

ASIC Implementation and Analysis of Extrinsic EHW Based Power and Area Optimised 8-Bit Asynchronous Parallel MAC

D.Dhanasekaran, and **Dr.K.Boopathy Bagan

*Assistant Professor, SVCE, Pennalur,Sriperumbudur-602105.

**Professor, Madras Institute of Technology, Chrompet, Chennai-44

ABSTRACT

In computing, especially in digital signal processing, multiply-accumulate is a common operation that computes the product of two numbers and adds that product to an accumulator.

The VERILOG code for MAC operation is simulated and synthesized in Vendors tool like XILINX ISE and ALTERA QUARTUS II with different devices. Once the coding is error free the schematic for MAC unit will be generated and the bit stream to used to download is used as reference to get the approximately the same circuit by the application of extrinsic EHW using evolutionary algorithm. This schematic obtained by EHW is realized as ASIC using Microwind to get the layout determining the area and power requirement. Depending on the different reports obtained, the optimized device which requires minimum area and less power consumption is identified. This can be applied to any devices like FPGA or CPLD of any vendor. Evolvable hardware (EHW) has attracted increasing attention since the early 1990's with the advent of easily reconfigurable hardware, such as field programmable gate arrays(FPGA's). It promises to provide an entirely new approach to complex electronic circuit design and new adaptive hardware. EHW has been demonstrated to be able to perform a wide range of tasks from pattern recognition to adaptive control. However, there are still many fundamental issues in EHW that remain open. In this paper it was more concentrated on the ASIC part rather than EHW because EXTINSIC EHW was used.

1. INTRODUCTION

In computing, especially in digital signal processing, multiply-accumulate is a common operation that computes the product of two numbers and adds that product to an accumulator.

$$A \leftarrow A + B \times C$$

Modern computers may contain a dedicated multiply-add unit, or "MAC-unit", consisting of a multiplier implemented in combinational logic followed by an adder and an accumulator register which stores the result when clocked. The output of the register is fed back to one input of the adder, so that on each clock the output of the multiplier is added to the register. Combinational multipliers require a large amount of logic, but can compute a product much more quickly than the method of shifting and adding typical of earlier computers.

2 .MAC ALGORITHM

This application is simply the internals for a matrix multiply. The application is setup to use a fully pipelined 32 bit integer multiplier, and a variable summation part. This particular application was only designed with two instances on the chip. The figure below shows the components used in this application. In order to get two multiply accumulates onto the FPGA the multiplier had to be redesigned from its original form to compensate for the fragmentation issue. As shown in the figure below the data flow for the first instance of the multiply accumulate goes from left to right, and the second instance flows right to left. The multipliers were too large to fit on top of one another or next to one another. Next, two instances of multiplier were built,

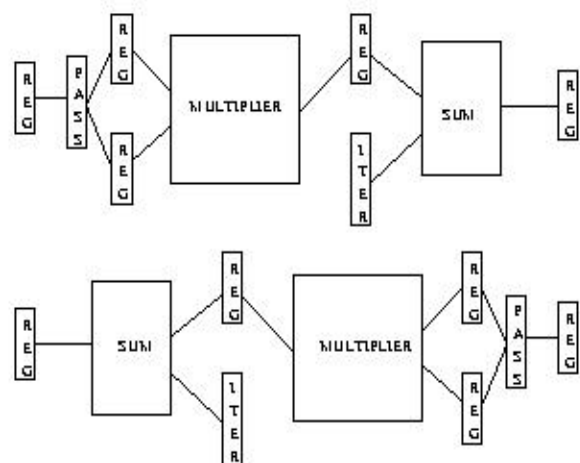


Fig-1.1 MAC Algorithms

one with data flow left to right and the other with data flow right to left. The adders, registers and summation parts do not require redesigning since the computational parts of the components are only one column wide.

3. PARALLEL MAC

A parallel multiply accumulate array circuit, comprising:

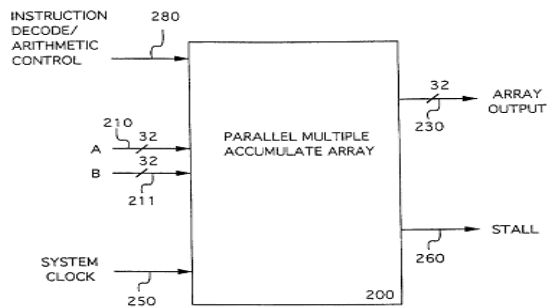


Fig-1.2 Parallel MAC

A plurality of n multipliers each coupled to receive a first x -bit operand and a second x -bit operand and generating a $2x$ -bit product; a first multiplexer having n inputs coupled to receive n $2x$ -bit products from said plurality of n multipliers and providing one $2x$ -bit product output; a downshift circuit coupled to receive said one $2x$ -bit product output, said downshift circuit for downshifting y bits of said one $2x$ -bit product output; a plurality of m accumulators each having an input coupled to receive a downshifted output from said downshift circuit, each of said accumulators for accumulating a separate summed value; and a second multiplexer including m inputs each coupled to receive summed values from one of said plurality of m accumulators and also having an output for supplying one of said summed values. The circuit of claim 1 further comprising an internal control circuit for controlling: A select bus of said first multiplexer; a downshift adjust bus of said downshift circuit; and an enable bus of said plurality of m accumulators. The circuit of claim 1 wherein each of said accumulators comprise an adder and a register. The circuit of claim 1 wherein a delay register is coupled between said downshift circuit and said plurality of m accumulators. The circuit of claim 1 wherein each of said plurality of n multipliers is a booth multiplier. The circuit of claim 5 wherein a P register of each booth multiplier is primed with a value (11 bits) so that said downshift circuit rounds. The circuit of claim 1 wherein $n=6$, $m=4$, $x=32$, and $y=12$. The circuit of claim 1 wherein $n=3$, $m=3$, $x=32$, and y is programmable for each clock cycle of said circuit of claim 1. The circuit of claim 1 wherein each of said plurality of n multipliers contains memory for storing data for downshift values and accumulate enable information associated with said

first x -bit value and said second x -bit value. Within a graphics card of a host computer system, a parallel multiply accumulate array circuit, comprising: A plurality of n multipliers each coupled to receive a first x -bit operand and a second x -bit operand and generating a $2x$ -bit product; a first multiplexer having n inputs coupled to receive n $2x$ -bit products from said plurality of n multipliers and providing one $2x$ -bit output; a downshift circuit coupled to receive said one $2x$ -bit output of said first multiplexer, said downshift circuit for downshifting y bits of said one $2x$ -bit output; a plurality of m accumulators each having an input coupled to receive a downshifted output from said downshift circuit, each of said accumulators for accumulating a separate summed value; and a second multiplexer including m inputs each coupled to receive summed values from one of said plurality of m accumulators and also having an output for supplying one of said summed values.

4. POWER OPTIMISATION

An apparatus for performing multiplications with reduced power includes an arithmetic logic unit and a decode block for performing an equivalent of a multiply instruction. A frequently-encountered multiply instruction occurs between a variable and a known constant. If the known constant is positive or negative one, the decode block enables the arithmetic logic unit to either add the variable to zero, or subtract the variable from zero, in response to the sign bit of the known constant. In response to a multiply and accumulate instruction between a variable and a known constant of positive or negative one, the decode block enables the arithmetic logic unit to either add the variable to the prior accumulated result or to subtract it there from, in response to the sign bit of the known constant. In either case, the high-speed multiplier is disabled and its power saved.

