
	THE UNIVERSITY OF SHEFFIELD School of Electrical and Electronic Engineering 3rd Year Individual Project Interim report		
Student Name	Amaan Mujawar		
Project Title	Implement an Arithmetic Unit utilising Approximate Computing into RISC-V SoC		
Supervisor	Mr Neil Powell	Second Marker	

1. Background, Aims and Objectives

Approximate computing has become an emerging technique that reduces execution time, or power consumption by allowing a small degree of error in computations [1]. It has gained significant attention in digital circuit design, particularly as the demand for energy-efficient computing continues to rise. This approach is finding applications in areas such as machine learning, computer vision, web search, and data analysis. Many signal processing, image processing, and multimedia tasks are inherently error-tolerant and can produce results that appear indistinguishable to the human eye, even without the need for exact computations. By leveraging this error tolerance, approximate computing can be applied in such error-tolerant operations by providing meaningful results faster and/or with lower power consumption at the cost of reducing accuracy [2].

In today's digital era, the demand for high-performance computing has grown exponentially, driven by data-intensive applications such as machine learning, big data analytics, computer vision, and multimedia processing. Historically, this demand has been met by advancements in semiconductor technology, guided by Moore's Law, which predicts that the number of transistors on a chip doubles approximately every two years. This trend has enabled continuous improvements in computational power, energy efficiency and cost.

However, as transistor sizes approach their physical and economic limits, the rate of progress predicted by Moore's Law is slowing. With traditional scaling facing significant challenges, it is becoming increasingly difficult to achieve the necessary performance and energy efficiency gains through conventional means alone. This has created a need for new approaches that can complement traditional scaling.

One such approach is approximate computing, as discussed above. It offers a model shift in how computations are performed. In contrast to focusing solely on precision, approximate computing introduces a controlled level of errors in computation to reduce execution time and/or power consumption. This technique is particularly well-suited for applications in signal processing, image processing, and multimedia where exact results are often unnecessary, and error-tolerant outputs can be perceived as correct by humans.

By manipulating this inherent error tolerance, approximate computing provides an opportunity to overcome the limitations of traditional scaling and

continue to deliver improvements in computational efficiency. This project aims to explore the potential of approximate computing in digital circuit design by integrating existing methods of approximate computing to provide a hardware solution that balances performance, power consumption, and accuracy. As the limitations of Moore's Law become more apparent, the importance of innovating techniques like approximate computing continues to grow. By addressing these challenges, this project aims to contribute to the development of sustainable, high-performance digital systems that are capable to meet demands of future technologies.

The primary aim of this project is to design and implement a 32-bit Multiply-Accumulate (MAC) unit utilising approximate computing techniques, specifically tailored for integration into a RISC-V System on Chip (SoC). To achieve this, the project will follow a series of specific objectives. The first objective involves conducting a comprehensive literature review to identify suitable approximate computing techniques, followed by the selection of the most appropriate methods based on performance and characteristics. A detailed design of the MAC will be developed, incorporating the chosen approximate computing methods. The third objective is to implement the MAC unit using Hardware Description Language (Verilog), and to perform initial testing through FPGA simulations to verify functionality. Once the MAC unit has been tested in isolation it will be integrated into a RISC-V SoC, ensuring seamless compatibility and functionality within the broader system architecture. After integration, the final objective is to conduct thorough testing, comparing the performance, power consumption and accuracy of the new design. These tests will focus on evaluating how the approximations affect the system's overall performance. Each of these objectives will be simulated, tested, and compared against current industry standards to assess the success of the project in achieving its goals.

2. Theory and Literature Review

The exponential growth in computational demands, driven by applications in machine learning, multimedia processing, and big data analytics, has strained traditional digital design paradigms. Classical computing architectures prioritise precision and exactness, which come at the cost of increased power consumption, area usage, and latency. With the diminishing benefits of Moore's Law and the rising need for energy-efficient hardware, approximate computing has emerged as a transformative approach to hardware design.

Approximate computing operates on the principle that not all applications require perfect accuracy. Many domains, especially those involving human perception or probabilistic outcomes, can tolerate small errors without significant degradation in performance. By trade-off of accuracy, approximate computing reduces hardware complexity, resulting in substantial improvements in energy efficiency, and processing speed.

