STM JOURNALS

Journal of
VLSI Design Tools & Technology

ISSN: 2249-474X (Online)
ISSN: 2321-6492 (Print)
Volume 14, Issue 1, 2024
January–April
DOI (Journal): 10.37591/JoVDTT

http://engineeringjournals.stmjournals.in/index.php/JoVDTT/index

Research JoVDTT

# Hardware Synchronization for Embedded Multi-core Processors

Tejasvini Bansode[1,*], Anjali Pise[2]

***Abstract***

*The focus is not on whether multi-core systems are becoming prevalent, but on how microcontroller architectures should be designed to comply with stringent field requirements. This paper illustrates the transition from single to multiple cores, ensuring coherence and consistency in shared memory through hardware mechanisms. Hardware barriers are implemented to allow for point-to-point synchronization between processor cores. Practical experiments center on the initial stages of transforming single-core systems into dual-core ones by utilizing an FPGA development board that has two PowerPC processor cores. Hardware solutions outperform software when both best- and worst-case situations are evaluated, and synchronization primitives are thoroughly benchmarked. Additionally, the study shows that when many cores use address-sensitive locking on shared memory to take advantage of inherent parallelism, dual-ported memory performs better than single-ported memory. One effective strategy involves using the results from the Booth encoder to correct the inaccuracies caused by the truncated modified Booth multiplier. Cutting off the lowest part of these partial products is a simple approximation technique that reduces latency and hardware overhead; we refer to this design approach as fixed-width multiplier design. A variety of error compensation techniques have been developed recently to improve multipliers with fixed widths' precision. This approach has shown to decrease errors by almost 50% when compared to a truncated design that lacks error correction.*

**Keywords:** PowerPC processor, FPGA, digital signal processing, SNR, RPR, VOS, VLSI systems,

## INTRODUCTION

Computer arithmetic is necessary in many applications using digital signal processing (DSP). Multipliers often regulate the speed at which a DSP system functions since they are more intricate than adders and substractors. While high precision is typically deemed essential, other factors like delay, hardware complexity, and power consumption are also critical considerations in design. Applications like media processing, recognition, and data mining—where error tolerance is permitted—can make use of approximate arithmetic units. One common task in its design is to handle the accumulation of incomplete products, which can represent a bottleneck in the approximate multiplier's operation. A straightforward approximation method for minimizing latency and hardware overhead involves cutting off the lowest portion of these partial products; [1-4] We call this design method fixed-width multiplier design. Recently, a number of error compensation strategies have been developed to increase the precision of multipliers with fixed widths. Overall, these advancements in error compensation techniques are enhancing the accuracy and performance of fixed-width multipliers in various applications. Additionally, a streamlined sorting network has been employed as

***Author for Correspondence**
Tejasvini Bansode
E-mail: tejasvini.bansode12@gmail.com

[1]PG Student. Department of Electronics Engineering, SKN Sinhgad College of Engineering, Korti, Pandharpur, Maharashtra, India
[2]Assistant. Professor, Eletronics Engineering, SKN Sinhgad College of Engineering, Korti, Pandharpur, Maharashtra, India

Received Date: April 29, 2024
Accepted Date: May 16, 2024
Published Date: May 24, 2024

**Citation:** Tejasvini Bansode, Anjali Pise. Hardware Synchronization for Embedded Multi-core Processors. Journal of VLSI Design Tools & Technology. 2024; 14(1): 33–42p.

 

