International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 01 | Jan -2017 www.irjet.net p-ISSN: 2395-0072

ASIC Implementation for SOBEL Accelerator

Prema C L[1], Dr Siva Yellampalli[2]

[1]PG student, VTU Extn. Centre, UTL Technologies Ltd, Bengaluru
[2]Principal, VTU Extn. Centre, UTL Technologies Ltd, Bengaluru

---

Abstract - *Due to the Rapid progress in the field of VLSI technology the device are faster than before .This proposed accelerator circuit is implemented in such a way that we can increase the circuit speed . If implement the design using ASIC the single IC can be programmed for several application in different times and it increases the performance or speed of the device. such as video surveillance and monitoring system in security system. so using ASIC implementation on the design will get the fault free chip so it won't cause harm to lives.Implement the accelerator to remaining edge detection algorithm to make the high volume production for asic chip to image processing application and make that chips as cost effective.*

Key Words Accelerator, BIST,ASIC design, Sobel operartor,ATPG,Fault coverage

1.INTRODUCTION

In modern days, to keep speed with the fast progressing of technologies, we have to make everything faster, that's why accelerator comes into the technology. Accelerators[1] which increases the device performance. Accelerators which used in edge detection which speeds up the edge detection process.

Detection of edges in Video images[[2]can be done more rapidly than it is achievable with software which are embedded on a core processor. It is used to achieve faster detection of edge in video images. It refer to a method of detecting and locate the quick changes in the intensity of the frequency of the image. The abrupt changes in the intensity of the pixel can be represented by discontinuity in the image.

Because the improvement in the fabrication technology and thus the increase within the logic FPGA logic block density, the consumption of FPGA is not constricted to any longer than debug and prototyping of digital circuits which leads to increment in the logic blocks in the FPGA .Whenever the time to point is important for some applications ASIC(Application Specific Integrated Circuit)[5] is the solution.

The Application Specific Integrated Circuits (ASIC) design of SOBEL accelerator has two major phases: logical or frontend design and physical or backend design. The steps involved in frontend design are verification of results in pre-synthesis simulation, compile, specifying design constraints, synthesis of design with ASIC technology libraries, Automatic Test Pattern Generation (ATPG), insertion of Design For Test (DFT)) creation and verification of results in post synthesis simulation. The physical layout design involves floor plan, power plan, placement, routing, various timing analysis and verifications of completed design, and verification of results in post layout simulation. In this project the ASIC implementation of Sobel accelerator is explained in detail.

This document is template. We ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to download the template, and replace(copy-paste) the content with your own material. Number the reference items consecutively in square brackets (e.g. [1]). However the authors name can be used along with the reference number in the running text. The order of reference in the running text should match with the list of references at the end of the paper.

2.Accelerators for edge detection in video images

An accelerator for edge detection[2] in video images which is compromise between what a real world accelerator ,might do.Edge detection is important part of analysing a scene in video images and has applications in many areas such as security monitoring and computer vision application[3]. It involves identifying places in image where there is abrupt change in intensity.Those places usually occur at the boundaries of objects. subsequent analysis of the edge can be recognizing what the objects are.

Assume Monochrome images of 640x480 pixels[1,6],each of 8 bits ,stored row by row in memory with successive pixels,left to right in a row,at successive addresses.Pixels values are interpreted as unsigned integer ranging from 0(black) to 255(white) .will use a algorithm called sobel edge detector .It works by computing the derivatives of the intensity signal in each of the x and y

directions and looking for maxima and minima in the derivatives.These are the places where the intensity is changing most rapidly.The sobel method approximates the derivatives in each direction for each pixel by a process called convolution. This involves adding the pixels and its eight neighbours ,each multiplied by a coefficient. The coefficients are represented in a 3x3 convolution mask. The sobel convolutions masks,Gx and Gy ,for the derivatives in the x and y directions respectively. the derivatives images being computed by centering each of the convolution masks over successive pixels in the original image.after multiplying the coefficient in each mask by the intensity values of the underlying pixel and sum the nine products together to form two partial derivatives for the derivative image,Dx and Dy Ideally will compute the magnitude of the derivative image pixel as