At the heart of this model shift are arithmetic units like adders and multipliers which constitute a significant portion of computational workloads in digital systems. Optimising these units for approximate computing forms the core of this paper's contributions.

2.1 Approximate Adders

Adders are a fundamental component in digital circuits, responsible for executing arithmetic operations that often dominate computational workload. Traditional adder designs prioritise accuracy, however, approximate adders introduce intentional inaccuracies to achieve resource savings. A proposed approximation approach using Lower-Part OR-based Approximate Adders [1] aligns with similar research, introducing the concept of approximate adders as a means to trade-off accuracy for reduced power consumptions and area in energy-efficient VLSI systems. Ramasamy et al. proposed a carry-based approximate full adder, demonstrating that bypassing the carry propagation chain in the least significant bits (LSB) can drastically improve speed and reduce area at the cost of negligible error [3].

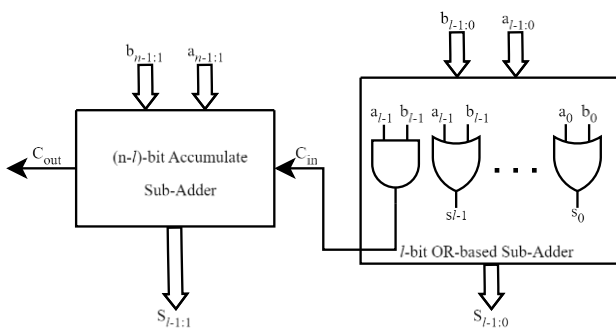


Fig 1 Lower-Part OR Adder [10]

2.1 Approximate Multipliers

Multiplication is a computationally intensive operation, making approximate multipliers a critical focus for energy-efficient design. Approximate multipliers reduce the complexity of partial product summation, which directly impacts delay and power consumption. Novel hardware design of approximate multipliers is provided, Lower-Part OR-based Approximate Multiplier [1], integrating the concept of Wallace Tree multipliers for accurate MSBs and OR-based logic for approximate LSBs. The combination of these techniques results in a novel multiplier design that balances accuracy, speed and resource utilisation, suitable for FPGA based implementations.

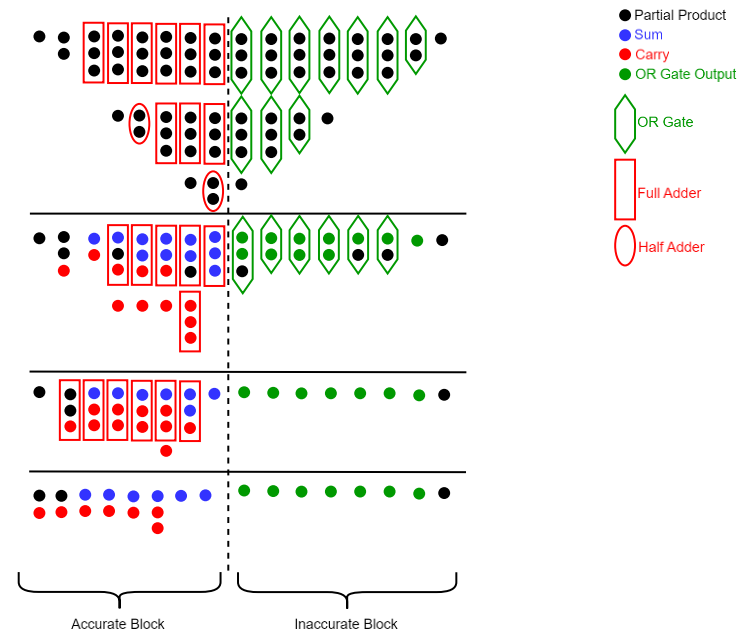


Fig 2 Lower-Part OR Wallace Tree Multiplier [11]

2.3 Approximate Matrix Multiplication

Matrix multiplication is a fundamental operation in numerous computational tasks, including AI, scientific computing, and graphics processing. Despite its importance, research into approximate matrix multipliers is limited. The proposed matrix multiplier design is a significant step forward, as it combines approximate multipliers and adders in a single hardware implementation [1], by targeting an FPGA platform and demonstrating scalability across different matrix sizes and bit-widths.