5. FPGA:

Before the advent of programmable logic, custom logic circuits were built at the board level using standard components, or at the gate level in expensive application-specific (custom) integrated circuits. The FPGA is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. Each logic cell can independently take on any one of a limited set of personalities. The individual cells are interconnected by a matrix of wires and programmable switches. A user's design is implemented by specifying the simple logic function for each cell and selectively closing the switches in the interconnect matrix. The array of logic cells and interconnect form a fabric of basic

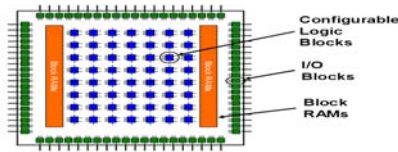


Fig-1.3 FPGA

building blocks for logic circuits. Complex designs are created by combining these basic blocks to create the desired circuit. A field-programmable gate array is a semiconductor device containing programmable logic components called "logic blocks", and programmable interconnects. Logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or simple mathematical functions. In most FPGAs, the logic blocks also include memory elements, which may be simple

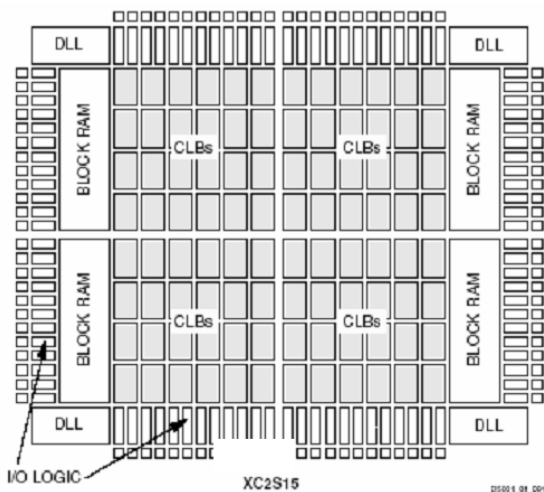


Fig-1.4 FPGA block diagram

flip-flops or more complete blocks of memories. A hierarchy of programmable interconnects allows logic blocks to be interconnected as needed by the system designer, somewhat like a one-chip programmable breadboard. Logic blocks and interconnects can be programmed by the customer or designer, after the FPGA is manufactured, to implement any logical function, hence the name "field-programmable". FPGAs are usually slower than their application-specific integrated circuit (ASIC) counterparts, as they cannot handle as complex a design, and draw more power. But their advantages include a shorter time to market, ability to re-program in the field to fix bugs, and lower non-recurring engineering costs. Vendors can sell cheaper, less flexible versions of their

FPGAs which cannot be modified after the design is committed. The designs are developed on regular FPGAs and then migrated into a fixed version that more resembles an ASIC. Another alternative are complex programmable logic devices (CPLDs).

6.FLEXIBILITY

eASIC's Structured ASIC technology combines the advantages of FPGA technology with those of Standard Cell ASICs by adopting the best features of each approach and avoiding their drawbacks. Thus, on one end, eASIC adopted the way FPGAs program logic while avoiding their inefficient approach to interconnect routing. On the other end, eASIC adopted the Standard Cell approach toward interconnect routing while avoiding the expense of its rigid approach to logic definition.










Fig-1.5 FPGA flexibility

7. STANDARD CELL CHALLENGES

In today's deep-submicron reality, when NRE costs are skyrocketing, turnaround times are getting longer and yield is a major challenge, Standard Cell ASICs are losing ground as the preferred ASIC solution. Unless very high volume is certain and very high performance is required, design projects based on Standard Cell technology can not be justified. It is not surprising, then, that the number of ASIC design starts decreased drastically from about 10,000 in 1998 to about 3,500 in 2002 and less than 2000 in 2004. As process technology advances and mask costs continue to increase exponentially, these issues will only become worse and Standard Cell design starts will decrease accordingly.

8.FPGA CHALLENGES

Targeted toward the low end of the market, FPGAs have the advantage of no upfront cost and of design flexibility and re-programmability. But the major disadvantages of FPGA technology stem from their programmable interconnect approach, as described below. Programmable interconnect relies on either pass transistors or active buffers controlled by SRAM cells. Pass transistors add a high resistance to the route, resulting in high and hard to estimate propagation delay. Active buffers, on the other hand, impose an extreme area penalty when deployed in sufficient quantity to create a well-connected routing network. Furthermore, interconnect delay becomes proportionally worse with each process

Cell Type	Colour
Buffer	
Configuration SRAM	
Multiplexer	
LUT	
Flip-Flop	
Pass Transistor Switch	
Buffered Switch	

shrink, intensifying the penalty imposed by programmable interconnect as technology progresses. In order to cope with this problem, the semiconductor industry embraced the solution of adding more metal layers. For Standard Cell ASICs this is a practical technique, but for FPGAs this solution has a major overhead. The additional metal layers need to be programmably connected, requiring diffusion-layer resources and decreasing the effective logic density of the device.

9. eASIC ADVANTAGE

eASIC's LUT-based logic and regular routing grid are able to defeat the Standard Cell design issues presented above. Direct-Write eBeam technology and multi-design wafers allow eASIC to offer an NRE-free cost model, avoiding the high fixed costs associated with Standard Cell mask production. Because wafers can be

pre-manufactured through the Metal 6 layer, turnaround times are drastically shorter. The eASIC fabric's regular structure reduces yield and reliability issues; each time these issues are solved within one portion of the fabric, the solution is then applied across the entire fabric. Moreover, because the same fabric is used for every customer design, eASIC is able to solve yield and reliability issues a single time for all its customers. Finally, the use of coarse-grained LUT-based logic helps to resolve the interconnect delay issues that are intensifying at 90nm processes and below. By combining LUT-based logic with metal-configurable routing, eASIC is able to overcome FPGA challenges as well. As depicted in the diagram below, the majority of the diffusion area in an FPGA tile is consumed by programmable routing buffers and their associated SRAM. Because eASIC's metal-configurable routing grid exists entirely in the higher metal layers and requires no diffusion-layer resources, eASIC enjoys a 25:1 density advantage over FPGAs. This is the primary driver of eASIC's unit cost advantage over FPGA technology. eASIC Fabric Density is ~25x of FPGA (Up to 95% of FPGA silicon is spent on programmable interconnect)

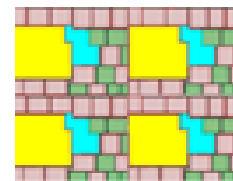


Fig-1.6 4x LUT4 Architecture Tile in eASIC Fabric
1,600 μ^2 @ 0.18 μ

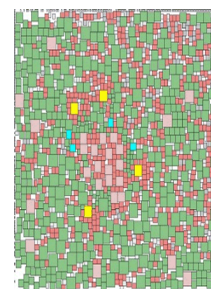


Fig-1.7 4X LUT4 Architecture Tile in FPGA* 35,462 μ^2
@ 0.18 μ

*Source: "Automatic Transistor and Physical Design of FPGA Tiles from an Architectural Specification" - K.Padalia, Jonathan Rose, et al.- FPGA2003 Conference

10. WORKING PRINCIPLE

Look-up table (LUT) with N-input can be used to implement any combinational function of N-inputs. LUT is programmed with the truth table.