-1	0	+1
-2	0	+2
-1	0	+1

GX

+1	+2	+1
0	0	0
-1	-2	-1

GY

Fig 2.1 Sobel Convolution mask

$$|D| = \sqrt{(Dx2 + Dy2)}$$

3 PROPOSED Accelerator DESIGN for Implementation

3.1 Working principle

Basically it is a pipeline structure, where with pixel data read from the original image entering into the registers at the top right, flowing through the 3x3 multiplier array on the left, then down through the adders to the Dx and Dy registers, then through the absolute value circuits and adder to the |D| register, and finally into the register at the bottom left.The resulting derivative pixels are then written from that register to memory. (While a right-to-left data flow is opposite to usual practice, in this case, it has the advantage of preserving the same arrangement of pixels as that in an image.) We will describe the operation of the pipeline assuming initially that it is full of data. We will then discuss how to deal with starting it up at the beginning of an image row and draining it at the end of the row.

The pipeline generates the derivative pixels for a given row in groups of four. The accelerator reads four pixels from each of the preceding, current, and next rows in memory into the three 32-bit registers at the top right of the figure. Each register consist of four 8-bit pixel registers. Over the four subsequent clock cycles, pixels are shifted out to the left, one pixel at a time, into the multiplier array. Each cell in the array contains a pixel register and one or two circuits that multiply the stored pixel by a constant coefficient value. Since the coefficients are all +1, -1, +2, or -2, the circuits are not full-blown multipliers. Instead, multiplying by -1 is simply a negator, multiplying by +1 is a through connection with no circuitry, multiplying by -2 is a left shift of the result of a negator, and multiplying by +2 is simply a left shift. On each clock cycle, the array provides the partial products for a single derivative pixel, and the partial products are added and stored in the Dx and Dy registers. Also, on each clock cycle, the Dx and Dy values for the preceding pixel have their absolute values computed and added and stored in the |D| register. The resulting derivative pixel values are shifted into the result row register.

When four result pixels are ready in the register, they are subsequently written to memory. In the steady state, during processing of a row, the accelerator needs to write the pixels to memory from the result register before it can shift new pixels into the multiplier array and the Dx, Dy and |D| registers. Otherwise, the result values would be overwritten. Having written four pixels, the accelerator can push four more pixels through the pipeline, thus emptying the read registers and filling the result register. It can then write those result pixels and read in three more groups of four pixels,and repeat the process. assuming a Wishbone bus connection with 32-bit-wide data signals and a 100MHz clock, as suggested earlier. Since the accelerator is one of

several masters on the memory bus, it must request use of the bus for the writes and reads and wait until granted access by the bus arbiter. We assume that the arbiter gives the accelerator sufficiently high priority that it can use the memory bandwidth it needs.

