High Speed and Performance analysis of Multiplier in Field Programming Gate Array

M Gowthami 1, Kekkesan Jalall S 2, Tony Aby Varkey M 3, K Kiruthika 4

1,2,3,4 Assistant Professor(ECE) - School of Engineering, Presidency University, Bengaluru.

gowthami.m@presidencyuniversity.in, kekkesan@presidencyuniversity.in, tonyaby.varkey@presidencyuniversity.in, kiruthika@presidencyuniversity.in

Abstract. This paper reads pipelined increase procedures for execution on FPGAs with accentuation on the usage of FPGA equipment asset. Execution of multiplier usage are estimated for monetarily accessible FPGA designs where two inborn issues are presented and examined. These being the lopsidedness of basic interconnect delay between broad directing and static convey interconnects, and the measure of FPGA rationale region utilized and its helpless usage. For every one of these issues proposals are proposed and researched.

Keywords: Look Up Table (LUT), FPGA, Ripple Carry Adder (RCA), Carry Look Ahead adder (CLA).

1 Introduction

Duplication on FPGAs is viewed as a costly activity. For high throughput multiplier executions where the outcome is determined in a pipelined equal style, huge Logic Cell (LC) tallies are required. The coming of quick convey rationale has permitted multiplier executions to accomplish reasonable paces for numerous DSP tasks. Anyway for expanding bit size duplication tasks, high LC tallies despite everything give poor FPGA gadget usage constraining FPGAs from a more extensive scope of DSP or number juggling serious applications. A few strategies have been proposed to give better use of the FPGA assets while executing augmentation. Mintzer [6] noticed that for fixed coefficient increase, by utilizing circulated number juggling and stockpiling look-into data in the Look-Up Table (LUT) component of LCs inside the FPGA, diminished rationale usage were accomplished. Run-Time Reconfiguration (RTR) has been utilized to refresh the substance of LUTs to give a new multiplicand esteem. Past work has indicated reconfigurable multiplier executions on the XC6200 arrangement FPGA [7,1]. Reconfiguration of LUT content is performed through the SRAM arrangement interface utilizing pre-figured design information. An epic methodology named the Self-Configurable augmentation strategy was introduced by the creator and ElGindy [10] for the Xilinx XC4000 arrangement by playing out the LUT reconfiguration on chip. On new information, the arrangement information is registered on chip and put away into the LUTs in equal. Other work to improve equipment productivity for multiplier executions recommends installing number juggling explicit Flexible Array Blocks (FABs) fit for actualizing a 4 × 4 bit multiplier, inside a traditional FPGA structure [2]. However, every one of these methods brings about a tradeoff. For the fixed coefficient procedure coefficient esteems can’t be refreshed. Reconfiguration of put away values requires extra time where the circuit is commonly disconnected while setup happens. Moreover, for a FPGA with
various rationale cell types, a division in equipment use between the cell type happens. This paper presents two choices for performing duplication on FPGA designs.

2 Designing Multipliers

2.1 Ripple Carry Adder

The core of each advanced PC is the CPU. The core of each CPU is the ALU. What’s more, the core of each ALU is the viper circuit. Adders are crucial. When you can include, you can take away by including a supplement, and you can likewise increase by tallying how often you include. It’s nothing unexpected then that FPGAs are injected without any difficulty the structure, and improve the presentation of adders. A full bit adder can be built of as not many as five entryways as appeared here. In a FPGA, we could actualize this in a three-input part or four-input parcel in the event that we had two outputs. In a few cases, the part engineering will have an extra yield for the do bit, separate from the whole out bit. This is the consequence of execution of a adder in an Altera MAX 10 FPGA. Appeared here utilizing the RTL watcher device which is a bit of the Quartus Prime FPGA plan suite. It looks simply like the entryway chart demonstrated previously. RTL represents enrolled move rationale or enlisted move level. RTL depends on the idea that development of information through advanced rationale can be seen as rationale charts with articulations speaking to the information factors that are put away in registers. It is commonly a more significant level portrayal than the entryway level. So this is the consequence of a usage of a turn in an Altera Max 10 FPGA, appeared here utilizing the innovation map watcher tool. It shows the execution is made utilizing two three info LUTs. We can utilize full 1-bit adders as parts to make end bit adders by associating them together. Here we see the association for a 4-bit adder comprised of four full 1-bit adders. This is known as the ripple carry adder. It is quick to structure, in any case, the wave convey adder is generally delayed on execution since each full adder must sit tight for the convey bit to be determined from the past full adder.