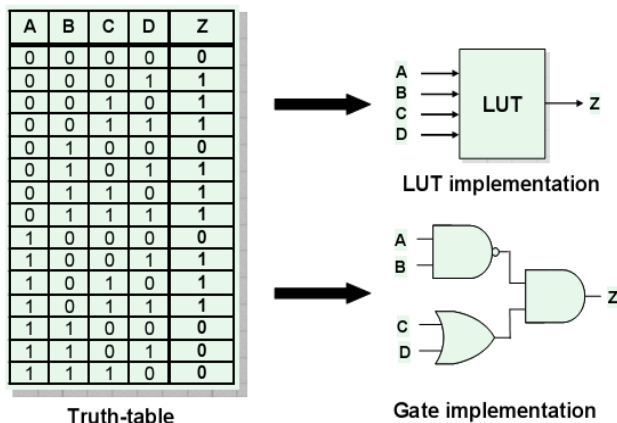


Fig-1.8 Working principle (LUT)

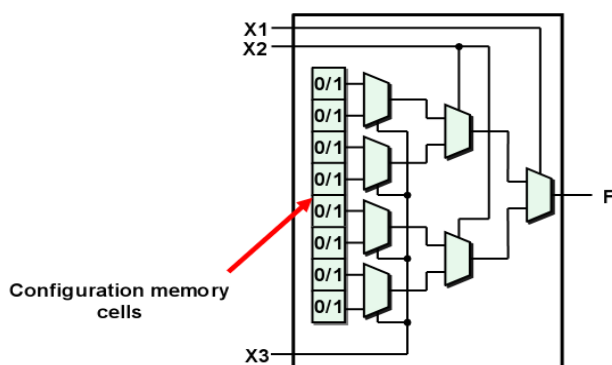


Fig-1.9 3 input LUT

11. MANUFACTURERS AND THEIR SPECIALTIES

As of late 2005, the FPGA market has mostly settled into a state where there are two major "general-purpose" FPGA manufacturers and a number of other players who differentiate themselves by offering unique capabilities. **Xilinx** and **Altera** are the current FPGA market leaders. Xilinx also provide free Linux design software. **Lattice Semiconductor** provides both SRAM and non-volatile, flash-based FPGAs. **Actel** has antifuse and reprogrammable flash-based FPGAs, and also offers mixed signal flash-based FPGAs. **Atmel** provides fine-grain reconfigurable devices, as the Xilinx XC62xx were. They focus on providing Atmel AVR Microcontrollers with FPGA fabric on the same die. **QuickLogic** has antifuse (programmable-only-once) products and heavily

focused on military applications. **Achronix Semiconductor** has very fast FPGAs in development, focusing on speeds approaching 2 GHz. **MathStar** offers an FPGA-like device called an FPOA (field programmable object array).

12. ADVANTAGES

Once used only for glue logic, FPGAs have progressed to a point where system-on-chip (SoC) designs can be built on a single device. The number of that have traditionally been offered through ASIC devices only. This article addresses some of the advantages of FPGA design methodologies over ASICs, including early time-to-market, easy transition to structured ASICs, and reduced NRE costs. As FPGA devices progressed both in terms of resources and performance, the latest FPGAs have come to provide "platform" solutions that are easily customizable for system connectivity, DSP, and/or data processing applications. As platform solutions are becoming more and more important, leading FPGA vendors are coming up with easy-to-use design development tools. These platform building tools accelerate time-to-market by automating the system definition and integration phases of system on programmable chip (SOPC) development. The tools not only improve design productivity, but also reduce the cost of buying these tools from 3rd party EDA vendors. Using such tools, system designers can define a complete system, from hardware to software, within one tool and in a fraction of the time of traditional system-on-a-chip (SOC) design.

13. ASIC DESIGN

An application-specific integrated circuit (ASIC) is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. For example, a chip designed solely to run a cell phone is an ASIC.

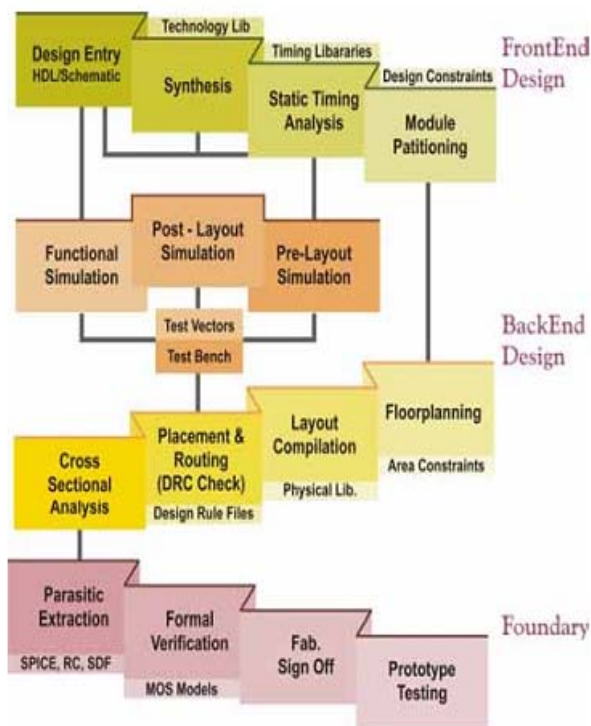


Fig-4.1 ASIC design flow

HardCopy Design Flow	Advantages
Full in-system verification using an FPGA	<ul style="list-style-type: none"> • Reduce risk of design re-spin • Enable early software co-design • Lowest total development cost
FPGA-like, front-end design flow	<ul style="list-style-type: none"> • Low-cost design environment compared to other structured ASICs or standard-cell ASIC design flows • Minimal design tools and methodology learning curve
Seamless migration	<ul style="list-style-type: none"> • No board re-spin needed because of the same intellectual property (IP) and pin-out for both FPGA and HardCopy devices • Flexible production choice using either FPGA or HardCopy devices, depending on volume and product life • Enable fast time-to-market using an FPGA for early production

Table 4.1 Design flow advantages

14. SWITCHING FROM ASIC DESIGN

Although the underlying structure of FPGAs is different than ASICs, There are softwares that provide methodologies and features that enable ASIC designers to successfully design for structured ASICs with high performance and productivity. As FPGAs have evolved to become closer in application space to ASICs, FPGA design flows have become fundamentally similar to ASIC design flows. Also, these softwares offer some innovative technologies to speed system design and take advantage of the programmable nature of FPGAs for in-system verification.

15. HIERARCHICAL DESIGN

To support ASIC designers, software supports the LogicLock block-based design methodology, which is similar to the block-based design flows used in ASIC design flows. Using the LogicLock methodology, you can partition a design into several functional blocks and assign them to individual team members for independent design, optimization, and implementation. These blocks can then be imported into a top-level system design while maintaining design performance of the individual blocks. Optimized blocks may be reused in subsequent projects with the same performance

4.5 ASIC Vs FPGA DESIGN FLOW

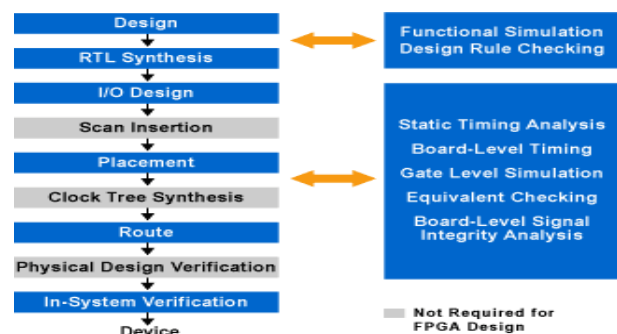


Fig-4.2 . FPGA and ASIC Design Flows Fundamentally Similar

16. ADVANTAGES

There are many reasons for choosing an ASIC-based solution over discrete components: Overall function and performance often can be much better than the corresponding discrete solution, especially concerning power consumption. The product's weight and size can often be reduced considerably. For portable products both of these parameters are important, and sometimes crucial, advantages. ASIC solutions often involve a clearly better

product economy. Thanks to small size and packaging, an ASIC gives automatic intellectual property protection. Minimizing the total number of design components leads to very high reliability, that is, low error frequency. Also, an ASIC is rigorously tested before delivery. This all results in considerably lower maintenance costs.

