

EUDOXUS: A WWW-based Generator of Reusable Arithmetic Cores

**D. Bakalis, K. D. Adaos, D. Lympelopoulos,
G. Ph. Alexiou and D. Nikolos**

*Dept. of Computer Engineering & Informatics,
University of Patras, 26500 Rio, Patras, Greece*

*Computer Technology Institute
3 Kolokotroni Str., 26221 Patras, Greece*

MOTIVATION

- ❖ ***Silicon technology* now allows to build chips consisting of tens of millions of transistors**
- ❖ **It also promises new levels of system integration on a single-chip but presents significant challenges to the chip designer**

MOTIVATION

- ❖ ***System-on-a-Chip (SoC)* design creates several problems that have to be addressed:**
 - ❖ **time-to-market goals demand rapid prototyping**
 - ❖ **increased complexity**
 - ❖ **results must be competitive in terms of performance, area and power**
 - ❖ **the development teams consist of many designers with different levels of expertise**

MOTIVATION

- ❖ **Design automation has increased productivity while a reuse strategy has become a means of amortizing the development effort across multiple projects**
- ❖ **Many design teams are turning to a block-based design approach which emphasizes *design reuse*, that is the use of pre-designed and pre-verified blocks**
- ❖ **An effective block-based design methodology requires an extensive library of reusable blocks or macros**

MOTIVATION

- ❖ ***Hard macros*** are firm design blocks already targeted to a specific implementation technology
 - ❖ impose restrictions to the designer
- ❖ ***Soft macros*** are provided as a RTL-level HDL description
 - ❖ give the ability to the designer to map them to any ASIC or FPGA implementation technology via logic synthesis

MOTIVATION

- ❖ ***Network-based* apps and design tools are becoming significantly more popular than their stand-alone desktop counterparts**

- ❖ **Advantages**
 - ❖ **single platform development**
 - ❖ **minimum installation costs**
 - ❖ **updates and bug fixes immediately available to users**
 - ❖ **better authorization**
 - ❖ **lower-performance desktop computers can be utilized**

MOTIVATION

- ❖ ***Arithmetic modules* are common to almost every system design**
- ❖ **Creating a library or a generator of such modules provides the system designer with efficient and versatile building blocks for developing SoC solutions**

OUTLINE OF THE PRESENTATION

- ❖ **Requirements**
- ❖ **Specifications of the core generator**
- ❖ **The implemented algorithms for arithmetic operations**
- ❖ **Implementation issues**
- ❖ **Results**

