IMPLEMENTATION OF AN ADVANCED LUT METHODOLOGY BASED FIR FILTER DESIGN PROCESS

[1] G. Sowmya Bala [2] A. Rama Krishna
1 PG student, Dept. of ECM. K.L.University, Vaddeswaram, A.P, India,
2 Assistant Professor, Dept. of ECM. K.L.University, Vaddeswaram, A.P, India

ABSTRACT:

This manuscript describes the process of implementing the techniques for improving the area efficiency of an FIR filter by combined coding approaches. This shows the flexibility in partitioning the filter coefficients for lookup-table (LUT) based implementation. Implementation of an FIR filter with this improved techniques results that, it can handle up to n number of input bits as FIR filter coefficients, and optimally partition these bits to achieve area-efficiency. This work is implemented using software XILINX 9.2i ISE synthesis and simulation tool. Spatern-3e FPGA can be used to test the design process that provides low cost, high performance logic solutions. Designing and implementing an FIR filter using the advanced approaches which can reduce the size of LUTs is tested on the Spatern-3e FPGA results in improving systems performance.

Key words: - LUT, FPGA, FIR Filter, Spatern-3e FPGA, Xilinx ISE.

1. INTRODUCTION:

Silicon area-efficiency, speed, and power are three metrics where there remains a significant gap between FPGAs and ASICs. With the growth of VLSI technology, reconfigurable design styles are widely used for either pre-silicon hardware/software co-verification or for small and medium volume ASIC products. Field programmability enables fast re-spin turn around time and hence speeds up time to market. There are two major categories of reconfigurable devices, the field programmable gate array (FPGA) and the complex programmable logic device (CPLD).

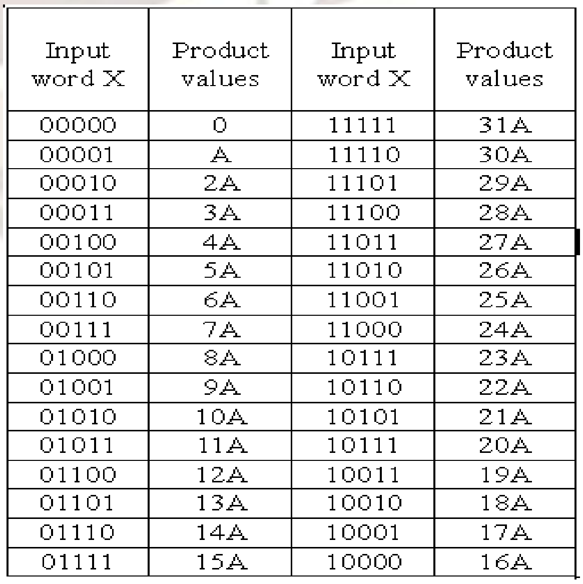
The FPGA utilizes lookup tables (LUT) to implement multi-level functions in order to maximize node sharing in a Boolean network [2]. Since the invention of FPGAs in the mid-1980s, Look-up-tables (LUTs) have been the basis of FPGA logic blocks. A *K*-LUT is a single-output memory with *K* address lines that can implement *any* Boolean function that uses up to *K* variables. The earliest FPGAs used 4-LUTs, established as the best LUT size to maximize area efficiency [1].

The commercial vendors add extra outputs onto their LUTs – a straightforward modification due to the nature of a LUT's implementation in hardware, which is a tree structure. The LUTs in modern FPGAs are reduced to smaller LUTs. LUTs in the Xilinx Virtex-6 FPGA can implement any single 6-variable logic function, or any two functions that together use up to 5 distinct variables [4]. The 6-LUT in Altera's Stratix IV FPGA offers even more flexibility, including the ability to implement two separate 4-variable functions [3].

Xilinx has two main FPGA families: the high-performance Virtex series and the high-volume Spartan series. The Virtex series of FPGAs have integrated features such as wired and wireless infrastructure equipment, advanced medical equipment, test and measurement, and defense systems. The Spartan series targets applications with a low-power footprint, extreme cost sensitivity and high-volume such as displays, set-top boxes, wireless routers and other applications. Spatern-6 is a low-cost solution for automotive, wireless communications, flat-panel display and video surveillance applications.