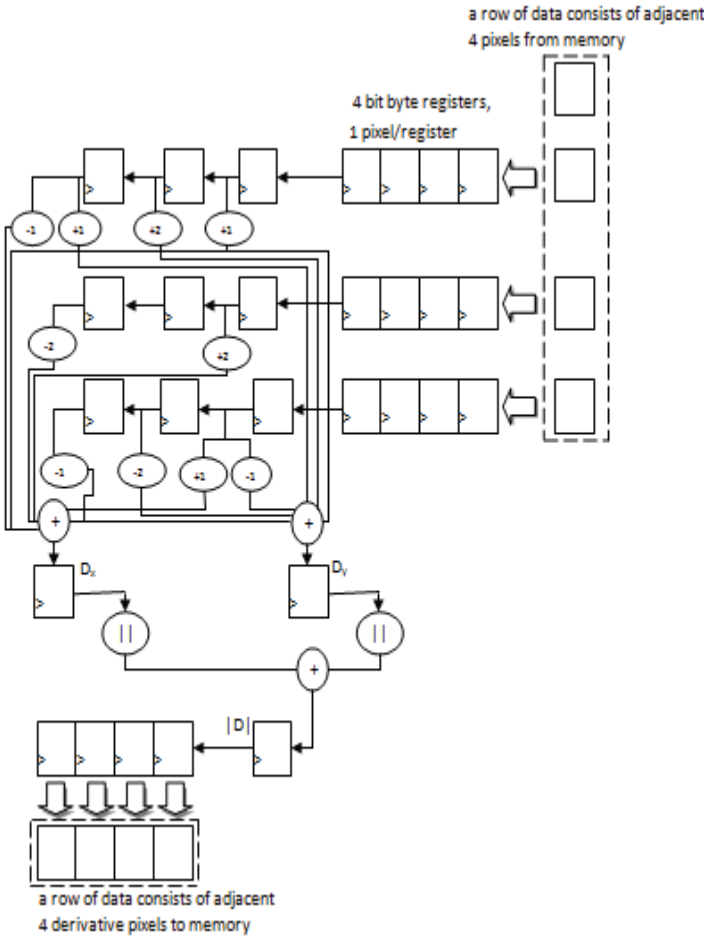

Fig 3.1 Sobel Accelerator Architecture

4. ASIC DESIGN OF SOBEL Accelerator

As compared to FPGA which has limited number of logics, the ASIC supports integration of larger number of logic gates (higher in density). ASICs are designed to fit a certain application. ASIC implementation supports digital or mixed-signal circuit in the design. The ASIC design saves cost, fabrication time for mass production of the system.

There are two methods of design flow in ASIC: frontend and backend design flow. The front end design involves inclusion of technology library files, design compilation, elaboration, creation of design constraints, synthesize, Design For Test (DFT) insertion and Logic Equivalence Checking (LEC).

The backend physical layout design involves floor planning of blocks, power planning, placement, layout routing and timing verifications. The ASIC design of CCG has been carried out with the help Cadence tools. The ASIC design aspects are described in the following sections

4.1. ASIC FRONTEND DESIGN

Compile and simulation of design

The Verilog design files and test bench modules are compiled, elaborated, simulated for timing verification [10]. The simulation waveforms of pre-synthesize are as shown in fig-5.

Creating design constraints for Synthesis Clocks are used to provide timing and synchronization information for digital design. The design constraints are specified for optimizing synthesize tool to meet timing requirements. The constraints [11] are identified in this design are master clock signal and global reset signal which are propagated all the parts of the design. These signals require larger fan-out and should have lowest skew in timing.

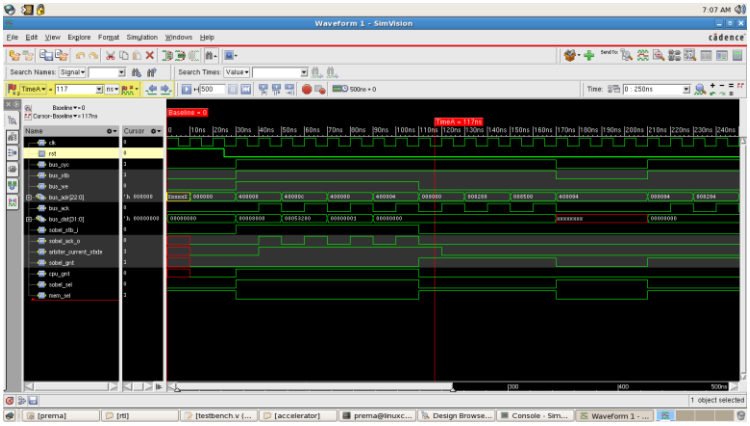

Fig 4.1 Initialisation of sobel

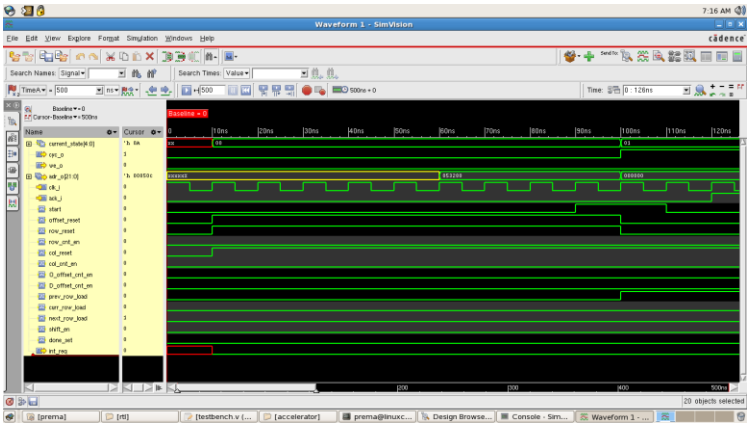

Fig 4.2 The timings are verified successfully as per design specification

The derived clock used in clock driver section is also included in constraint list. All the design constraints are