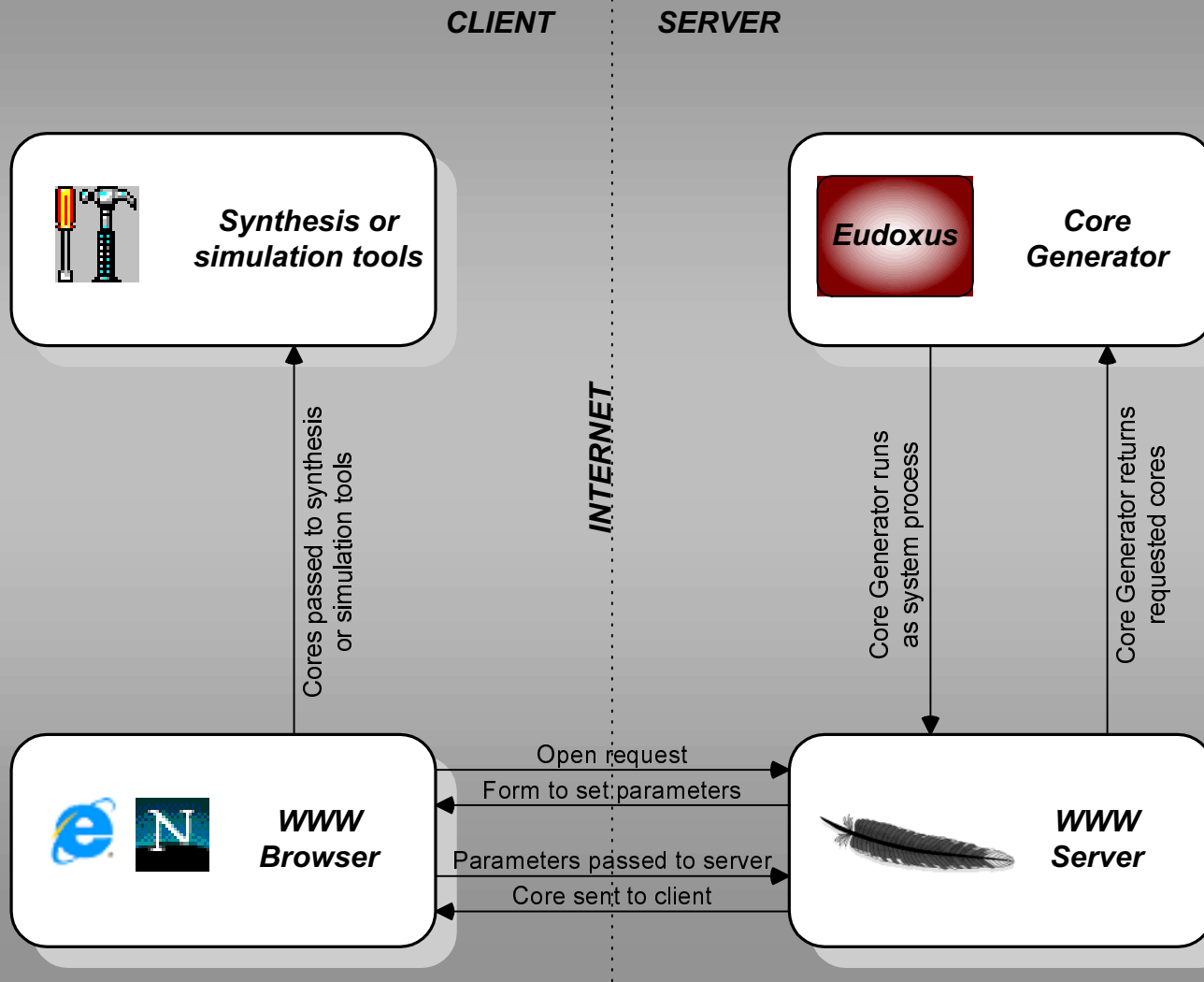
REQUIREMENTS

- ❖ ***Technology independence:*** it should not restrict to specific gates or technology libraries
- ❖ ***Easy to integrate:*** the modules generated must be easily integrated with the rest of the design
- ❖ ***Many alternatives:*** the generator must implement several algorithms for several arithmetic operations
- ❖ ***No restrictions on the operands size***
- ❖ ***Applicability:*** the generator must at-least produce the most frequently used arithmetic modules
- ❖ ***Easy-to-use***
- ❖ ***Expandability***

SPECIFICATIONS

- ❖ **Technology independence**
- ❖ **Easy to integrate**
- ❖ **Many alternatives**
- ❖ **No restrictions on the operand size**
- ❖ **Applicability**
- ❖ **Easy-to-use**
- ❖ **Expandability**
- ✓ **Structural description in Verilog and VHDL**
- ✓ **Implements the most frequently-used algorithms for several arithmetic operations (+, -, *, /, >, ², √)**
- ✓ **Software-tool with www-based interface**
- ✓ **Open architecture in C programming language**

A BLOCK DIAGRAM OF THE GENERATOR



OPERATION OF THE GENERATOR

Type: Modified Booth Multiplier

Description: A parallel multiplier for 2's complement operands. It uses 2-bit Booth encoding, carry-save adders for adding the partial products and a ripple carry adder for producing the final product.

Supported Sizes: 2, 4, 6, ..., 254, 256

Supported Languages: VHDL, Verilog

Example: An 8-bit Modified Booth Multiplier

Modified Booth Multiplier [\(i\)](#)

Serial Parallel Multiplier [\(i\)](#)

Dividers

Restoring Cellular Array Divider [\(i\)](#)

Non-Restoring Cellular Array Divider [\(i\)](#)

Square Root

Non-Restoring Fractional Square Rooter [\(i\)](#)

Squarer

Non-Restoring Fractional Squarer [\(i\)](#)