17. ARCHITECTURE OF MICROWIND AND DSCH

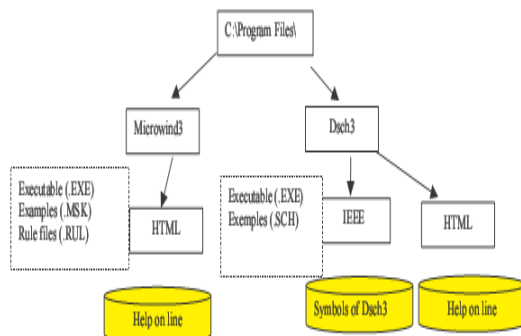


Fig-5.1 Architecture of Microwind and DSCH

18. DSCH-SCHEMATIC EDITOR AND SIMULATOR

18.1 INTRODUCTION

DSCH3 is the companion software for logic design. Based on primitives, a hierarchical circuit is built and simulated. Interactive symbols are used to friendly simulation, which includes delay and power consumption evaluation.

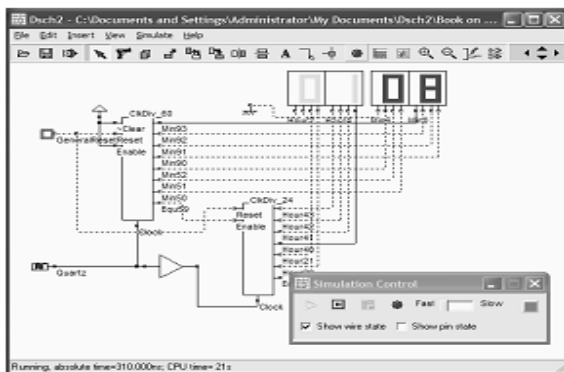


Fig-5.2 DSCH3

18.2 FEATURES

User-friendly environment for rapid design of logic circuits. Handles both conventional pattern-based logic simulation and intuitive on-screen mouse-driven simulation. Supports hierarchical logic design. Built-in extractor which generates a SPICE netlist from the schematic diagram (Compatible with PSPICE™ and WinSpice™). Current and power consumption analysis.

Generates a VERILOG description of the schematic for layout editor. Immediate access to symbol properties (Delay, fanout). Models and assembly support for 8051 and PIC 16F84. Sub-micron, deep-submicron, nanoscale technology support. Supported by huge symbol library.