a crucial step in the error correction process to build a more accurate version of the fixed width modified Booth multiplier [5]. Additionally, a fixed-width Booth multiplier's quantization errors have been approached using a probabilistic methodology. The research literature has also addressed an adaptive conditional-probability estimator [6]. An exhaustive simulation takes a lot of time, but a probabilistic analysis yields the estimator. One Karnaugh Map (K-Map) entry has been modified to suggest roughly a 2 * 2-bit multiplier. Instead of using four bits, the 2 * 2 multiplier's output uses three. The fundamental building block for more complex operand multipliers is this roughly 2 * 2-bit multiplier. With power dissipation savings ranging from 31.78 to 45.4 percent, this design achieves an average error of just 1.39 to 3.32 percent. Another way to do this is to make approximately 4 * 4-bit Wallace multipliers using counters or compressors found in Dadda trees or Wallace trees an imperfect 4: 2 counter is employed in this instance. By adjusting the output values, two approximately 4×2 compressors were produced using the truth table. Four distinct designs for the Dadda tree of an 8×8 multiplier were created using these compressor approximations, which resulted in significant power dissipation and latency reductions. A unique approximate multiplier was developed by employing an approximation adder that eliminates carry propagation across partial products. This approach resulted in a 20% reduction in latency and 69% power savings for an 8×8 bit multiplier [7–10]. The majority of current approximation multiplier designs concentrate on the partial product accumulation phase and are mainly designed for unsigned number operations. Although signed multiplication is frequently performed using the Booth technique, this work explores Booth multiplier [11–14]. The radix-8 approach is discussed here because it is effective in producing fewer partial products, which reduces the number of adders needed to aggregate these partial products for high-speed operations. Often used is the radix-4 recoded Booth method. The hardware-efficient radix-8 recoding technique is rare since it requires more time to handle odd multiples of the multiplicand. To be more precise, multiplying the multiplicand by three requires an extra step and an additional adder for preliminary processing, which could cause major delays because of carry propagation [15–18]. Comparing this additional adder to the radix-4 approach, which creates multiplicands by simply shifting, can result in a latency increase of 10–20%. In order to solve this problem, this study presents approximation designs for a Booth multiplier that include an approximation scheme. These designs address partial product accumulation as well as the development of recoded multiplicands.

An original design of a 2-bit approximate recoding adder aimed to minimize the excess latency previously seen in radix-8 schemes in order to speed up the radix-8 Booth algorithm. By calculating the sum of partial products using a Wallace tree, this technique drastically lowered the addition time. Next, a truncation technique was used on the least significant partial products in order to minimize delay and power usage. Next came the introduction of two signed 16 * 16 bit approximation radix-8 Booth multipliers, called approximate Booth multipliers 1 and 2 (ABM1 and ABM2). It was demonstrated by simulated findings that ABM1 outperforms the exact radix-8 Booth by 20%. Ultimately, by including the ABMs into a low-pass FIR filter, their superior performance over alternative approximation multipliers that have been described in technical literature was demonstrated.

## EXISTING SYSTEM

Voltage over scaling (VOS) is a technique used in the current system to aggressively lower supply voltage without sacrificing throughput. However, it has been found that VOS leads to a considerable decrease in the signal-to-noise ratio (SNR). To address this issue, a new algorithmic noise-tolerant (ANT) approach that combines a reduced-precision replica (RPR) and the VOS main block has been devised. This combination saves a significant amount of energy and efficiently reduces soft errors. The concept of ANT is extended to a system level and various designs for its implementation are considered. Low-power signal processing systems can be built to function with energy efficiency higher than those of existing systems by merging ANT with VOS [19]. Soft Digital Signal Processing is the low-power application of Adaptive Near-Threshold (ANT) and Voltage and Frequency Over scaling (VOS) technologies. The difficulties of energy saving and dependability are addressed simultaneously in Soft

 

DSP systems. When clock frequency hits a critical number, it can be used to construct high-throughput systems that work similarly to VOS. Moreover, by using ANT to lessen the effects of deep submicron (DSM) noise, such as cosmic rays, ground bounce, crosstalk, or process variations, error-tolerant digital signal processing systems can be produced.

The paper "On energy-efficiency bounds of DSM VLSI systems in the presence of noise" makes a compelling case for the necessity of creating noise-tolerance-focused design strategies in order to concurrently address energy efficiency and dependability. According to the 2001 International Technology Roadmap for Semiconductors, error-tolerance will be a major design problem during the next ten years. Within design hierarchy, algorithmic and circuit strategies for noise-tolerance have been developed. To mitigate faults at the system level in digital signal processing systems, algorithmic noise-tolerance (ANT) is one such method. Furthermore, voltage over scaling (VOS), a proactive low-power technique, was suggested in this context.

Under advanced network topologies (ANT), the current system presents a novel technique called Reduced Precision Redundancy (RPR). This method achieves significant energy savings while also addressing soft-errors with effectiveness [20]. RPR is the process of identifying and correcting mistakes that arise at the output of a Digital Signal Processing (DSP) system by employing a lower precision duplicate of the main DSP (MDSP).