The Spartan-3A consumes more than 70-90 percent less power in suspend mode and 40-50 percent less for static power compared to standard devices. In addition, the integration of dedicated DSP circuitry in the Spartan series has inherent power advantages of approximately 25 percent over competing low-power FPGAs. The Spartan-3E family of Field-Programmable Gate Arrays (FPGAs) is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications.

Table 1 General LUT Table



Input word X	Product values	Input word X	Product values
00000	0	11111	31A
00001	A	11110	30A
00010	2A	11101	29A
00011	3A	11100	28A
00100	4A	11011	27A
00101	5A	11010	26A
00110	6A	11001	25A
00111	7A	11000	24A
01000	8A	10111	23A
01001	9A	10110	22A
01010	10A	10101	21A
01011	11A	10111	20A
01100	12A	10011	19A
01101	13A	10010	18A
01110	14A	10001	17A
01111	15A	10000	16A

The Spartan-3E family builds on the success of the earlier Spartan-3 family by increasing the amount of logic per I/O, significantly reducing the cost per logic cell [5]. New features improve system performance and reduce the cost of configuration.

2. FLOW GRAPH OF THE PROPOSED SYSTEM:

In this paper, it is shown that the input word is decomposed into equivalent 5-bit words and the each decomposed word now undergoes the loop LUT. Every word say xi undergoes the loop LUT at the same time intervals and thus saves the required time. In the loop-LUT, the specific functional/logical and arithmetic operation is performed. The flow graph representation in figure 2 shows the performance of the corresponding operation. A concurrent operation takes place for an improved processing speed and an accurate desired output is resulted at the end of the entire process.

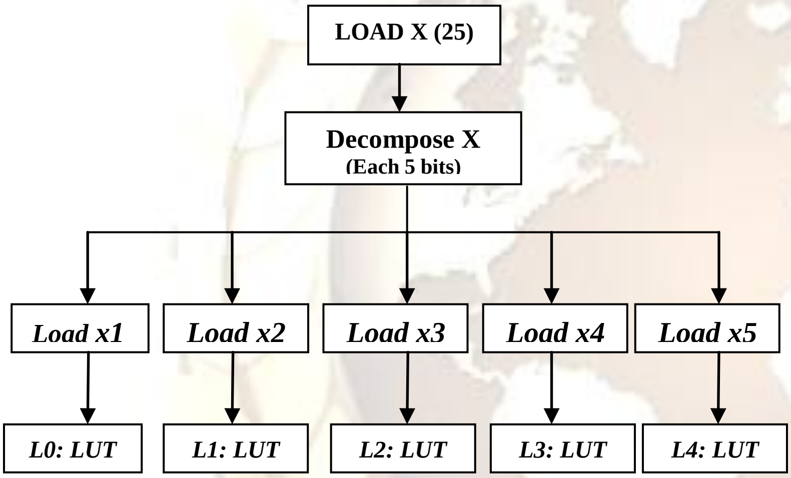


Figure 1 Flow chart of the design process with decomposition scheme

The flow chart representation of the methodology employed is shown in figure 1 and figure 2. The look up table is initially consisting of some desired results and they are stored in the corresponding addresses. The operation of LOOP LUT is to extract the correct value as output as per the input so far given to it. This output should be same when computed manually. In addition, for an n number of input values the desired results can be computed and the desired output is drawn from the LUT buffer in which the multiplication results are initially get stored. With the help of simple shifters and adders with in the device the operations are performed this reduces the need of additional components to compute the special functions and normal arithmetic and logic functions.