19. SCHEMATICS

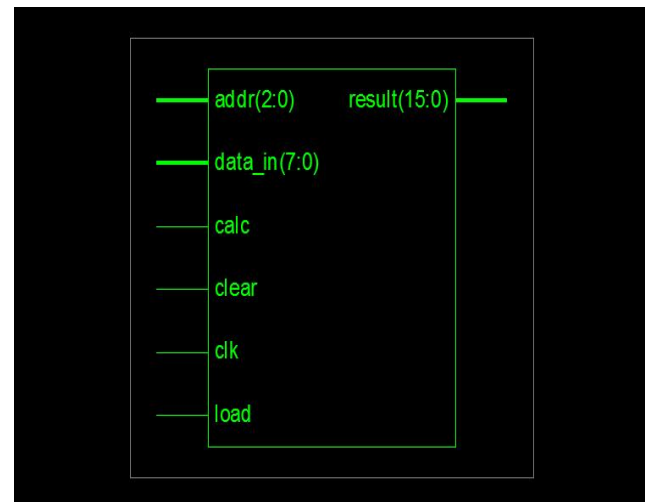


Fig-7.1 Xilinx FPGA basic schematic

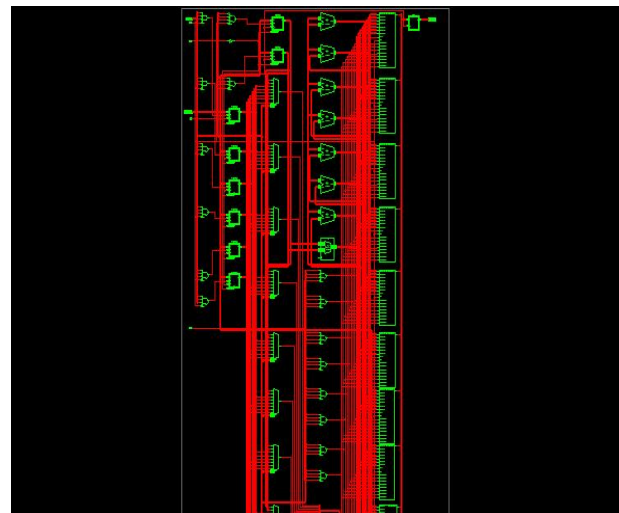


Fig-7.2 Xilinx FPGA detailed schematic

7.2 XILINX CPLD

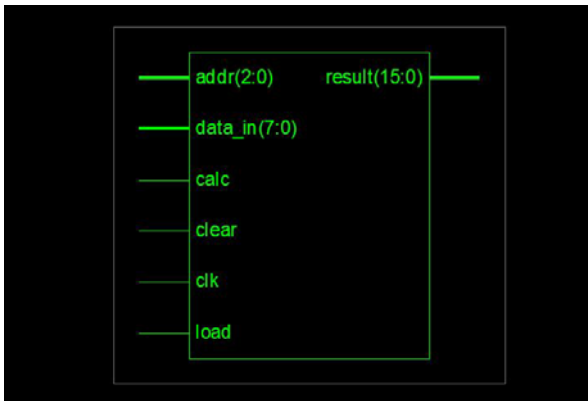


Fig-7.3 Xilinx CPLD basic schematic

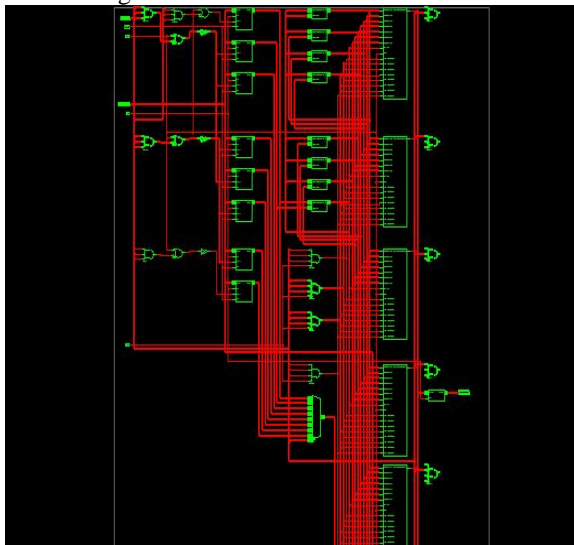


Fig-7.4 Xilinx CPLD detailed schematic

XILINX VIRTEX

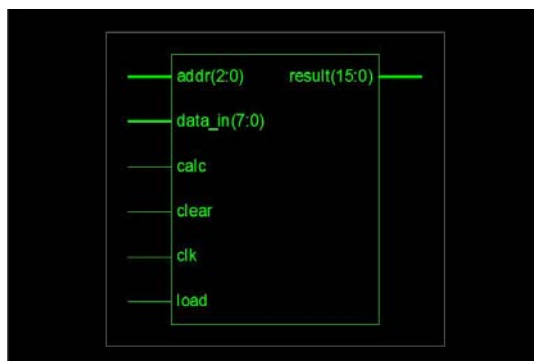


Fig-7.5 Xilinx virtex basic schematic

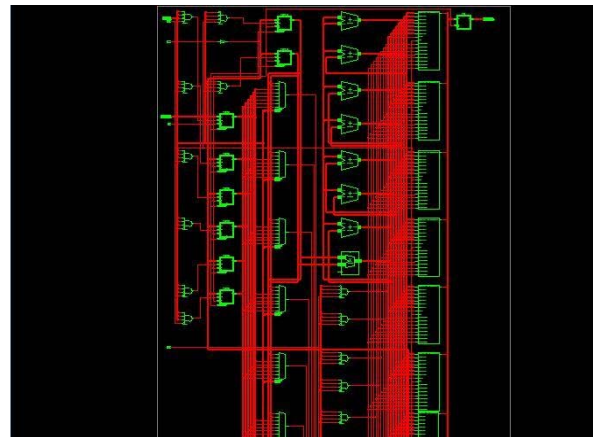


Fig-7.6 Xilinx Virtex detailed schematic

20. LAYOUTS

8.1 TWO INPUT AND GATE

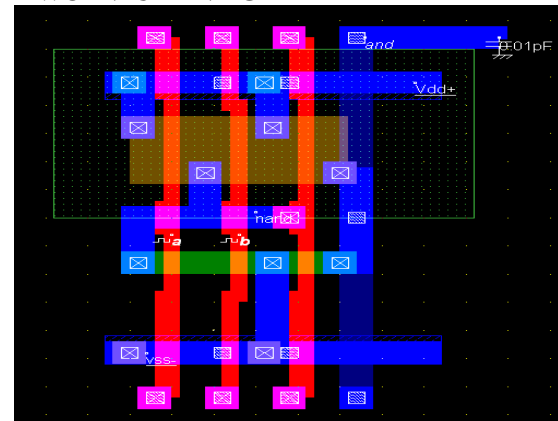


Fig-8.1 Two input AND gate

Total area occupied by the 2 input and gate = $2.45 \times 10^{-9} \text{ m}^2$
 Total power consumed by the 2 input and gate = 1.510 mW

8.2 THREE INPUT AND GATE

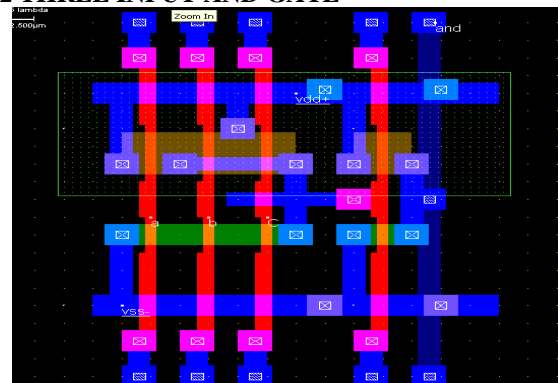


Fig-8.2 Three input AND gate

Total area occupied by the three input and gate = $2.133 \times 10^{-9} \text{ m}^2$ Total power consumed by the three input and gate = 2.272 mW

8.3 FOUR INPUT AND GATE

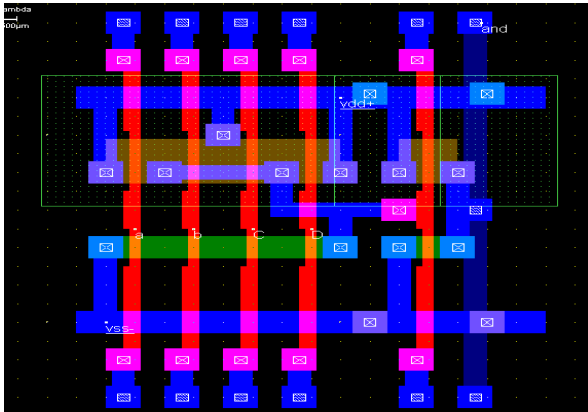


Fig-8.3 Four input AND gate

Total area occupied by the four input and gate = $2.332 \times 10^{-9} \text{ m}^2$ Total power consumed by the four input and gate = 2.979 mW

8.4 INVERTER

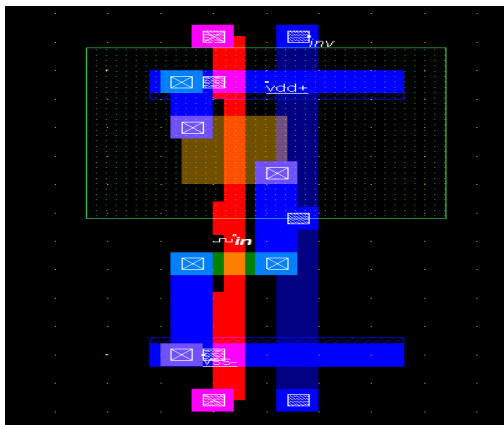


Fig-8.4 Inverter

Total area occupied by the inverter = $578 \times 10^{-12} \text{ m}^2$
Total power consumed by the inverter = 9.274 mW

8.5 TWO INPUT OR GATE

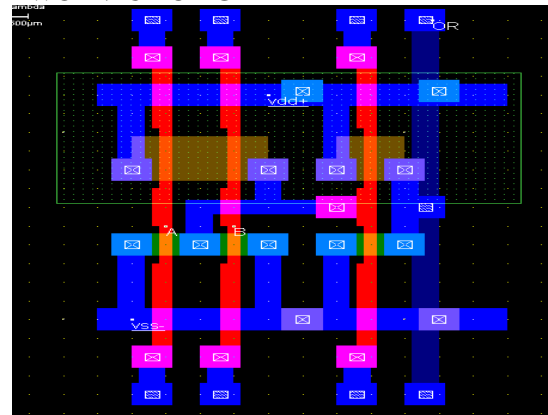


Fig-8.5 Two input OR gate

Total area occupied by the two input or gate = $1.802 \times 10^{-9} \text{ m}^2$ Total power consumed by the two input or gate = 3.078 mW

8.6 THREE INPUT OR GATE

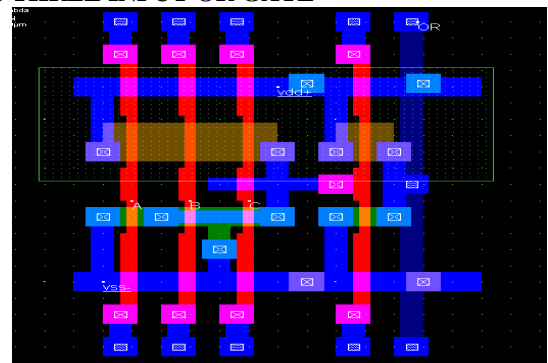


Fig-8.6 Three input AND gate

Total area occupied by the three input or gate = $2.067 \times 10^{-9} \text{ m}^2$ Total power consumed by the three input or gate = 3.811 mW