Shifters

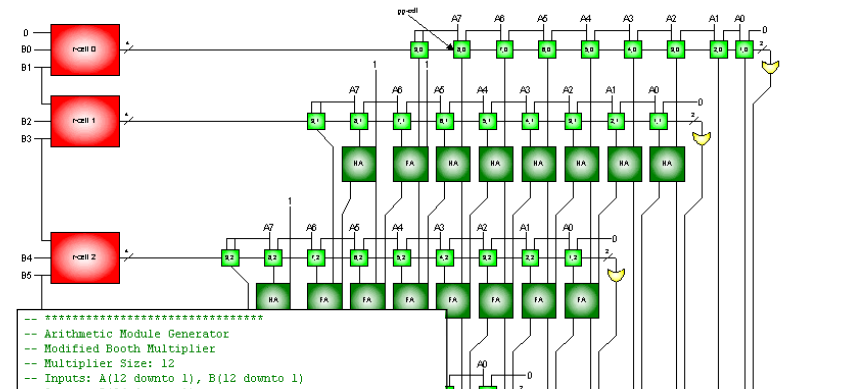
Multiplexer Based Shifter [\(i\)](#)

2. Choose Circuit Size:

3. Choose HDL Language: VHDL Verilog

Create Description

Reset



```

-- *****
-- Arithmetic Module Generator
-- Modified Booth Multiplier
-- Multiplier Size: 12
-- Inputs: A(12 downto 1), B(12 downto 1)
-- Output: P(24 downto 1)
-- *****

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity full_adder is
    port (
        i1 : in std_logic;
        i2 : in std_logic;
        c_in : in std_logic;

        o : out std_logic;
        c_out : out std_logic
    );
end full_adder;

architecture structure of full_adder is
    signal temp : std_logic;
begin
    temp <= i1 xor i2;
    o <= temp xor c_in;
    c_out <= (i1 and i2) or (i1 and c_in) or (i2 and c_in);
end structure;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity half_adder is
    port (

```

CORE DESCRIPTION

Each core description consists of 4 distinct parts:

- ❖ a core information part**
- ❖ the basic cell definition part**
- ❖ the core description**
- ❖ a test-bench for functional verification**

THE IMPLEMENTED ALGORITHMS

- ❖ **Parallel arithmetic cores**
- ❖ **Serial/Parallel arithmetic cores**
- ❖ **Serial arithmetic cores**
- ❖ **Pseudorandom testing cores**

PARALLEL ARITHMETIC CORES (I)

Adder/Subtractor Cores

- ❖ **adders/subtractors with Ripple Carry propagation**
- ❖ **group Carry Look-Ahead adders/subtractors with ripple carry propagation between groups**
- ❖ **Multiple-Level Carry Look-Ahead adders**
- ❖ **Sklansky parallel-prefix adders**
- ❖ **Kogge-Stone parallel-prefix adders**

PARALLEL ARITHMETIC CORES (II)

Unsigned multiplier cores

- ❖ **Carry Propagate Array Multipliers**
- ❖ **Carry Save Array Multipliers**

2's complement multiplier cores

- ❖ **TriSection Pezaris array multipliers**
- ❖ **Baugh-Wooley Array multipliers**
- ❖ **Booth-encoded multipliers**

PARALLEL ARITHMETIC CORES (III)

Divider Cores

- ❖ **Non-Restoring Cellular Array Dividers**
- ❖ **Restoring Cellular Array Dividers**

Square Rooter Cores

- ❖ **Non-Restoring Fractional Square Rooter**

Squarer Cores

- ❖ **Non-Restoring Fractional Squarer**

Shifter Cores

- ❖ **Multiplexer-based Shifter**

SERIAL/PARALLEL ARITHMETIC CORES

Multiplier Cores

- ❖ **Unsigned serial/parallel multiplier**
- ❖ **2's complement serial/parallel multiplier**

SERIAL ARITHMETIC CORES

Adder Cores

- ❖ **Serial Adder**
- ❖ **Serial Column Adder**

Multiplier Cores

- ❖ **Serial Multiplier**

Squarer Cores

- ❖ **Serial Squarer**

Complementer Cores

- ❖ **1's/2's complementer**

PSEUDORANDOM TESTING CORES

Test Pattern Generator Cores

- ❖ **Linear Feedback Shift Register Cores (LFSRs)**

Test Response Analyzer Cores

- ❖ **Multiple-Input Linear Feedback Shift Register Cores (MISRs)**

CORES AND OPERAND SIZES (I)

