P(x))$
- average total error  $\epsilon_{avg}$
- standard deviation  $\sigma$
- maximum error  $\epsilon_{max}$

Values computed of **all** values of  $x$  on  $[a, b]$

## The 3-bit Coefficients and Estimations of Powers of $x$ Method

The proposed method is based on the following steps:

**Step 1** determination of the minimax polynomial approximation  $P_{th}$  (done using Maple)

**Step 2** quantification of the coefficients of  $P_{th}$  to 3 non-zero bits, this gives polynomial  $P_q$

**Step 3** estimation of the powers of  $x$  in polynomial  $P_q$

**Step 4** fine tuning of the coefficients, this gives the polynomial  $P_t$

## 3-bit Quantification of the Coefficients

Goal: **replace** the large reduction tree of the multipliers by a **small number of additions**

Coefficient  $p_0$  is not quantified (additive term)

**Quantification** of coefficient  $p_i$  to  $NZ$  non-zero bits (here  $NZ \leq 3$ ) and a relative accuracy of  $2^{-k}$  (the span of the  $NZ$  bits is at most  $k$ -bit wide)

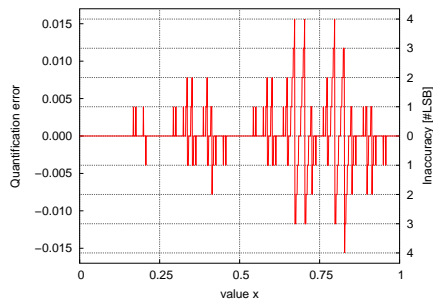
Minimum span :  $(11.01)_2$  rather than  $(10\bar{1}.01)_2$  for 3.25

**Example 2** Quantification of the value  $v = 1.0017332478$  with 3 non-zero bits for several spans (the quantified value is denoted  $v_q$ ):

- $k = 8$ ,  $v_q = 2^0 = (1.0000000)_2$ , (accuracy 9.1 bits)
- $k = 10$ ,  $v_q = 2^0 + 2^{-9} = (1.000000001)_2$ , (accuracy 12.1 bits)
- $k = 13$ ,  $v_q = 2^0 + 2^{-9} - 2^{-12} = (1.00000000100\bar{1})_2$ , (accuracy 15.3 bits)

## Quantification example

Quantification error for 10-bit values in  $[0, 1[$  with  $NZ = 3$  and  $k = 8$



- average error 0.0000534057 (14.1 bits of accuracy)
- standard deviation is 0.0031827562
- maximum error is 0.015625 (4 LSBs)

## Low estimation of $x^2$ on a real example

Complete computation of the sine function on  $[0, \pi/4[$  with  $w_I = w_O = k = 8, g = 2$  and  $NZ = 3$

$$P(x) = -(0.000000001)_2 + (1.000100\bar{1})_2 \times x - (0.0011000001)_2 \times x^2$$

$c$	$\epsilon_{avg}$	$\sigma$	$\epsilon_{max}$
3	$0.62 \times 10^{-2}$ (7.3)	$0.49 \times 10^{-2}$	$0.23 \times 10^{-1}$ (5.4)
4	$0.44 \times 10^{-2}$ (7.8)	$0.35 \times 10^{-2}$	$0.15 \times 10^{-1}$ (6.0)
5	$0.23 \times 10^{-2}$ (8.7)	$0.16 \times 10^{-2}$	$0.75 \times 10^{-2}$ (7.0)
6	$0.22 \times 10^{-2}$ (8.8)	$0.15 \times 10^{-2}$	$0.75 \times 10^{-2}$ (7.0)
7	$0.21 \times 10^{-2}$ (8.8)	$0.15 \times 10^{-2}$	$0.75 \times 10^{-2}$ (7.0)
all	$0.21 \times 10^{-2}$ (8.8)	$0.15 \times 10^{-2}$	$0.75 \times 10^{-2}$ (7.0)

## Low-Precision Estimations of the Powers of $x$

After quantification, some multiplications remain:  $x^2$  and  $x^3$

Goal: replace these “multiplications” by a **small number of additions**

Use **estimations** of  $x^2$  and  $x^3$  based on the  **$c$  first columns** of the partial products