This RPR-based ANT strategy differs from prediction-based error-control (PEC) or adaptive error-cancellation (AEC) strategies. RPR is unique among ANT installations because of its specialized design, which is difficult to duplicate. RPR designs in ANT systems have a significant hardware complexity despite their ability to work quickly. Even of its complexity, the RPR design is still a common option in ANT systems because of its simplicity. RPR integration, however, can result in higher power and area overhead requirements [21].

## PROPOSED SYSTEM

The proposed system suggests using a fixed-width RPR in favor of the full-width RPR block, which provides a more straightforward method. The fixed-width RPR allows computation errors to be corrected with less power and overhead. Partial product weight analysis, statistics, and probability are used to estimate the approximation compensation vector for a more exact RPR design. It ensures that the compensating circuit has no effect on the critical path delay. As a result, the circuit size, power consumption, and crucial supply voltage are reduced, resulting in an ANT design for the system [22]. In the ANT method, the principal digital signal processor (MDSP) and the error correction (EC) block are both involved.

To improve error compensation accuracy, the system compensates for truncation mistakes using a variable correction value. The system builds an error compensation circuit primarily employing partial product terms with the largest weight in the least significant segment. The error compensation approach uses probability, statistics, and linear regression analysis to estimate the compensation value [23]. Since the compensation vector in the partial product terms with the highest weight in the least significant segment is instantly inserted into the fixed-width RPR, minimizing hardware complexity, there is no need for extra compensation logic gates. In order to minimize compensating errors, the system takes into account the effect of truncated products utilizing the second most important bits. The system incorporates an error compensation circuit that uses a little input correction vector to address any remaining defects. To avoid any delays in the critical path, the compensation circuit is positioned in the fixed-width RPR's non-critical path. In comparison to the full-width RPR design, the suggested fixed-width RPR multiplier works better in terms of power consumption, circuitry size, and signal-to-noise ratio (SNR). The system aims to further minimize compensation errors by accounting for the impact of truncated products with the second most important bits on the error compensation. It suggests utilizing a straightforward minor input correction vector in an error compensation circuit to rectify the residual error. To ensure the fixed-width RPR's non-critical path, the compensating circuit is placed there [24].

 

In addition to offering a higher SNR than the full-width RPR design, the recommended fixed-width RPR multiplier also consumes less power and circuitry area. Moreover, the system suggests replacing the ANT architecture's fixed-width RPR block with a full-width RPR block. This substitution can improve computation precision, reduce power consumption, and reduce area overhead in RPR in addition to offering improved SNR, increased area efficiency, reduced operational supply voltage, and decreased power consumption in accomplishing the ANT design. In an ANT multiplier, the system illustrates the fixed-width RPR-based ANT design [25].

## FLOW DIAGRAM

The flow diagram for synchronization of multicore processor is shown in Figure 1.

## MODULE DESCRIPTION

### 24-bit Multiplier

A 24-bit multiplier is designed by first creating the basic gate logic, followed by inputting 12*12 bits to generate 144 AND gates. These gates are then utilized to create partial products, enabling the execution of the 24-bit multiplier logic. The output of this process results in a 24-bit product.

### Reduced-precision Replica (RPR) Block

#### *Truncated*

The RPR block executes a shortened operation while taking into account the most significant part (MSP) of the 24-bit. The MSP is separated into least significant parts (LSP) and MSP. The LSP consists of an input correction vector (ICV β) and a minor input correction vector (MICV α) [26]. The LSP is then removed because the RPR block's output is in 12-bits; for this reason, it is referred to as the truncated portion.