specified with the help of RTL Compiler before synthesize the design.

Synthesis of Design

Synthesis [13] process maps the HDL design with standard cells specified in technology driven library files. This library file has timing, power and area related information. The equivalent physical library file has information about size, internal delays of cells and input output signals of standard cells. The physical library files are Library Exchange Format (LEF), Quantus RC (QRC) file or Capacitance table are required for backend ASIC design. These library files are provided by the chip manufacturer. In order to reduce the effort in backend design, the physical library files [5] are used as a part of synthesize process. This method is used for generating floor planning information as Design Exchange Format (DEF) file required for backend design. Based on SDC constraints, the RTL compiler generated optimized hardware in the form of gate level netlist based on standard cells to meet design constraint requirements.

Generation of report files and backend import files

After synthesize with technology library, the report files are generated for the mapped gate level netlist. The fig- shows report of power and area for the synthesized design.

```
================================================
  Generated by:           Encounter(R) RTL Compiler v14.20-s038_1
  Generated on:           Apr 05 2016  04:57:51 am
  Module:                 sobel
  Technology library:     fast
  Operating conditions:   fast (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
================================================

                                    Leakage    Dynamic     Total
        Instance          Cells   Power(nW)   Power(nW)  Power(nW)
-------------------------------------------------------------------
sobel                      1245 129405.704  399943.648 529349.352
  csa_tree_s..128_13_groupi   84   6649.613   16278.008  22927.621
  csa_tree_sub00281_groupi    84   6428.147   15217.803  21645.950
  csa_tree_a..153_36_groupi   35   4367.067   20661.621  25028.688
  csa_tree_a..154_36_groupi   34   4210.434   19824.801  24035.235
  add_163_26                  24   2622.600   12075.334  14697.934
  final_adder_add_152_31      24   2622.600   11202.188  13824.788
  inc_add_150_52_3            46   1615.725    2942.381   4558.107
  inc_add_161_52_4            46   1615.725    3389.627   5005.353
```

Fig-4 Synthesize Report from RTL Compiler

Further, the synthesized reports based on PLE are generated to simplify the burden of design import to physical layout design.

Insertion of Design For Test (DFT)

The DFT adds additional hardware for the design which includes testability features to the verified design. The added hardware validates the product hardware from manufacturing defects.

In this design the simple method of scan chain DFT with Full scan mode has been designed. The scan chain test patterns are used to access the internal nodes of the chip by shifting. The synthesize steps are repeated to include DFT along with design. The additional hardware pins as the result of DFT insertions are Scan In (SI), Scan Out (SO) and Scan Enable (SE).

Automatic Test Pattern Generation (ATPG) are generated with test vectors in Verilog form for scan test and logic test mode in order to verify fault and global coverage of the design. The ATPG engine used for generating test patterns to test the structural integrity of the design.

6. ASIC BACKEND DESIGN

During frontend design, the reports and timings are analyzed and verified before importing the design to layout. The following are the major steps involved during backend ASIC design.