2.4 Compressor-Based Approximate Multipliers

Traditional Multiplier Architectures typically involve, partial product generation, partial product accumulation, and final addition. Partial product generation involves producing intermediate results by multiplying bits of input operands, followed by partial product accumulation, summing the intermediate results using adders or compressors, and lastly final

addition produces the output from accumulated partial products. A compressor is a combinational logic circuit used to sum multiple binary inputs and produce a small number of outputs, usually two, a sum and a carry. The most commonly used compressors are 3:2, 4:2 and 5:2 [2].

In traditional designs, compressors play a critical role in the accumulation phase. However, conventional exact compressors are power-intensive and complex, especially in FPGA-based implementations due to limited logic resources and/or cascading delays and increased power consumption from logic circuits.

2.4.1 Conventional 3:2 Compressors (Full Adder)

A 3:2 compressor is equivalent to a full adder. It takes three input bits and outputs two bits, a sum, the least significant bit of the result, and a carry, the most significant bit of the result.

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = (A \cdot B) + (B \cdot C_{in}) + (A \cdot C_{in})$$

This is the simplest compressor and serves as the building block for higher-order compressors.

2.4.2 Conventional 4:2 Compressor

A 4:2 compressor takes 4 inputs and produces two output bits and an additional carry-in and carry-out.

$$S = A \oplus B \oplus C \oplus D \oplus C_{in}$$

$$C = (A \cdot B) + (C \cdot D) + (C_{in} \cdot (A \oplus B \oplus C \oplus D))$$

This style of compressor is advantageous as it reduces four rows of partial products to two, with a carry propagated to next stage and minimises delay compared to a series of 3:2 compressors. Most commonly used in high-performance multipliers to speed up partial product reduction especially in Dadda multipliers.

2.4.3 Conventional 5:2 Compressor

The 5:2 compressor takes five input bits and produces two output bits, along with two carry bits (one from previous stage and one for the next stage).

$$S = A \oplus B \oplus C \oplus D \oplus E \oplus C_{in1} \oplus C_{in2}$$

$$C = \text{Majority function of inputs}$$

A 5:2 compressor is particularly efficient for reducing a large number of partial product rows in multipliers, for instance 16x16 or 32x32.

2.5 Approximate m:2 Compressor

Traditional compressors focus on exact computations, which are not always necessary for error-tolerant applications such as image processing or machine learning. The Approximate m:2 Compressor is

designed to aggregate multiple elements in two equal-weight outputs while minimising hardware complexity and power consumption.

The Approximate m:2 Compressor is designed into two output bits, Sum (S) and Carry (C) outputs represent the cumulative result of m elements with reduced precision. Probability analysis is used to determine the most partial product values are concentrated between 0 and 2, making it feasible to represent them with two outputs.

$$S = A + B + C + \dots + m =$$

$$C = (A \cdot (B + C + \dots + m)) + (B \cdot (C + \dots + m))$$

$$+ \dots$$

The use of OR-based logic gates reduces the number of LUTs required, compared to traditional compressors using XOR and AND gates furthermore reducing propagation delay. Fewer logic gates results in lower power consumption.

2.5.1 Once-Through Multiplier Architecture

The primary design objectives of CAM2 are to minimise power consumption, area utilisation and delay by using simple logic operations, such as OR-gates, instead of more complex compressors in specific stages [2]. The CAM2 is implemented in three stages:

- **Stage 1:** Initial compression of partial products using carry-lagged compressors.
- **Stage 2:** Approximate compression of remaining partial products using OR operations.
- **Stage 3:** Final summation using carry chains to product the final product.

2.5.2 Hardware Efficiency and metrics

CAM2 achieves a power reduction of 57.90% compared to exact multipliers. The use of OR operations significantly reduce the area, leading to a 33.80% reduction in LUT usage. By simplifying logic in Stage 2 and avoiding recursive carry propagation, CAM2 reduces delay by 24.78% [2], with a Mean Relative Error Distance (MRED) of 5.86% and an Error Rate (ER) of 84.50%. CAM2 sacrifices accuracy for greater efficiency, making it suitable for applications where minor inaccuracies are acceptable.