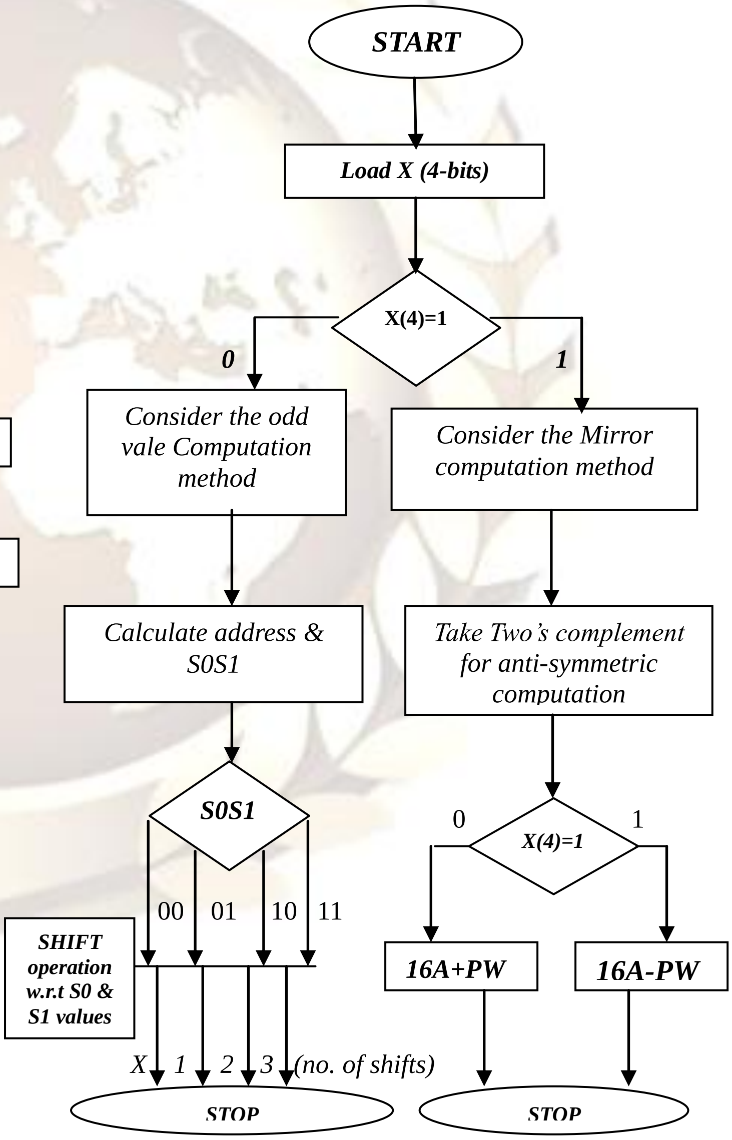
Advantages:

1. Area: By using this technique, there is a reduction of around 75% of memory from the wastage.
2. Time: The complexities in terms of time required for retrieving the outputs (with in less span of time) is reduced.

Applicable Areas

1. For high-speed computations in FPGAs
2. Communication Technologies i.e. wireless technology especially for spectrum sensing techniques in cognitive radio of a Software Defined Radio
3. FIR filter, when designing it resembling the Matched filter structure, which is applicable for many applications especially for spectrum sensing in cognitive radio of SDR

LOOP "LUT":


START
Load X (4-bits)
X(4)=1
0
1
Consider the odd vale Computation method
Consider the Mirror computation method
Calculate address & S0S1
Take Two's complement for anti-symmetric computation
S0S1
00
01
10
11
SHIFT operation w.r.t S0 & S1 values
X
1
2
3
(no. of shifts)
STOP
0
1
X(4)=1
16A+PW
16A-PW
STOP

PW-Product Word

Figure 2 Flow chart of the loop LUT in the decomposition scheme

3. IMPLEMENTATION PROCESS:

3.1. ALGORITHM:

A) Top module:

Step 1: Load the 25-bits multiplicand values to the input unit/ register X
Step 2: Decompose the 25-bits input value into five 5-bit words
Step 3: Load the word of 5-bits each into another register x
Step 4: Assign this to a sub module.

B) Sub module:
Step1: Load input multiplicand value into x register
Step2: Decide whether to implement Method 1 or Method 2
Step3: If x (4) =1 then select Method 1 i.e., considering the mirror or anti-symmetric computation.
Step4: Else select Method 2 i.e., considering odd values first store method.