 

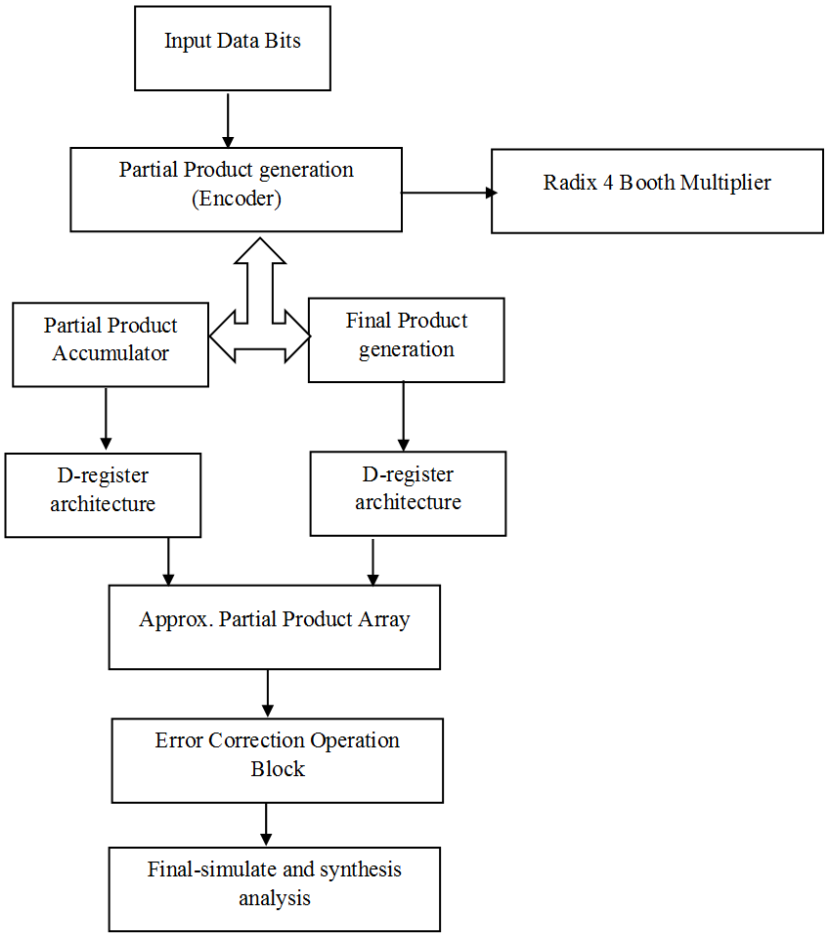


**Figure 1.** Flow Diagram of Hardware synchronization of multicore processor.

In the 24-bit input (p0-23), the truncated part is from (p12-23). In this section, a series of p11 half adders and full adders are converted into an a0 block. The a0 block contains a combination of AND gates and OR gates, producing two outputs. The outputs from the a0 block are fed as inputs to the truncated block in a series. The final result from the RPR block is in 12-bits.

**Register Element**

In this circuit, a D flip-flop element is utilized within a block register. The flip-flop is regulated by both clock and reset signals. Two D flip-flops are employed in this setup: the first flip-flop receives a 24-bit input from the multiplier output and produces a 24-bit output, while the second flip-flop receives a 12-bit input from the truncated output and generates a 12-bit output.

**Comparison Block**

In this block, an XOR logic is employed to compare the two inputs from two flip flops. The first flip flop provides a 24-bit input, while the other provides a 12-bit output. Consequently, the comparison block yields a 12-bit result.

**Error Correction Block**

By using the comparison block's output as the error correction block's input, the error threshold is adjusted. There isn't an error if the input is equal to zero. If the input is not equal to zero, then a truncated output is generated. The outputs from two D flip-flops are input to a multiplexer, which is controlled by a selection line. The output of the error correction block serves as the selection line. The final result includes the simulated and synthesized outputs.

 

## SYSTEM REQUIREMENTS

### Hardware Requirements

- 1GB DDR RAM
- Pentium IV 2.7 GHz
- A 250 GB hard drive

### Software Prerequisite

- XP is the operating system.
- Vivado 2018.2 is the version of XILINX.

## SET UP THE SIMULATION

You can simulate your design using the ISim standalone flow w first create a project in ISE Project Navigator. In order to create an executable for simulation, one must manually create an ISim project file, which is then used by the fuse command. Once this is finished, you may see the ISim graphical user interface by starting the simulation software.

1. Create a manual ISim project file
   Typically, an ISim project file looks like this:
   {.v|.vhd} in VHDL verilog, where:
   - verilog|A Verilog or VHDL file is designated as the source with vhdl. Include Verilog or VHDL source files.
   - specifies the library that needs to be utilized in order to put together a particular source on a certain line.