8.7 TWO INPUT XOR GATE

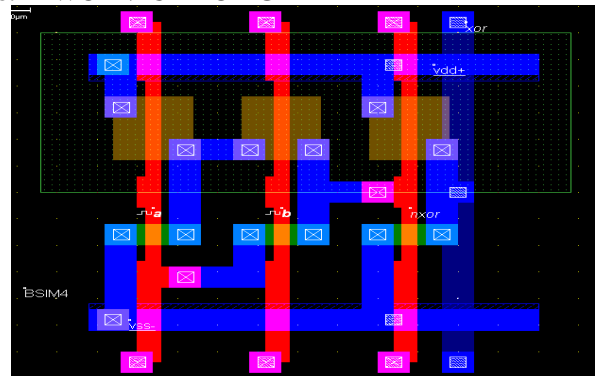


Fig-8.7 Two input XOR gate

Total area occupied by the two input xor gate = $1.122 \times 10^{-9} \text{ m}^2$
 Total power consumed by the two input xor gate = 11.249 mW

8.8 FLIP FLOP

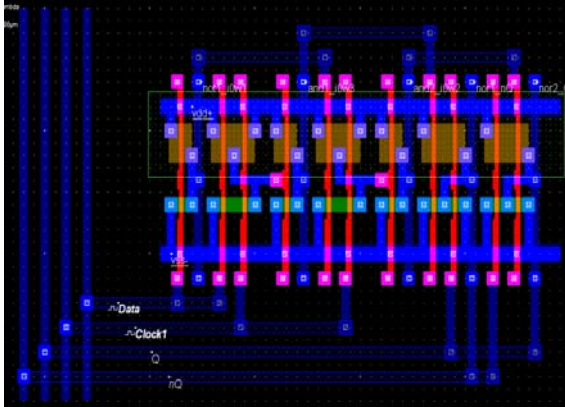


Fig-8.8 Flipflop

Total area occupied by the flip flop = $1.0465 \times 10^{-8} \text{ m}^2$
 Total power consumed by the flip flop = 34.56 mW

8.9 LUT1

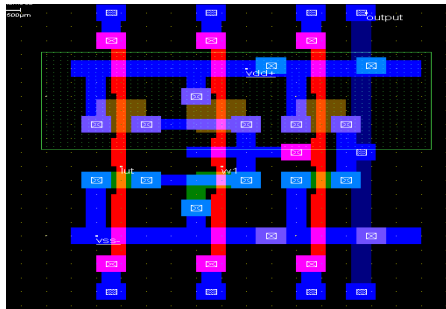


Fig-8.9 LUT1

Total area occupied by the LUT1 = $2.0935 \times 10^{-12} \text{ m}^2$
 Total power consumed by the LUT1 = 9.274 mW

8.10 LUT2

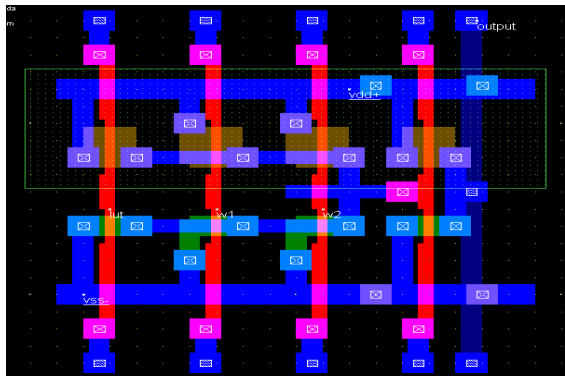


Fig-8.10 LUT2

Total area occupied by the LUT2 = $2.597 \times 10^{-9} \text{ m}^2$
 Total power consumed by the LUT2 = 1.561 mW

8.11 LUT3

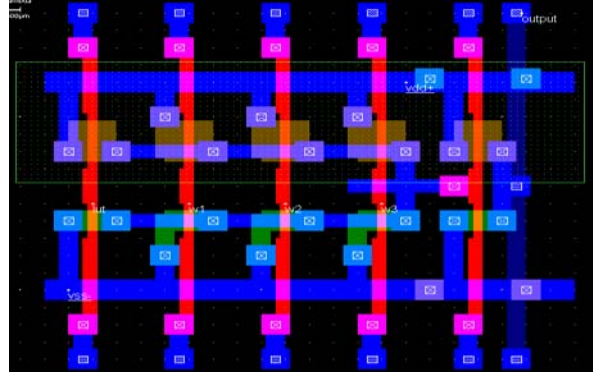


Fig-8.11 LUT3

Total area occupied by the LUT3 = $3.127 \times 10^{-9} \text{ m}^2$
 Total power consumed by the LUT3 = 2.979 mW

8.12 LUT4

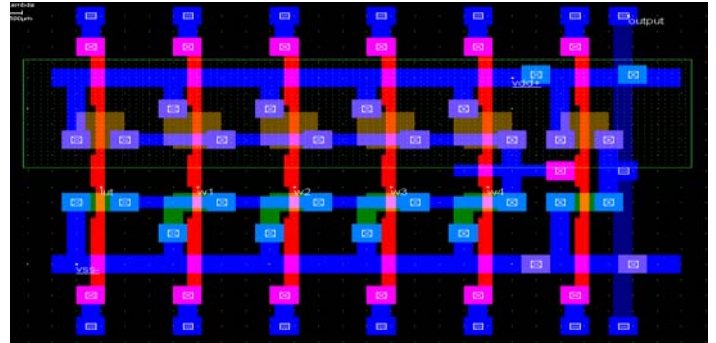


Fig-8.12 LUT4

Total area occupied by the LUT4 = $3.657 \times 10^{-9} \text{ m}^2$
 Total power consumed by the LUT4 = 3.686 mW

8.13 MUX

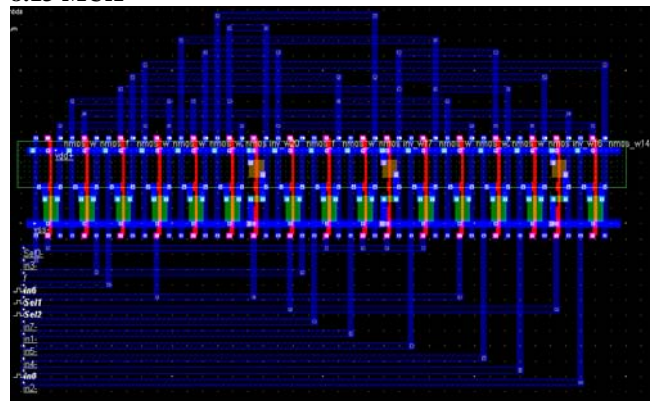


Fig-8.13 MUX

Total area occupied by the MUX = $2.4785 \times 10^{-8} \text{ m}^2$
 Total power consumed by the MUX = 5.129 mW

9. REPORTS

9.1 DESIGN REPORT FOR FPGA

Source Parameters

Input File Name :
 "SURFPGA.prj"

Input Format : mixed
 Ignore Synthesis Constraint File : NO

Target Parameters

Output File Name :
 "SURFPGA"
 Output Format : NGC
 Target Device : xc3s50-4-
 vq100

Source Options

Top Module Name : SURFPGA
 Automatic FSM Extraction : YES
 FSM Encoding Algorithm : Auto
 Safe Implementation : No
 FSM Style : lut
 RAM Extraction : Yes
 RAM Style : Auto
 ROM Extraction : Yes
 Mux Style : Auto
 Decoder Extraction : YES
 Priority Encoder Extraction : YES
 Shift Register Extraction : YES
 Logical Shifter Extraction : YES
 XOR Collapsing : YES
 ROM Style : Auto
 Mux Extraction : YES
 Resource Sharing : YES
 Asynchronous To Synchronous : NO
 Multiplier Style : auto
 Automatic Register Balancing : No