B1) Method 1:
Step1: Take 2's complement of x and pass to next block
Step2: Calculate product word of x
Step3: If x (4) =1 then

```
         Output <= 16A - product word(x)
Else     Output <= 16A + product word(x)
```

Here 16A is the fixed coefficient value to which the product word is added or subtracted as per the condition applied.

B2) Method 2:
Step 1: Takes last four bits of x.
Step 2: Calculate Control bits s0, s1 and address.
Step 3: Depends on control bits s0, s1 the desired result is shifted and then stored into final output.

3.2. SPARTAN-3E FPGA

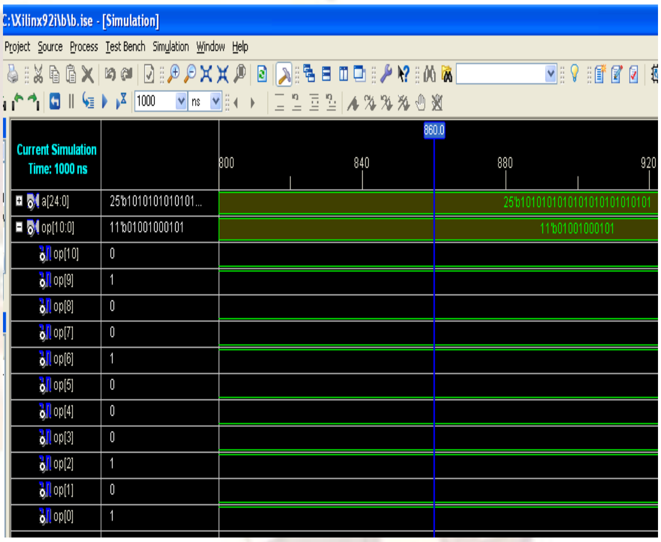
In this project for the simulation purpose the Xilinx version used is Xilinx 9.2i ISE which is used to dump the program into the SPARTAN 3E (TQ144) kit which is having 144 pins where 100 pins are for input and output, 1 pin for global clock and remaining pins for future purpose. The Xilinx is the interface between the Modelsim and the FPGA kit, which converts the code in the Modelsim into the code that can be dumped into the FPGA kit.

3.2.1. Features of Spatarn-3E

1. Very low cost, high-performance logic solution for high-volume, consumer-oriented applications [7].
2. Proven advanced 90-nanometer process technology, Multi-voltage, multi-standard Select IO™ interface pins
3. Enhanced Double Data Rate (DDR) support, Abundant, flexible logic resources
4. Efficient wide multiplexers, wide logic, Fast look-ahead carry logic
5. Eight global clocks plus eight additional clocks per each half of device, plus abundant low-skew routing Configuration interface to industry-standard PROMs
6. Fully compliant 32-/64-bit 33 MHz PCI support (66MHz in some devices)

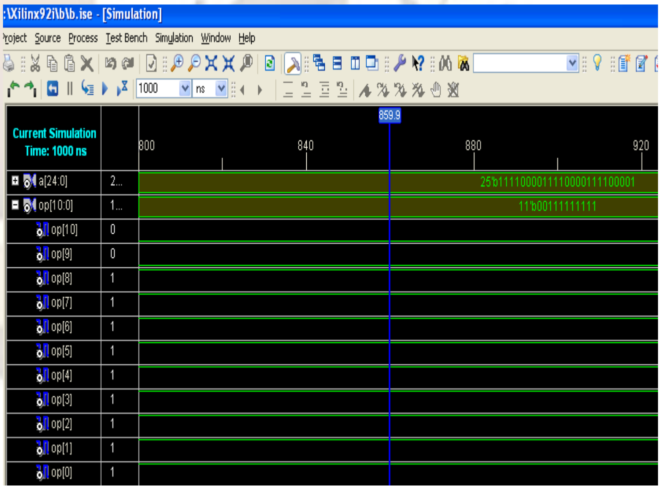
4. RESULTS:

The waveform shows the simulated result of the proposed design. The overall implementation process is simulated using XILINX 9.2i ISE Project Navigator.