*Note:* While several Verilog source files can be specified on a single line, only one VHDL source can be specified at once.

1. Open the script folder in order to create an ISim project file for the lesson design.
2. Use a text editor to open the simulate_isim.prj project file.

There are gaps in the project file.

3. Using the syntax guidelines, list the missing sources.

Absent sources:

- Source file for VHDL: drp_dcm.vhd. It has to be built using the /work library.
- The file drp_tb_pkg.vhd is the VHDL package. It must be compiled using the /drp_tb_lib..

## IMPLEMENT THE EXECUTABLE SIMULATION

### Use the Fuse Command

The following describes the standard syntax for using the Fuse tool:

fuse -incremental -o \library.top_unit> -prj

The parts are broken out as follows:

- incremental: instructs Fuse to only assemble the files that have changed since the last assemble.
- prj: indicates which ISim project file should be used as input.
- o: finds the executable file that is the output of the simulation.

Take the following actions to use fusion to parse, create, and elaborate the tutorial design:

1. Open the/scripts folder from the downloaded files.
2. Locate the fuse_batch.bat file in a text editor.
3. This fuse instruction has a gap in it. With the above-mentioned syntactic details in mind, modify the command line as follows:

 

   a. Use progressive compilation.
   b. Verify that simulate_isim.prj is the project file.
   c. Simulate_isim.exe is the simulation executable.
   d. Use work drp_demo_tb as the top design unit in the simulation.
4. After saving the batch file, close it.
5. Use the ISE Command prompt to enter the/scripts folder, then run the fuse_batch.bat file to start fuse.

**SIMULATE THE DESIGN MANUALLY**

In this step of the simulation process, you will initiate the ISim GUI by executing the simulation executable, which was created using the fuse command in the previous section called "Build the Simulation Executable." After completing this step, you will be able to study the design more closely because you will have access to the ISim GUI.

Using the Simulation Executable

The command syntax to run the simulation executable is as follows:

isim_exe –view –gui –wdb

where:

- gui : Starts ISim using the GUI.
- view : Causes the ISim GUI to open the designated waveform file.
- wdb : Indicates the filename of the simulation database's output file.

**LAUNCH SIMULATION**

Take these actions to start the simulation:

1. Open the downloaded files and navigate to the/scripts folder.
2. Use a text editor to open the simulate_isim.bat file. At first, this batch file will be empty.
3. Change the batch file with the following parameters, the provided syntax rules as a reference:
   a. Type simulate_isim.exe in the simulation executable's name field.
   b. the GUI mode of launch.
   c. Set simulate_isim.wdb as the output name for the simulation database.
4. After making the required changes, save and exit the file.
5. To launch the simulator, open the simulate_isim.bat file and run it from the ISE Command prompt.

**RESULT**

Only cores that actually require access are taken into account by the mechanism. Awaiting processor cores are set aside till it's their turn. The most challenging scenario arises when all accessible cores need to access the shared memory simultaneously, leading to disparate wait times. Following the loading of a memory block's higher and lower addresses into the corresponding MACtrl registers, address-sensitive locking is triggered in order to prevent any bias or discrimination towards a processor core as shown in Figure 2. This is accomplished through the use of a dynamic priority system to decide the access order. Promptly, the controller tries to lock the memory block. By definition, global locking and address-sensitive locking are incompatible, meaning that the shared memory cannot be accessed by either access mechanism at the same time.

An order was issued to simulate the overlapping of memory blocks in an address-sensitive locking scenario. Each CPU core in this configuration reserves a block of memory as shown in Figure 3, reads it, and then moves it by a random offset. The memory blocks are moved in opposing directions during the relocation process, which causes conflicts between the processor cores as shown in Figure 4.

 

## CONCLUSION

A viable tactic entails leveraging Booth encoder outcomes to rectify inaccuracies stemming from the truncated modified Booth multiplier. The research also reveals that in scenarios where multiple cores employ address-sensitive locking on shared memory to capitalize on inherent parallelism, dual-ported memory outperforms single-ported memory. Trimming the least significant portion of these partial products constitutes a straightforward approximation method that minimizes latency and hardware overhead; this design methodology is denoted as fixed-width multiplier design. Recent advancements have introduced a range of error compensation techniques aimed at enhancing the precision of fixed-width multipliers. This approach has demonstrated a nearly 50% reduction in errors compared to a truncated design lacking error correction mechanisms.