Size and accuracy of the evaluation of  $p_2 \times x^2$  for several values of  $c$  with  $p_2 = 0.1882871881, w_I = k = 8$  bits and  $NZ = 3$

$c$	$\#_{pp}$	$\epsilon_{avg}$	$\sigma$	$\epsilon_{max}$
3	6	$0.16 \times 10^{-1}$ (5.9)	$0.12 \times 10^{-1}$	$0.47 \times 10^{-1}$ (4.4)
4	8	$0.78 \times 10^{-2}$ (6.9)	$0.58 \times 10^{-2}$	$0.23 \times 10^{-1}$ (5.4)
5	12	$0.59 \times 10^{-2}$ (7.4)	$0.42 \times 10^{-2}$	$0.18 \times 10^{-1}$ (5.8)
6	16	$0.28 \times 10^{-2}$ (8.4)	$0.19 \times 10^{-2}$	$0.88 \times 10^{-2}$ (6.8)
7	20	$0.18 \times 10^{-2}$ (9.0)	$0.11 \times 10^{-2}$	$0.52 \times 10^{-2}$ (7.5)
all	36	$0.13 \times 10^{-2}$ (9.5)	$0.67 \times 10^{-3}$	$0.32 \times 10^{-2}$ (8.3)

## Fine Tuning of the $p_i$ 's

Problem: truncate the partial products  $\implies$  underestimate  $x^i$

Solution: modify (increase) the coefficient  $p_i$

**Example 3** Sine on  $[0, \pi/4[$  with  $w_I = w_O = 12, NZ = 3, g = 2, c = 4$  for  $x^2$  and  $c = 8$  for  $x^3$

- $p_1$ : no modification
- $p_2$ :  $-2^{-7} - 2^{-9} + 2^{-13}$  modified to  $-2^{-7} - 2^{-9}$  ( $\epsilon_2$ :  $-0.27 \times 10^{-3} \longrightarrow 0.26 \times 10^{-3}$ )
- $p_3$ :  $-2^{-3} - 2^{-5} + 2^{-8}$  modified to  $-2^{-3} - 2^{-5}$  ( $\epsilon_3$ :  $-0.84 \times 10^{-3} \longrightarrow 0.56 \times 10^{-3}$ )

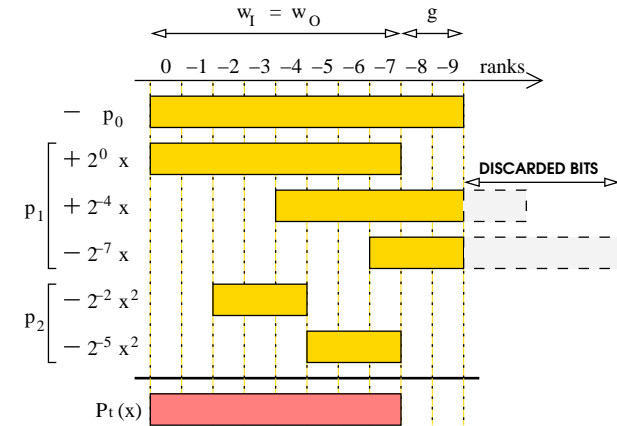
$\epsilon_{avg} = 0.91 \times 10^{-3}$  (9.9) modified to  $\epsilon_{avg} = 0.72 \times 10^{-3}$  (10.4) and two additions have been suppressed.