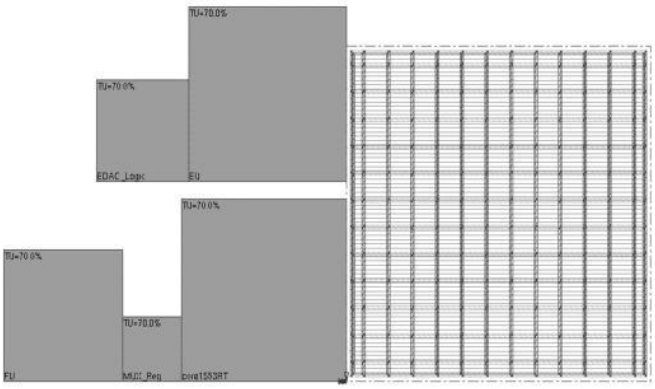

Fig 6.1 Floor planning and power planning

Power and ground rings are added with vertical and horizontal metal layers placed around the core. The main aim of the power planning [15] is to provide equal amount of supply to all standard cells without any supply loss. The change in supply may adversely affect threshold voltage of standard cells, setup or hold timing violations of the design.

In order to provide equal amount of power to the entire design, the power strips are created vertically and horizontally, as mesh inside the core. For ease of signal routing, top layer has been selected for power and ground nets.

```
Command:          analyzeFloorplan -cong -timing -effort medium
################################################################
********************** Analyze Floorplan ************************
   Die Area(um^2)              : 144437.580000
   Core Area(um^2)             : 144437.580000
   Chip Density (Counting Std Cells and MACROs and IOs): 99.955%
   Core Density (Counting Std Cells and MACROs)        : 99.955%
   Average utilization         : 100.120%
   Number of instance(s)       : 7225
   Number of Macro(s)          : 0
   Number of IO Pin(s)         : 202
   Number of Power Domain(s) : 0
********************** Estimation Results ************************
   Total wire length.          : 1.2610e+05
   Congestion (H).             : 0.036%
   Congestion (V).             : 0.931%
   WNS reg2reg (ns).           : 3.4558e+00
   TNS reg2reg (ns).           : 0.0000e+00
******************************************************************
```

Fig 6.2 Analysis of floor plan

Timing analysis

Timing analysis [15] is carried out to check setup or hold timing violations. The timing analysis of design requires timing library files for slow, fast and Signal Integrity (SI) files. The set of library files are created for min and maximum timing analysis. The design constraints created during synthesize process also included for layout timing analysis. The timing analysis is carried out in three different phases of ASIC layout design: Pre Clock Tree Synthesize (CTS) post CTS and post route.
The additional delays are introduced by metal layers and coupling capacitances analyzed after post routing. Capacitance table library file has been included for post route analysis.

Placing CTS

Clock tree or clock buffer has higher driving capability and are usually placed in design based on larger fan-out requirements. The clock buffer uniformly distributes the clock signals with lowest skew in timing. The clock tree buffers are identified from the standard cells. The identified cells are placed inside the core.

Layout Routing

Layout routing [17] is classified as global and detailed routing. The global routing generates estimated delays according to statics by observing previous manufactured chips. The detailed routing generates actual wire delays that can be obtained by several timing optimizations.
Clock tree or clock buffer has higher driving capability and are usually placed in design based on larger fan-out requirements. The clock buffer uniformly distributes the clock signals with lowest skew in timing. The clock tree buffers are identified from the standard cells. The identified cells are placed inside the core.

Post route timing analysis

The post route timing has been extracted for analysis. The values of Worst Negative Slack (WNS) and Total Negative Slack (TNS) are positive.

```
**optDesign ... cpu = 0:00:04, real = 0:00:04, mem = 893.3M, totSessionCpu=0:01:41 **
Reported timing to dir ./timingReports
**optDesign ... cpu = 0:00:04, real = 0:00:04, mem = 893.3M, totSessionCpu=0:01:42 **
Found active setup analysis view best_case
Found active hold analysis view worst_case

------------------------------------------------------------
     optDesign Final Summary
------------------------------------------------------------

+--------------------+--------+--------+---------+
|     Setup mode     |   all  | reg2reg | default |
+--------------------+--------+--------+---------+
|           WNS (ns):|  4.296 |  6.180 |  4.296  |
|           TNS (ns):|  0.000 |  0.000 |  0.000  |
|    Violating Paths:|    0   |    0   |    0    |
|         All Paths:|  1239  |   813  |   592   |
+--------------------+--------+--------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 69.810%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
**optDesign ... cpu = 0:00:04, real = 0:00:04, mem = 891.3M, totSessionCpu=0:01:42 **
*** Finished optDesign ***
```

Fig 6.3 Post route timing analysis

Post route verification

The following verifications are carried out after post routing: Verification on Geometry, Design Rule Checks (DRC), connectivity, Bus guide, metal via and process antenna. All the verifications are passed with zero violations.