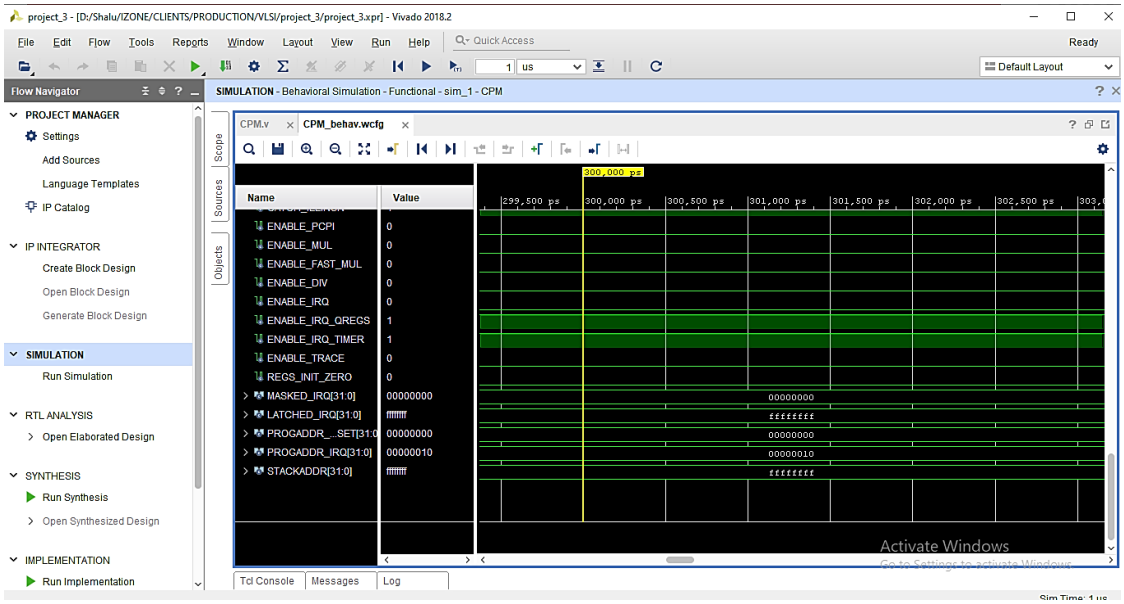


**Figure 2.** Behavioral Simulation.

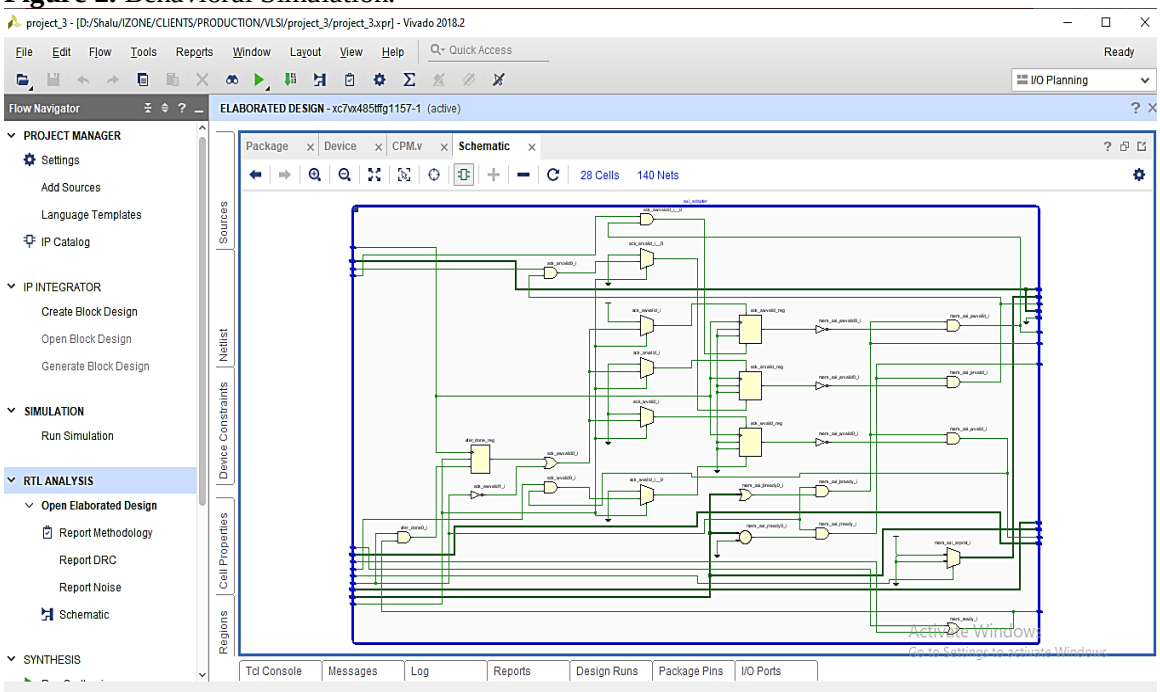


**Figure 3.** Schematic of Data.

 