The total delay through ripple-carry adder to the sum out is expressed by the equation

\[ T_{adder}(n) = (n-1) \times T_c + T_s \]
\[ T_{adder}(n) = (n-1) \times 3D + 2D \]
\[ T_{adder}(n) = (3n-1)D. \]
2.2 Carry look Ahead Adder

A Carry Lookahead (Look Ahead) Adder is made of various full-adders fell together. It is utilized to include two paired numbers utilizing just basic rationale doors. The figure beneath shows 4 full-adders associated together to create a 4-piece convey lookahead snake. Convey lookahead adders are like Ripple Carry Adders. The thing that matters is that convey lookahead adders can compute the Carry bit before the Full Adder is finished with its activity. This gives it a favorable position over the Ripple Carry Adder since it can include two numbers together quicker. The downside is that it takes more rationale. You will discover there is regularly a harmony between speed of execution and assets utilized when structuring FPGAs and ASICs. There are two models for each VHDL and Verilog demonstrated as follows. The first contains a basic convey look ahead snake comprised of four full adders (it can include any four-piece inputs). The subsequent model uses a conventional that makes a convey look forward viper that acknowledges as an info boundary the WIDTH of the sources of info. In this way it is versatile for any information widths. Carry Look Ahead Adder is quickest snake contrasted with Ripple convey Adder. With the end goal of convey Propagation, Carry look Ahead Adder develop Partial Full Adder, Propagation and age Carry square. It evade Carry engendering through each adder. In request to actualize Carry Look Ahead Adder, first execute Partial Full Adder and afterward Carry rationale utilizing Propagation and age Block.

Where, D is the delay through one gate and n is the number of bits in the adder. The Above equation, if you go through 32bit adder we need 95 gate delays. We can use more sophisticated circuits to create better performingadders with less delay. One such possibility is the Carry Look Ahead addershown here. The Carry bit can be calculated as generate and propagate terms. we have

\[ C_i + 1 = G_i + P_i C_i \]

In which the \( G_i = A_i B_i \)
and \( P_i = A \ XOR \ B \).
Now the delay is roughly equal to n, a big improvement over the ripple carry adder, but at the cost of a much more complex carry bit calculation.

### 2.3. FPGA Technological View

This is the after effect of the execution of the 4-bit ripple carry adder in the Altera MAX 10 FPGA, appeared here utilizing the Technology Map Viewer instrument. It shows the execution is made of a course of four sets of three info LUTs. The deferral through this circuit will be just four LUT delays, not 11 entryway delays as dependent on the postpone condition. Besides, we may get a greater amount of a preferred position on the off chance that we utilize a further developed FPGA design.

![Altera MAX 10 FPGA technological view](image)

*Figure 3: Altera MAX 10 FPGA technological view*

### 3. Designing Multipliers in FPGA

If Increasing 2 numbers is a typical computerized rationale or ALU work that can take impressive measures of hardware. This has been an issue for architects for quite a long time. Luckily, there are an assortment of approaches to execute multiplier circuits inside FPGAs. Here's a model, a 2-bit by 2-bit parallel multiplier. Can we, when all is said in done, make multipliers with gates? Indeed, utilizing cluster multipliers. It works like duplication by hand, which incorporates halfway item age and incomplete item summation. In any case, the quantity of gates increments as $n^2$. Past 4x4 bits, a consecutive plan is progressively proficient, at that point the adder multiplier. There are numerous approaches to fabricate a multiplier in a FPGA, combinational circuits, and quick however huge. Successive move and include, a state machine approach that is little yet moderate. Claim to fame calculations like Booth's Algorithm, the Dadda Multiplier, or the Wallace Tree Multiplier, quick and little however intricate A 2bit by 2 bit binary multiplier in digital Logic gates.
There are several ways to build a multiplier in an FPGA:
1. Sequential Shift and adder (State machine approach)
2. Combinational circuits
3. Algorithms like booth’s or Dadda or Wallace tree multipliers
4. Memories – look up tables
5. Combination of above
6. Hard multiplier blocks

Table 1: Gates Approximation

| bits | Approximate gates |
|------|-------------------|
| 2x2  | 8                 |
| 3x3  | 54                |
| 4x4  | 96                |
| 5x5  | 150               |
| 6x6  | 216               |
| 7x7  | 294               |
| 8x8  | 384               |

3.1 32-bit pipemult schematic
In this paper you utilize the RTL, Technology Viewers and Chip Planner to examine a FPGA plan in Quartus Prime. We can, accumulate plan in Quartus Prime so you can track and play out each progression route. We will see a plan at the RTL level, see a plan at the innovation level, break down the plan utilizing the Chip Planner and afterward dissect the plan with the Power Play Power Analyzer. Quartus Prime gives you various instruments that can be utilized to investigate the aftereffects of the plan. You ought to have a smart thought of how your plan should end up yet in some cases the compiler will give you some sudden outcomes.