## A Complete Example (1/2)

- Sine on  $[0, \pi/4[$  with  $w_I = w_O = 8$  and  $d = 2$   
 $P_{th}(x) = -0.0023098047 + 1.0540785973x - 0.1882871881x^2$
- Quantification with  $k = 8$ ,  $NZ = 3$  and  $g = 2$   
 $P_q(x) = -(0.000000001)_2 + (1.000100\bar{1})_2x - (0.0011000001)_2x^2$
- Estimation of  $x^2$  with  $c = 3$
- Fine tuning of coefficient  $p_2$   
 $P_t(x) = -(0.000000001)_2 + (1.000100\bar{1})_2x - (0.0100\bar{1})_2x^2$

Step	$\epsilon_{avg}$	$\sigma$	$\epsilon_{max}$	Cost
Minimax $P_{th}$	$\epsilon_{th} = 0.23 \times 10^{-2}$ (8.7)			3 mult. + 2 add.
Quantification	$0.16 \times 10^{-2}$ (9.3)	$0.12 \times 10^{-2}$	$0.53 \times 10^{-2}$ (7.5)	1 mult. + 2 add.
$x^2$ estimation	$0.69 \times 10^{-2}$ (7.1)	$0.52 \times 10^{-2}$	$0.23 \times 10^{-1}$ (5.4)	7 add.
Fine tuning	$0.41 \times 10^{-2}$ (7.5)	$0.41 \times 10^{-2}$	$0.18 \times 10^{-1}$ (5.8)	6 add.

## A Complete Example (2/2)



## Some Accuracy Results

Target	$c$	$\epsilon_{avg}$	$\sigma$	$\epsilon_{max}$
$w_I = 8$ $d = 2$	3	$0.57 \times 10^{-2}$ (7.4)	$0.41 \times 10^{-2}$	$0.19 \times 10^{-1}$ (5.7)
	4	$0.43 \times 10^{-2}$ (7.8)	$0.30 \times 10^{-2}$	$0.14 \times 10^{-1}$ (6.1)
	5	$0.24 \times 10^{-2}$ (8.7)	$0.17 \times 10^{-2}$	$0.09 \times 10^{-1}$ (6.7)
	6	$0.18 \times 10^{-2}$ (9.1)	$0.12 \times 10^{-2}$	$0.05 \times 10^{-1}$ (7.6)
	all	$0.16 \times 10^{-2}$ (9.3)	$0.11 \times 10^{-2}$	$0.05 \times 10^{-2}$ (7.6)
$w_I = 12$ $d = 3$	1,7	$0.87 \times 10^{-3}$ (10.1)	$0.86 \times 10^{-3}$	$0.51 \times 10^{-2}$ (7.6)
	2,8	$0.61 \times 10^{-3}$ (10.6)	$0.57 \times 10^{-3}$	$0.31 \times 10^{-2}$ (8.3)
	3,10	$0.44 \times 10^{-3}$ (11.1)	$0.38 \times 10^{-3}$	$0.24 \times 10^{-2}$ (8.7)
	4,11	$0.38 \times 10^{-3}$ (11.3)	$0.30 \times 10^{-3}$	$0.16 \times 10^{-2}$ (9.2)
	5,12	$0.17 \times 10^{-3}$ (12.5)	$0.13 \times 10^{-3}$	$0.08 \times 10^{-2}$ (10.2)
	all	$0.08 \times 10^{-3}$ (13.5)	$0.06 \times 10^{-3}$	$0.03 \times 10^{-2}$ (11.7)

Accuracy results for  $\sin(x)$  on  $[0, \pi/4[$ ,  $w_I = w_O = k$ ,  $NZ = 3, g = 2$