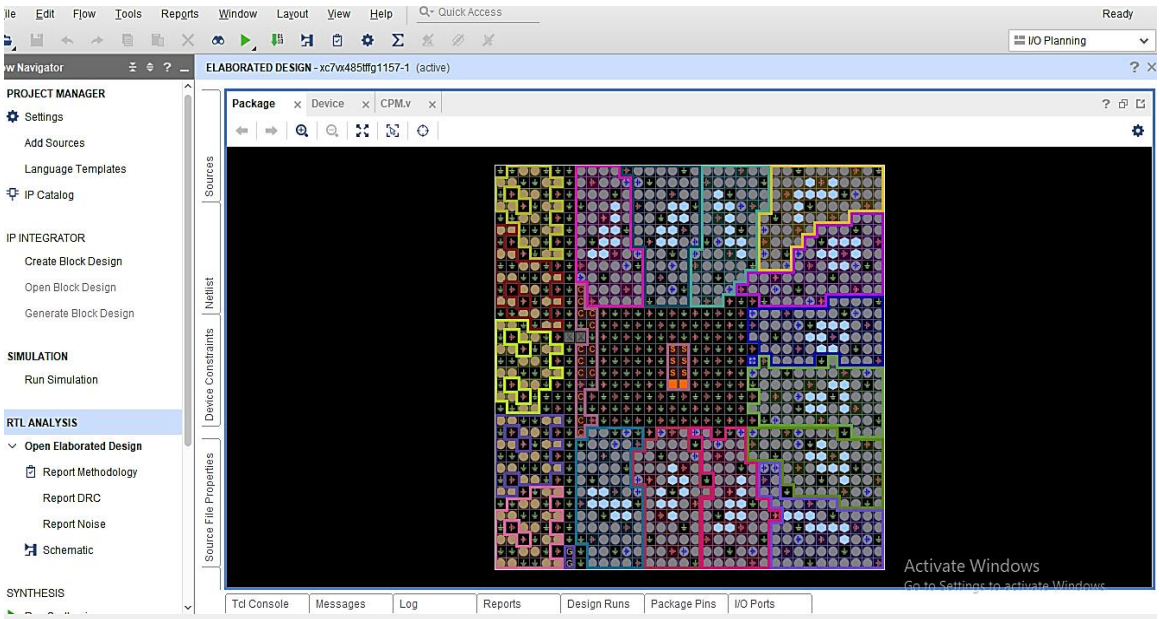


**Figure 4.** Elaborated Design.

## REFERENCES

1. Najem N. Sirhan, Sami I. Serhan "Multi-Core Processors: Concepts and Implementations" International Journal of Computer Science & Information Technology (IJCSIT) Vol 10, No 1, February 2018
2. H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-in spired imprecise computational blocks for efficient VLSI imple mentation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, pp. 850-862, 2010.
3. Mulani, A. O., & Mane, P. B. (2017). Watermarking and cryptography-based image authentication on reconfigurable platform. Bulletin of Electrical Engineering and Informatics, 6(2), 181-187.
4. Deshpande, H. S., Karande, K. J., & Mulani, A. O. (2014, April). Efficient implementation of AES algorithm on FPGA. In 2014 International Conference on Communication and Signal Processing (pp. 1895-1899). IEEE.
5. Mane, P. B., & Mulani, A. O. (2018). High speed area efficient FPGA implementation of AES algorithm. International Journal of Reconfigurable and Embedded Systems, 7(3), 157-165.
6. Mulani AO, Mane PB. Area efficient high speed FPGA based invisible watermarking for image authentication. Indian journal of Science and Technology. 2016 Oct 24.
7. Mulani, A. O., & Mane, D. P. (2017). An Efficient implementation of DWT for image compression on reconfigurable platform. International Journal of Control Theory and Applications, 10(15), 1-7.
8. Mane, D. P., & Mulani, A. O. (2019). High throughput and area efficient FPGA implementation of AES algorithm. International Journal of Engineering and Advanced Technology, 8(4).
9. Mulani, A. O., & Mane, D. P. (2018). Secure and area efficient implementation of digital image watermarking on reconfigurable platform. International Journal of Innovative Technology and Exploring Engineering, 8(2), 56-61.
10. Deshpande, H. S., Karande, K. J., & Mulani, A. O. (2015, April). Area optimized implementation of AES algorithm on FPGA. In 2015 International Conference on Communications and Signal Processing (ICCSP) (pp. 0010-0014). IEEE.
11. Mulani, A. O., & Mane, P. B. (2019). High-Speed area-efficient implementation of AES algorithm on reconfigurable platform. Computer and Network Security, 119.

 

12. Mulani, A. O., & Mane, P. B. (2014, October). Area optimization of cryptographic algorithm on less dense reconfigurable platform. In 2014 International Conference on Smart Structures and Systems (ICSSS) (pp. 86-89). IEEE.