3.2. Simulation results of 32-bit multiplier
Figure 6: Simulation output for 32bit pipelmult.

FPGA creators utilize the examination apparatuses to affirm the plan result is the thing that they anticipated. One gathering of apparatuses utilized for examination are the watchers. The RTL watcher, the innovation map (post planning) watcher, the innovation map (post fitting) watcher and the state machine watcher. The netlist watchers are graphical apparatuses that allowed you to see the consequences of amalgamation and furthermore spot and course. The watcher shows the multiplier RAM and D flip failure register on the yield a lot of like your schematic. The information ports are on the left and the Q alpha port is on the right. We can inspect what segments make up every one of the primary squares by extending the netlist guide on the left sheet. the postings of examples, natives and ports. On the off chance that we click on the ports triangle, it grows to show sources of info and yields. Tapping on information sources grows all the information ports. On the off chance that we click on information it will grow to show each piece in the transport. Snap the ports triangle again to fall this tree. In the event that you click on occurrences you'll see the multiplier and the RAM IP examples. Continue tapping on the multiplier case on each level until the whole tree is extended. we will see that the multiplier is made of two nuclear units that are natives in the FPGA design. The examples triangle to fall this tree. Presently in the schematic, click on the in addition to sign in the upper right corner of the multinst square. This square will at that point extend to show the LPM under mult part. on the in addition to sign to grow the part. It grows again to show a crude. Snap on the in addition to sign again to extend to the nuclear pieces, the mac_mult1 and the mac_out2 blocks. This shows you the most fundamental degree of the multiplier occasion. As you did this, the netlist guide ought to have extended too. Snap the less signs in the squares until you have gotten back to the primary RTL see. Presently the in addition to sign in the upper right corner of the RAM block. It shows three crude squares, the simultaneous RAM block itself and a read address register and yield register.

FPGA device 10M08DAF484C8GES family Max 10 which shows the synthesis report below. FPGA natives including each cradle for each info and yield to a pin. Clearly there's significantly more detail in proposed method.

| Device Name         | Frequency in MHz | No of Slice | No of LUTs | No of IOBs |
|---------------------|------------------|-------------|------------|------------|
| 10M08DAF484C8GES    | 3.58             | 64          | 0          | 0          |

4. Conclusion

In this 32-bit pipelined multiplier is compared with existing multiplier of two stage 2LUTs and 4LUTs of the multiplier. In this paper we used three stage pipelined multiplier to get minimum delay that is 2.826ns. The three stage pipelined multiplier has been reduced the partial products into half. The overall performance of this pipelined multiplier is reduce the delay by performance basis. The proposed pipelined multiplier is giving the less delay compare to existing methodology.

References
1. “High Speed, Power and Area efficient Algorithms for ALU using Vedic Mathematics”, V.Vamshi Krishna,S.Naveen Kumar / International Journal of Scientific and Research Publications, ISSN: 2250-3153, Vol. 3, Issue 7, July 2012.
2. “Design of High Speed Multiplier using Vedic Mathematics”, Surbhi Bhardwaj,Ashwin Singh Dodan / International Journal of Engineering Research and General Science ISSN: 2091-2730 Vol.2, Issue 4, June-July 2014.
3. “High Speed Efficient Karatsuba-Ofman Pipelined Multiplier for low Contrast Image Enhancement”,Triveni.K.S,E.Elavarasi/International Journal of Engineering and Advance Technology (IJEAT), ISSN:2249-8958, Vol.-2,Issue-5,June 2013.
4. “Arithmetic Unit Implementation Using Delay Optimized Vedic Multiplier with BIST Capability”, Abhishekh Gupta/ International Journal of Engineering and Innovative Technology (IJEIT) Volume 1, Issue 5, May 2012.
5. International Journal of Advance Research in Engineering, Science & Technology e-ISSN: 2393-9877, p-ISSN: 2394-2444 Volume 4, Issue 4, April -2017 All Rights Reserved, @IJAREST-2017 Impact Factor (SJIF): 4.542 160 Comparative Study on Pipelined and Parallel Multiplier using Vedic Mathematics: A Review
6. Comparative Study on Pipelined and Parallel Multiplier using Vedic Mathematics: A Review. Available from https://www.researchgate.net/publication/317259330_Comparative_Study_on_Pipelined_and_Parallel_Multiplier_using_Vedic_Mathematics_A_Review [accessed Jul 29 2020].
7. E. H. Miller, “A note on reflector arrays (Periodical style—Accepted for publication),” IEEE Trans. Antennas Propagat., to be published.
8. Vijayalakshmi, R. Seshadri, Dr. S. Ramkrishnan, “Design And Implementation Of 32-Bit Unsigned Multiplier Using CLAA And CSLA” International Conference on Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System, DOI 10.1109/ICEVENT.2013.6496579, Jan 2013.
9. Soniya, Suresh Kumar, “A Review of Different Type of Multipliers and Multiplier Accumulator Unit”, International Journal of Emerging Trends and Technology in Computer Science, Vol. 2 No. 4, August 2013.
10. Abdullah-Al-Kafi, Atul Rahman, Bushra Mahjabeen, Mahmudur Rahman, “An Efficient Design Of FSM Based 32-Bit Unsigned HighSpeed Pipelined Multiplier Using Verilog HDL” 8th International Conference on Electrical and Computer Engineering, DOI 10.1109/ICECE.2014.7027026, Dec 2014