Target Options

Add IO Buffers : YES
 Global Maximum Fanout : 500
 Add Generic Clock Buffer(BUFG) : 8
 Register Duplication : YES
 Slice Packing : YES
 Optimize Instantiated Primitives : NO
 Use Clock Enable : Yes
 Use Synchronous Set : Yes
 Use Synchronous Reset : Yes
 Pack IO Registers into IOBs : auto
 Equivalent register Removal : YES

General Options

Optimization Goal : Speed
 Optimization Effort : 1
 Library Search Order :
 SURFPGA.lso
 Keep Hierarchy : NO

RTL Output : Yes
 Global Optimization :
 AllClockNets
 Read Cores : YES
 Write Timing Constraints : NO
 Cross Clock Analysis : NO
 Hierarchy Separator : /
 Bus Delimiter : <>
 Case Specifier : maintain
 Slice Utilization Ratio : 100
 BRAM Utilization Ratio : 100
 Verilog 2001 : YES
 Auto BRAM Packing : NO
 Slice Utilization Ratio Delta : 5

HDL Synthesis Report

Macro Statistics
 # Multipliers : 1
 8x8-bit multiplier : 1
 # Adders/Subtractors : 7
 16-bit adder : 7
 # Registers : 9
 16-bit register : 1
 8-bit register : 8
 # Multiplexers : 1
 8-bit 8-to-1 multiplexer : 1

Advanced HDL Synthesis Report

Macro Statistics
 # Multipliers : 1
 8x8-bit multiplier : 1
 # Adders/Subtractors : 7
 16-bit adder : 7
 # Registers : 80
 Flip-Flops : 80
 # Multiplexers : 8
 1-bit 8-to-1 multiplexer : 8

Design Statistics

IOs : 31

Cell Usage :

BELS : 505
 # GND : 1
 # INV : 1
 # LUT1 : 3
 # LUT2 : 99
 # LUT3 : 38
 # LUT4 : 95
 # LUT4_D : 21
 # LUT4_L : 13
 # MUXCY : 105
 # MUXF5 : 30
 # MUXF6 : 8
 # VCC : 1
 # XORCY : 90
 # FlipFlops/Latches : 80

#	FD	: 8	16-bit adder	: 7
#	FDRE	: 64	# Registers	: 9
#	FDS	: 8	16-bit register	: 1
#	Clock Buffers	: 1	8-bit register	: 8
#	BUFGP	: 1	# Multiplexers	: 1
#	IO Buffers	: 30	8-bit 8-to-1 multiplexer	: 1
#	IBUF	: 14	<u>Advanced HDL Synthesis Report</u>	
#	OBUF	: 16	<u>Macro Statistics</u>	
#	MULTs	: 1	# Multipliers	: 1
#	MULT18X18	: 1	8x8-bit multiplier	: 1

9.2 DESIGN REPORT FOR CPLD

Source Parameters

Input File Name	:
"SURENCPLD1.prj"	
Input Format	: mixed
Ignore Synthesis Constraint File	: NO
Target Parameters	
Output File Name	:
"SURENCPLD1"	
Output Format	: NGC
Target Device	: XC9500XL

CPLDs

Source Options

Top Module Name	:
SURENCPLD1	
Automatic FSM Extraction	: YES
FSM Encoding Algorithm	: Auto
Safe Implementation	: No
Mux Extraction	: YES
Resource Sharing	: YES

Target Options

Add IO Buffers	: YES
MACRO Preserve	: YES
XOR Preserve	: YES
Equivalent register Removal	: YES

General Options

Optimization Goal	: Speed
Optimization Effort	: 1
Library Search Order	:
SURENCPLD1.iso	
Keep Hierarchy	: YES
RTL Output	: Yes
Hierarchy Separator	: /
Bus Delimiter	: <>
Case Specifier	: maintain
Verilog 2001	: YES

Other Options

Clock Enable	: YES
wysiwyg	: NO

HDL Synthesis Report

Macro Statistics

# Multipliers	: 1
8x8-bit multiplier	: 1
# Adders/Subtractors	: 7

# Adders/Subtractors	: 7
16-bit adder	: 7
# Registers	: 16
Flip-Flops	: 16
# Multiplexers	: 1
8-bit 8-to-1 multiplexer	: 1
<u>Final Results</u>	
RTL Top Level Output File Name	:
SURENCPLD1.ngc	
Top Level Output File Name	:
SURENCPLD1	
Output Format	: NGC
Optimization Goal	: Speed
Keep Hierarchy	: YES
Target Technology	: XC9500XL
CPLDs	
Macro Preserve	: YES
XOR Preserve	: YES
Clock Enable	: YES
wysiwyg	: NO

Design Statistics

# IOs	: 31
<u>Cell Usage :</u>	
# BELS	: 1737
# AND2	: 733
# AND3	: 9
# AND4	: 4
# GND	: 1
# INV	: 219
# OR2	: 444
# OR3	: 6
# XOR2	: 321
# FlipFlops/Latches	: 80
# FD	: 16
# FDCE	: 64
# IO Buffers	: 31
# IBUF	: 15
# OBUF	: 16

9.3 DESIGN REPORT FOR VIRTEX

Source Parameters

Input File Name	:
"SUREN12345.prj"	
Input Format	: mixed
Ignore Synthesis Constraint File	: NO

Target Parameters		Case Specifier	: maintain
Output File Name	:	Slice Utilization Ratio	: 100
"SUREN12345"		BRAM Utilization Ratio	: 100
Output Format	: NGC	Verilog 2001	: YES
Target Device	: xcv50-6-	Auto BRAM Packing	: NO
bg256		Slice Utilization Ratio Delta	: 5
Source Options		<u>HDL Synthesis Report</u>	
Top Module Name	:	<u>Macro Statistics</u>	
SUREN12345		# Multipliers	: 1
Automatic FSM Extraction	: YES	8x8-bit multiplier	: 1
FSM Encoding Algorithm	: Auto	# Adders/Subtractors	: 7
Safe Implementation	: No	16-bit adder	: 7
FSM Style	: lut	# Registers	: 9
RAM Extraction	: Yes	16-bit register	: 1
RAM Style	: Auto	8-bit register	: 8
ROM Extraction	: Yes	# Multiplexers	: 1
Mux Style	: Auto	8-bit 8-to-1 multiplexer	: 1
Decoder Extraction	: YES	<u>Advanced HDL Synthesis Report</u>	
Priority Encoder Extraction	: YES	<u>Macro Statistics</u>	
Shift Register Extraction	: YES	# Multipliers	: 1
Logical Shifter Extraction	: YES	8x8-bit multiplier	: 1
XOR Collapsing	: YES	# Adders/Subtractors	: 7
ROM Style	: Auto	16-bit adder	: 7
Mux Extraction	: YES	# Registers	: 80
Resource Sharing	: YES	Flip-Flops	: 80
Asynchronous To Synchronous	: NO	# Multiplexers	: 8
Multiplier Style	: lut	1-bit 8-to-1 multiplexer	: 8
Automatic Register Balancing	: No	<u>Final Register Report</u>	
<u>Target Options</u>		<u>Macro Statistics</u>	
Add IO Buffers	: YES	# Registers	: 80
Global Maximum Fanout	: 100	Flip-Flops	: 80
Add Generic Clock Buffer(BUFG)	: 4	<u>Final Results</u>	
Register Duplication	: YES	RTL Top Level Output File Name	:
Slice Packing	: YES	SUREN12345.ngr	
Optimize Instantiated Primitives	: NO	Top Level Output File Name	:
Convert Tristates To Logic	: Yes	SUREN12345	
Use Clock Enable	: Yes	Output Format	: NGC
Use Synchronous Set	: Yes	Optimization Goal	: Speed
Use Synchronous Reset	: Yes	Keep Hierarchy	: NO
Pack IO Registers into IOBs	: auto	<u>Design Statistics</u>	
Equivalent register Removal	: YES	# IOs	: 31
<u>General Options</u>		<u>Cell Usage :</u>	
Optimization Goal	: Speed	# BELS	: 704
Optimization Effort	: 1	# GND	: 1
Library Search Order	:	# INV	: 1
SUREN12345.lso		# LUT1	: 2
Keep Hierarchy	: NO	# LUT2	: 149
RTL Output	: Yes	# LUT3	: 36
Global Optimization	:	# LUT4	: 104
AllClockNets		# LUT4_D	: 10
Read Cores	: YES	# LUT4_L	: 15
Write Timing Constraints	: NO	# MULT_AND	: 24
Cross Clock Analysis	: NO	# MUXCY	: 173
Hierarchy Separator	: /	# MUXF5	: 27
Bus Delimiter	: <>	# MUXF6	: 8