C:\Xilinx92i\b\b.ise - [Simulation]
Project Source Process Test Bench Simulation Window Help
1000
ns
Current Simulation Time: 1000 ns
860.0
800
840
880
920

```
a[24:0]    25'b1010101010101...   25'b1010101010101010101010101
op[10:0]   11'b01001000101        11'b01001000101
op[10]     0
op[9]      1
op[8]      0
op[7]      0
op[6]      1
op[5]      0
op[4]      0
op[3]      0
op[2]      1
op[1]      0
op[0]      1
```

Figure 3(a) ISE Simulated output of top module for input a= 25'b (1010101010101010101010101).


:\Xilinx92i\b\b.ise - [Simulation]
roject Source Process Test Bench Simulation Window Help
1000
ns
Current Simulation Time: 1000 ns
859.9
800
840
880
920

```
a[24:0]    2...   25'b1111000011110000111100001
op[10:0]   1...   11'b00111111111
op[10]     0
op[9]      0
op[8]      1
op[7]      1
op[6]      1
op[5]      1
op[4]      1
op[3]      1
op[2]      1
op[1]      1
op[0]      1
```

Figure 3(b) ISE Simulated output for to module for input a= 25'b (1111000011110000111100001).

The waveform in figure 4 shows the simulation result for the general LUT, which is the basic technique. The simulation result for different input values are as shown in figure 3(a) and 3(b). For a given input value, the desired output is obtained with an improved processing speed. As shown in the figure 4 for a general LUT the input word size is limited to only 4-bits, where as the proposed system is able to compute n number of bits with nothing difference in terms of the processing speed and the computation time. Even the input-word size increases, the number of slices required and the number of input LUT's required so far remains same. The detailed information regarding Macro Statistics, Device Utilization, and Design Statistics is clearly mentioned in the HDL Synthesis Report.

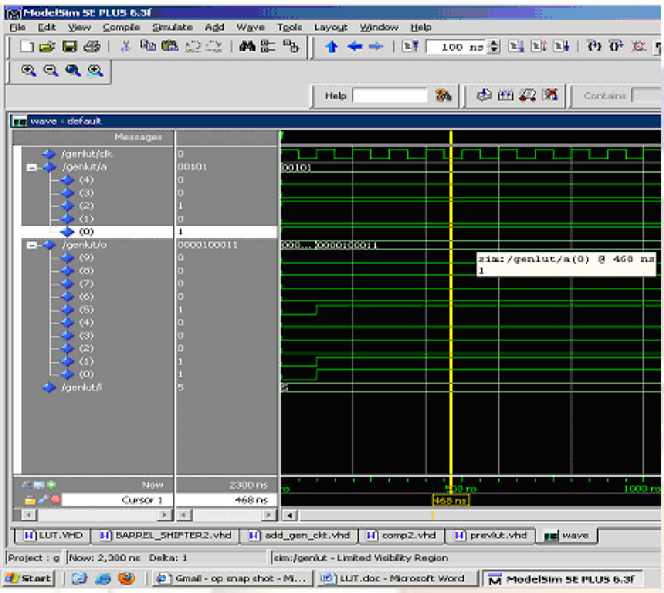


Figure 4 Simulation result of General LUT

HDL Synthesis Report

Macro Statistics:

ROMs	: 05
16x11-bit ROM	: 05
Adders/Subtractors	: 14
11-bit adder	: 09
11-bit adder carry in	: 05
Logic shifters	: 10
11-bit shifter rotate left	: 05
4-bit shifter rotate right	: 05

Final output:

RTL Top Level Output File Name	: top.ngr
Top Level Output File Name	: top
Output Format	: NGC
Optimization Goal	: Speed
Keep Hierarchy	: NO

Design Statistics:

# IOs	: 36

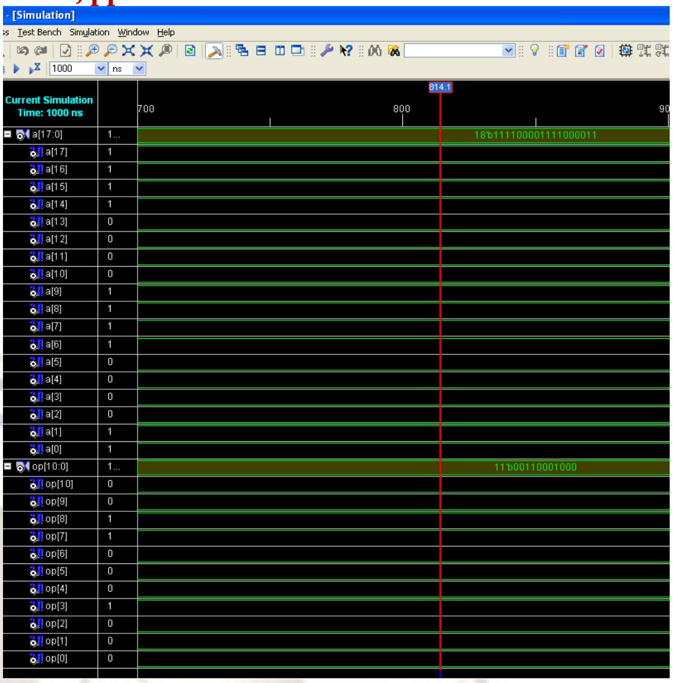


Figure 5 Simulation result for a 20-bit input value.

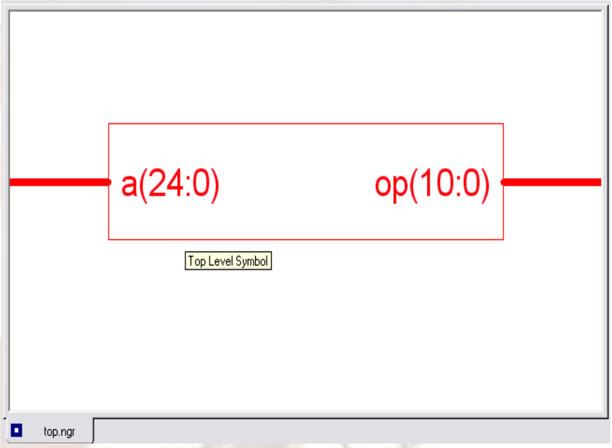


Figure 6(a) Top Level Symbol.

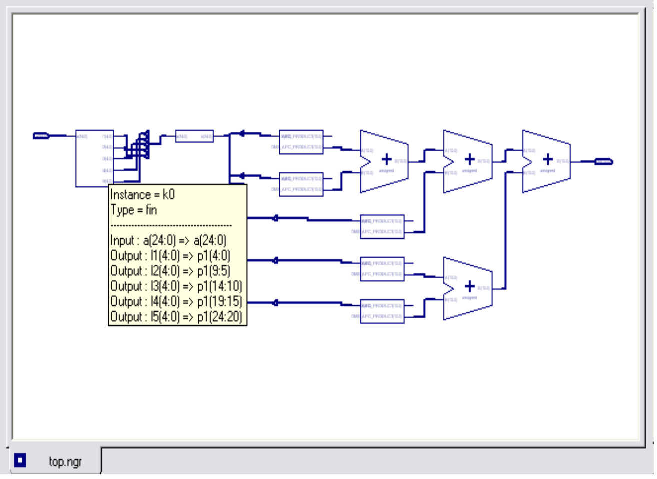
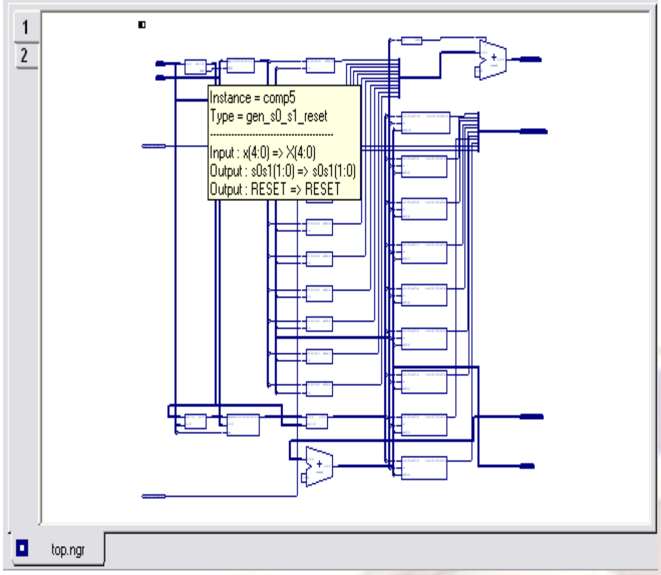