		Minimum size	Step	Maximum Size
<i>Parallel Cores</i>				
<i>Adders/Subtractors</i>	Ripple Carry Adder/Subtractor	2	1	1024
	Group Carry Look-Ahead Adder/Subtractor	4	4	1024
	Multi-Level Carry Look-Ahead Adder	4	4	256
	Sklansky Parallel-Prefix Adder	2	1	1024
	Kogge-Stone Parallel-Prefix Adder	2	1	1024
<i>Multipliers</i>	Carry Propagate Array Multiplier	2	1	256
	Carry Save Array Multiplier	2	1	256
	TriSection Pezaris Array Multiplier	2	1	256
	Baugh-Wooley Array Multiplier	2	1	256
	Modified Booth Multiplier	2	2	256
<i>Dividers</i>	Restoring Cellular Array Dividers	8	power of 2	64
	Non-Restoring Cellular Array Dividers	8	power of 2	64
<i>Square Rooters</i>	Non-Restoring Fractional Square Rooter	4	2	256
<i>Squarers</i>	Non-Restoring Fractional Squarer	4	2	256
<i>Shifters</i>	Multiplexer-based Shifter	2	1	256

CORES AND OPERAND SIZES (II)

		Minimum size	Step	Maximum Size
<i>Serial/Parallel Cores</i>				
<i>Multipliers</i>	Unsigned Serial/Parallel Multiplier	2	1	256
	2's complement Serial/Parallel Multiplier	2	1	256
<i>Serial Cores</i>				
<i>Adders</i>	Serial adder	2	1	1024
	Serial Column Adder	4	2	1024
<i>Multipliers</i>	Serial multiplier	4	2	1024
<i>Squarers</i>	Serial squarer	4	2	1024
<i>Complementers</i>	Serial 1's/2's complement	2	1	1024
<i>Testing Structures</i>				
<i>Pattern Generators</i>	Linear Feedback Shift Registers (LFSRs)	2	1	256
<i>Response Analyzers</i>	Multiple Input Linear Feedback Shift Register (MISR)	2	1	256

FUNCTIONAL VERIFICATION OF THE CORES

- ❖ **Exhaustive simulation has been carried out for small sizes**
- ❖ **Simulation with pseudorandom input values has been carried out for larger sizes**
- ❖ **The regularity of the circuit descriptions ensures correct functionality also for larger sizes**

LOGIC SYNTHESIS RESULTS (I)

	Synopsys		Mentor Graphics		Altera		Xilinx	
	<i>Area (mils)</i>	<i>Delay (ns)</i>	<i>Area (eq.gate)</i>	<i>Delay (ns)</i>	<i>Area (cells)</i>	<i>Delay (ns)</i>	<i>Area (LUTs)</i>	<i>Delay (ns)</i>
14-bit Carry Propagate Multiplier	1574	34.5	4590	33.3	401	99.4	415	84.2
14-bit Carry Save Multiplier	1579	24.3	4621	22.6	426	63.4	443	53.0
14-bit Baugh-Wooley Multiplier	1531	26.1	4632	29.0	386	67.7	560	53.6
14-bit Booth Multiplier	1341	19.1	4303	16.9	397	49.7	407	49.3
16/8-bit Restoring Array Divider	640	78.7	1461	42.6	212	103.5	210	74.3
16/8-bit Non-Restoring Array Divider	556	50.3	1159	39.6	188	105.5	210	80.8

LOGIC SYNTHESIS RESULTS (II)

	Altera		Xilinx	
	<i>Area (cells)</i>	<i>Speed (MHz)</i>	<i>Area (LUTs)</i>	<i>Speed (MHz)</i>
14-bit Unsigned Serial/Parallel Multiplier	42	250	42	257
14-bit Signed Serial/parallel Multiplier	71	250	74	249
14-bit Serial Multiplier	164	138.88	157	125.597
14-bit Serial Squarer	105	192.30	91	170.412
64 input Column Adder	126	250	126	198

CONCLUSIONS

The presented core generator:

- ❖ **produces synthesizable soft core descriptions in structural Verilog and/or VHDL**
- ❖ **covers several arithmetic operations: addition, subtraction, multiplication, division, squaring, square rooting and shifting**
- ❖ **produces some pseudorandom testing structures (LFSRs, MISRs)**
- ❖ **is capable of producing more than one different architectures for many operations**
- ❖ **The use of the tool gives the ability to the designer to select in the smallest possible time, between alternative designs, the one that best meets his design goals**