# VCC	: 1
# XORCY	: 153
# FlipFlops/Latches	: 80
# FD	: 8
# FDRE	: 64
# FDS	: 8
# Clock Buffers	: 1
# BUFGP	: 1
# IO Buffers	: 30
# IBUF	: 14
# OBUF	: 16

10. DESIGN SUMMARY

10.1 AREA AND POWER DESIGN SUMMARY OF FPGA:

CIRCUIT	TOTAL	AREA OCCUPIED PER CIRCUIT (nm ²)	POWER CONSUMED PER CIRCUIT (mwatts)	TOTAL AREA OCCUPIED (nm ²)	TOTAL POWER CONSUMED (mwatts)
AND2	733	2.45	.0015	1796	1.107
AND3	9	2.133	.00272	19.197	.02245
AND4	4	2.322	.002979	9.328	.0119
INV	219	.578	.009274	126.582	2.031006
OR2	444	1.802	.003078	800.088	1.366632
OR3	6	2.067	.003811	12.462	.022866
XOR2	321	1.122	.0011249	360.162	3.610929
FF	80	10.465	.03456	837.2	2.7648

Table-10.1 AREA AND POWER DESIGN SUMMARY OF FPGA

Total area occupied by the 8 bit mac layout using FPGA = $5297.53 \times 10^{-9} \text{ m}^2$. Total power consumed by the 8 bit mac layout using FPGA = 4.1998 mwatts.

10.2 AREA AND POWER DESIGN SUMMARY OF CPLD:

CIRCUIT	TOTAL	AREA OCCUPIED PER CIRCUIT (nm ²)	POWER CONSUMED PER CIRCUIT (mwatts)	TOTAL AREA OCCUPIED (nm ²)	TOTAL POWER CONSUMED (mwatts)
LUT1	3	2.0935	.0015	6.2805	.004683
LUT2	99	2.597	.00272	257.103	.224928
LUT3	38	3.127	.002979	118.826	.113202
LUT4	95	3.657	.003686	347.415	.35017
INV	1	.578	.009274	.578	.009274

MUX	143	2.47845	.005129	3544.1835	.733447
XOR4	90	2.0647	.0011249	186.03	3.610929
FF	80	10.465	.03456	837.2	2.7648

Table-10.2 AREA AND POWER DESIGN SUMMARY OF CPLD

Total area occupied by the 8 bit mac layout using CPLD = $2166.815 \times 10^{-9} \text{ m}^2$. Total power consumed by the 8 bit mac layout using CPLD = 10.935 mwatts

10.3 AREA AND POWER DESIGN SUMMARY OF VIRTEX:

CIRCUIT	TOTAL	AREA OCCUPIED PER CIRCUIT (nm ²)	POWER CONSUMED PER CIRCUIT (mwatts)	TOTAL AREA OCCUPIED (nm ²)	TOTAL POWER CONSUMED (mwatts)
LUT1	2	2.0935	.0015	4.187	.004683
LUT2	149	2.597	.00272	386.9	.3385
LUT3	36	3.127	.002979	112.57	.107244
LUT4	129	3.657	.003686	471.725	.3833
INV	1	.578	.009274	.578	.009274
MUX	208	2.47845	.005129	5155.1	1.066
XOR4	153	2.0647	.0011249	171.66	1.721
FF	80	10.465	.03456	837.2	2.7648

Table-10.3 AREA AND POWER DESIGN SUMMARY OF VIRTEX

Total area occupied by the 8 bit mac layout using VIRTEX = $7144.628 \times 10^{-9} \text{ m}^2$. Total power consumed by the 8 bit mac layout using VIRTEX = 11.1587 mwatts

CONCLUSION:

Thus the layout for 8 bit MAC has been produced from the schematic obtained from Extrinsic EHW was mapped for FPGA, VIRTEX and CPLD devices to find the optimized resource consumption. ASIC solutions often involve a clearly better product economy. Thanks to small size and packaging, an ASIC gives automatic intellectual property protection. Minimizing the total number of design components leads to very high reliability, that is, low error frequency. Also, an ASIC is rigorously tested before delivery. This all results in considerably lower maintenance costs. From the results we have obtained, it was observed that VIRTEX FPGA is the best suited device for producing the 8 bit Mac in terms of area. The

layout produced by the VIRTEX device has occupied less area when compared to the layouts produced by the other FPGA and CPLD devices. Considering the power consumption it was observed that ALTERA CYCLONE FPGA is the best suited device for producing the 8 bit MAC layout. The layout produced by the FPGA device has consumed less power when compared to the layouts produced by the VIRTEX and CPLD devices. Since the MAC is the basic component in most of the DSP designs, even a small saving in power or area will yield better results at final entity design. It should also be noted that the future is for using the property of reconfigurability in any application to achieve the power and area consumption by loading the necessary hardware on demand directly making use of bit streams. It will be much more efficient if this hardware is evolved rather than conventional flow.

11. REFERENCES

- [1] Gerald R. Clark (1999), "A Novel Function-Level EHW Architecture within Modern FPGAs", Proceedings of the Congress on Evolutionary Computation (CEC 99), IEEE.
- [2] Hollingworth G, Smith S and Tyrrell A (2000), "Safe Intrinsic Evolution of Virtex Devices", Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware, IEEE, pp. 195-202.
- [3] Hollingworth G, Smith S and Tyrrell A (1999), "Design of Highly Parallel Edge Detection Nodes using Evolutionary Techniques", Proceedings of the 7th Euromicro Workshop on Parallel and Distributed Processing, IEEE, pp. 35 – 42.

12. BIOGRAPHIES

1) Mr. D.Dhanasekaran is working as an Assistant Professor, ECE dept. In Sri Venkataswara College Engg. College, Pennalur, Sriperumbudur, affiliated to the Anna university. His areas of interest include Evolvable Computing, reconfigurable computing, VLSI signal processing and neural networks.

2) Dr. K.Boopathy Bagan completed his doctoral degree from Anna university . He is presently working as a professor, Information and Communication Department . In Madras Institute of Technology, Chrompet, Chennai. His areas of interest include VLSI signal processing, Genetic Algorithms and evolvable hardware.