Instance = k0
Type = fin
Input : a(24:0) => a(24:0)
Output : I1(4:0) => p1(4:0)
Output : I2(4:0) => p1(9:5)
Output : I3(4:0) => p1(14:10)
Output : I4(4:0) => p1(19:15)
Output : I5(4:0) => p1(24:20)
top.ngr

Figure 6(b)



Instance = comp5
Type = gen_s0_s1_reset

Input : x(4:0) => X(4:0)
Output : s0s1(1:0) => s0s1(1:0)
Output : RESET => RESET

Figure 6(c)

Figure 6 RTL schematic view of the design Flow. 6(b), 6(c) are the internal structures of the top module

Device utilization summary:

Selected Device : xc3s250e-5tq144

1. Number of Slices	: 45 out of 2448	01%
2. Number of 4 input LUTs	: 80 out of 4896	01%
3. Number of bonded IOBs	: 36 out of 108	33%

The RTL schematic view of the design process is clearly shown in the figure 6. The internal structure of the proposed design is shown in figure 6(b) and 6(c).

The internal structure provides a clear view of the every component that is being used and the interlinks or connections between each is defined in it. The overall design uses the components that are available with in the target design and this reduces the need of the additional components, which further reduces the area occupied.

CONCLUSION

The implementation process of an advanced LUT design approach and the simulation results are neatly explained in this paper. It is clearly shown that the processing speed increases and there is no variation in terms of no. of slices used or area occupied with an increase in the number of the input bits for computing an arithmetic function with this method. As the input word size increases in the general LUT, the memory size is also increases. This requirement of additional memory size is overcomes with the specified new method. In the result part, the simulated result for an 18-bit input and a 25-bit input is shown. Even the input word length increases, the device utilization remains the same with the same processing speed and hence the computation time is reduced compared to the with the general LUT method. The simulation result shows that the LUT multiplier based design involves half the memory complexity of the usual LUT multiplier based design. Along with this the processing speed and the computation time is increased.

FUTURE SCOPE:

Even the processing speed increases, there are little variations in terms of time and power consumed. The power consumption is high when the n-input bits increases. Overcoming these variations can further enhance the system performance.

REFERENCES

[1]. Jason H. Anderson, Qiang Wang, “Area-Efficient FPGA Logic Elements: Architecture and Synthesis” in IEEE 2011 page no 369-375.

[2]. P. K. Meher, “New approach to LUT implementation and accumulation for memory-based multiplication,” May 2009.

[3]. Po-Yang Hsu, Ping-Chuan Lu, Yi-Yu Liu, “An Efficient Hybrid LUT/SOP Reconfigurable Architecture” in IEEE 2011 page no 173-176.

[4]. P. K. Meher, “New look-up-table optimizations for memory-based multiplication,” in Proc. ISIC, Dec. 2009.

[5]. *Stratix-IV FPGA Family Data Sheet*, Altera, Corp., San Jose, CA, 2010.

[6]. *Virtex-6 FPGA Data Sheet*, Xilinx, Inc., San Jose, CA, 2010.

[7]. *SPATARN-3E FPGA Family: Introduction and ordering information* DS312 (V3.8) August 26, 2009