13. Mandwale, A., & Mulani, A. O. (2015, January). Different Approaches For Implementation of Viterbi decoder. In IEEE International Conference on Pervasive Computing (ICPC).
14. Mandwale, A., & Mulani, A. O. (2014, December). Implementation of Convolutional Encoder & Different Approaches for Viterbi Decoder. In IEEE International Conference on Communications, Signal Processing Computing and Information technologies.
15. Mulani, A. O., & Mane, P. B. (2017) Fast and Efficient VLSI Implementation of DWT for Image Compression. International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 5 Issue IX, pp. 1397-1402.
16. V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise Adders for Low-Power Approximate Computing," Proc. Int. Symp. Low Power Electronics and Design (ISLPED), pp. 1-3, 2011.
17. W. Liu, L. Chen, C. Wang, M. O'Neill and F. Lombardi, "Design and analysis of floating-point adders", IEEE Trans. Computers, vol. 65, pp. 308-314, Jan. 2016.
18. J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Computers, vol. 63, pp. 1760-1771, Sep. 2013.
19. A. Booth, "A signed binary multiplication technique," Quarterly J. Mechanics and Applied Mathematics, vol. 4, pp/236-240, June 1951.
20. O. MacSorley, High-speed arithmetic in binary computers, Proc. IRE, vol. 49, pp. 67-91, 1961.
21. K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low error fixed-width modified Booth multiplier," IEEE Trans. VLSI Systems, vol. 12, no. 5, pp. 522–531, 2004.
22. Sarda, M., Deshpande, B., Deo, S., & Karanjkar, R. (2018). A comparative study on Maslow's theory and Indian Ashrama system.". International Journal of Innovative Technology and Exploring Engineering, 8(2), 48-50.
23. Deo, S., & Deo, S. (2019). Cybersquatting: Threat to domain name. International Journal of Innovative Technology and Exploring Engineering, 8(6), 1432-1434.
24. Shambhavee, H. M. (2019). Cyber-Stalking: Threat to People or Bane to Technology. International Journal on Trend in Scientific Research and Development, 3(2), 350-355.
25. Deo, S., & Deo, D. S. (2019). Domain name and its protection in India. International Journal of Recent Technology and Engineering.
26. Sarda, M., Deshpande, B., Deo, S., & Pathak, M. A. (2018). Intellectual Property and Mechanical Engineering-A Study Emphasizing the Importance of Knowledge of Intellectual Property Rights Amongst Mechanical Engineers. International Journal of Social Science and Economic Research, 3(12), 6591-6596.

 