TABLE – I POST ROUTE VERIFICATIONS

Verification on	Violations
Bus guide	Zero
Geometry	Zero
Connectivity	Zero
Process antenna	Zero
DRC	Zero

SUMMARY OF TOOLS USED

The following tool sets from Cadence are used for completing this project.

TABLE -2 Tools Used

Cadence tool name	Used for
NCLAUNCH	Presynthesis
NCSIM	Simulation
RTL COMPILER	Synthesis
ENCOUNTER TEST	DFT

7.Conclusion

The ASIC Implementation for Sobel Accelerator completed successfully which compiles all the design requirements. The timing in PreCTS, PostCTS and Post-Route

analysis are verified successfully. The ASIC post layout verifications carried out on geometry, DRC, connectivity, metal density, process antenna. The GLN and SDF file are extracted from the layout for simulation. Sobel Accelerator is implemented using ASIC flow. Area, power, timing and gates reports analysed.In the ASIC flow have also Implemented SDC files,DFT insertion, Physical design and verified with timing violations. The Sobel accelerator implementation in ASIC has been completed with Taiwan Semiconductor Manufacturing Technology (TSMC) 90nm technology library files. The core die size is 0.11mm^2 and 6 metal layers are used for the layout design to complete. After all the verifications, the GDS file has been generated which can be supplied to chip manufacturer for fabrication.

Future work

ASIC implementation done for Sobel accelerator for video images in medical and defence application which depend on sobel edge detection algorithm but various algorithms are used for edge detection such as prewitt,canny,Robert algorithms so by developing accelerator for those algorithms and implement that to into ASIC flow to make their own chip in large volume for corresponding application.

References:

[1]. P.J. Ashenden, Digital Design An Embedded System Approach Using Verilog, Morgan Kaufmann, 2008.

[2]. An FPGA based Hardware Accelerator for Real Time Video Segmentation System ICACSIS 2011 ISBN: 978-979-1421-11-9

[3].Hardware Description of Multi-Directional Fast Sobel Edge Detection Processor by VHDL for Implementing on FPGA , Volume 47– No.25, June 2012

[4].FPGA based Image Edge Detection and Segmentation, Vol.9.No.2, 2011,p.187-192.

[5]. Michael John Sebastian Smith, "Application Specific Integrated Circutis", Pearsion Education Inc, 12th impression, 2013

[6]. I.Yasri, N.H.Hamid, V.V.Yap, "Implementation of an FPGA based Sobel Edge Detection Operator", IGCES, 2008.

[7].Shukor, Lo HaiHiung, Patrick Sebastian3, 2007. —Implementation of Real-time Simple Edge Detection on FPGA‖ pp. 1404-1405,IEEE.

[8].Cadence NCLaunch User Guide, Product Version 14.1, June 2014

[9].Cadence RTL Compiler User Guide, Product Version 12.2, August

[10].Setting Constraints and Performing Timing Analysis Using Encounter RTL Compiler, Product Version 12.2, August 2013

[11]. Design with RTL Compiler Physical, Product Version 12.2, May 2013

[12].Design For Test Encounter RTL Compiler, Product Version 12.2, August 2013

[13].A. Chandra, S. Chebiyam, and R. Kapur Synopsys, Inc., "A Case Study on Implementing Compressed DFT Architecture" , IEEE 23rd Asian Test Symposium, 2014, pp. 336 - 341

[14] Message Reference for Encounter RTL Compiler, Product Version 12.2, May 2013

[5] Cadence EDI User Guide, Product Version 14.20, October 2014

[16] Cadence I/O Planner: Application Note, Product Version 16.2, November 2008

[17] Plato NanoRoute User‟s Guide, Version 2.5, Rev. D

[19] SDF Timing Annotation, Product Version 14.2, January 2015