## Some Accuracy Results

$f$	$d$	$w_I$	$c$	$\epsilon_{avg}$	$\sigma$	$\epsilon_{max}$
$1/x$	2	8	5	$0.39 \times 10^{-2}$ (8.0)	$0.31 \times 10^{-2}$	$0.24 \times 10^{-1}$ (5.4)
	$[1, 2[$	3	12, 9, 9	$0.10 \times 10^{-2}$ (9.9)	$0.78 \times 10^{-3}$	$0.36 \times 10^{-2}$ (8.1)
$2^x$	2	8	6	$0.37 \times 10^{-2}$ (8.0)	$0.27 \times 10^{-2}$	$0.11 \times 10^{-1}$ (6.5)
	$[0, 1[$	3	12, 6, 6	$0.68 \times 10^{-2}$ (7.2)	$0. \times 36^{-2}$	$0. \times 21^{-1}$ (5.5)
		3*	12, 6, 6	$0.19 \times 10^{-2}$ (9.0)	$0. \times 15^{-2}$	$0. \times 08^{-1}$ (6.9)
$\sqrt{x}$	2	8	5	$0.15 \times 10^{-2}$ (9.3)	$0.11 \times 10^{-2}$	$0.52 \times 10^{-2}$ (7.5)
	$[1, 2[$	3	12, 7, 7	$0.21 \times 10^{-3}$ (12.2)	$0.16 \times 10^{-3}$	$0.98 \times 10^{-3}$ (9.9)
$1/\sqrt{x}$	2	8	5	$0.32 \times 10^{-2}$ (8.3)	$0.22 \times 10^{-2}$	$0.11 \times 10^{-1}$ (6.5)
	$[1, 2[$	3	12, 7, 7	$0.89 \times 10^{-3}$ (10.1)	$0.77 \times 10^{-3}$	$0.40 \times 10^{-2}$ (7.9)

## FPGA Implementations

Version	$d$	2			3		
	$w_I = w_O$	8	12	16	8	12	16
Generic	Area [# slices]	115	235	411	324	895	1886
	Period [ns]	24.8	35.9	37.9	38.6	54.6	50.4
Constant	Area [# slices]	43	107	192	204	697	1570
	Period [ns]	15.2	27.9	29.9	33.9	52.3	59.0
Quantified	Area [# slices]	43	85	136	202	651	1475
	Period [ns]	14.3	21.3	22.7	26.8	35.5	38.5

FPGA implementation results for generic polynomials (using multipliers) for  $\sin(x)$  on  $[0, \pi/4[$

## FPGA Implementations

Version	$d$	2			3		
	$w_I = w_O$	8	12	16	8	12	16
$c = 6$	Area [# slices]	34	50	61	56	78	99
	Period [ns]	15.1	18.7	19.5	21.5	23.7	22.9
$c = 4$	Area [# slices]	28	46	57	38	61	82
	Period [ns]	14.8	18.6	18.9	18.3	20.1	22.1
$c = 3$	Area [# slices]	27	45	56	28	50	71
	Period [ns]	14.9	18.2	18.5	18.2	20.5	23.5

FPGA implementation results for polynomials with 3-bit coefficients and estimations of the powers of  $x$  for  $\sin(x)$  on  $[0, \pi/4[$ .

## FPGA Implementations

$f$	$\sin(x)$ on $[0, \pi/4[$		$\sqrt{x}$ on $[1, 2[$	
Degree	2	3	2	3
$w_I = w_O$	8	12	8	12
$c$	5	3, 10	5	7, 7
Area [# slices]	27	141	17	86
Period [ns]	16.7	28.5	17.2	29.4

FPGA implementation results for other parameters or functions

## Comparison with Previous Work

$w_I$	Multipartite [1]		SMSO [2]		Our		
	Area [# slices]	Period [ns]	Area [# slices]	Period [ns]	Area [# slices]	Period [ns]	$d$
8	19	16.6	21	8	27	14.9	2
12	76	18.0	63	14	50	20.5	3
16	280	24.8	123	19	71	23.5	3

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

[1] F. de Dinechin and A. Tisserand. Some improvements on multipartite tables methods. *IEEE Transactions on Computers*, 54(3):319–330, Mar. 2005

[2] J. Detrey and F. de Dinechin. Second order function approximation using a single multiplication on FPGAs. *14th Conf. Field-Programmable Logic and Applications*, pp. 221–230, Aug. 2004

## Future Prospects (for this generator)

- ASIC targets
- study of the relations between the numerous parameters
- finalize the automatic generator
- improve accuracy (up to 24-bit words)

## This is the end. . .

Questions ?

Contacts:

- [arnaud.tisserand@ens-lyon.fr](mailto:arnaud.tisserand@ens-lyon.fr)
- <http://perso.ens-lyon.fr/arnaud.tisserand/>
- LIP, ENS Lyon. 46 allée d'Italie. F-69364 Lyon cedex 07. France.

Thank you.