2.6 Fuzzy Memoization

Fuzzy memoization is an approximate computing technique based on instruction memoization where approximate results are cached and reused instead of recomputing exact results. Both Instruction and Fuzzy memoization store input and output data for a process as an entry in the memo table, and reuse it to reduce execution time by skipping the original process [4].

This is particularly useful in applications where slight inaccuracies in the output are acceptable, such as image processing, machine learning, and signal processing.

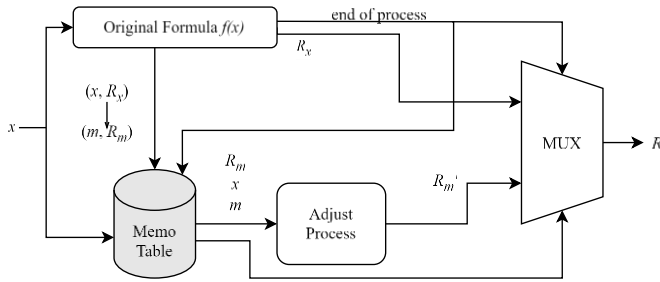


Fig 3a Proposed Memoization System [4]

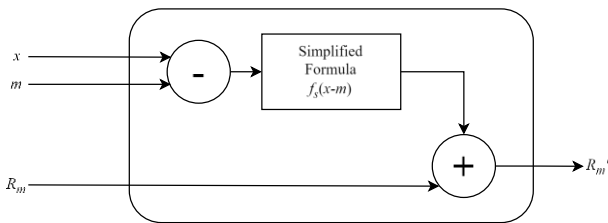


Fig 3b Adjust Process [4]

2.6.1 Cache Initialisaiton

A cache (or lookup) table is created to store the results of previous computations. Each entry in the cache consists of inputs and outputs. The original input value of vectors used for computation and the output is the result of those inputs. For instance, in a MAC operation:

- **Inputs:** A and B
- **Output:** $A * B + \text{previous result}$

2.6.2 Fuzzy Matching

Search the cache for a stored input that is “similar enough” to the new input, based on a predefined similarity threshold. If a similar input is found (cache hit), reuse the cached result instead of performing the computation. If no similar input is found (cache miss), compute the result, store it in the cache, and use it for future queries [4].

2.6.3 Similarity Metrics

Several metrics can be used to determine the similarity between inputs, depending on the application. Hamming distance counts the number of bit positions in which the inputs differ, Euclidean distance measures the geometric distance between two vectors in a multi-dimensional space. Manhattan distance measures the sum of absolute differences between corresponding elements of two vectors and custom thresholding is a user-defined threshold that dictates the maximum allowable threshold difference between inputs.

2.6.4 Updating the Cache

When a cache miss occurs, the cache is updated with the new input-output pair. Depending on the cache size and replacement policy, older or less relevant entries may be removed. Common cache replacement policies include:

- **Least Recently Used (LRU):** Removes the least recently accessed entry.
- **First-In-First-Out (FIFO):** Removes the oldest entry.
- **Random Replacement:** Randomly selects an entry to replace.

2.7 Proposed Novel Approximate Arithmetic Unit Design

The proposed Approximate Arithmetic Unit Design (AAUD) is a hybrid architecture that combines multiple approximation techniques to achieve significant improvements in power consumption, area, delay, tailored for a MAC unit. The AAUD integrates OR-based approximate adders, approximate multipliers, compressor-based approximations, and fuzzy memoization. This architecture is designed for error-tolerant applications like fuzzy-filter-based FIR filters used in image processing where small inaccuracies in computations are acceptable.

2.7.1 Architecture

The architecture of the AAUD is split into 4 stages. Approximate Partial Product Generation, Partial Product Reduction with Compressor-Based Approximations, Approximate Adders for Accumulation and Fuzzy Memoization for reuse of results. The detailed AAUD-MAC flow is provided below:

1. **Input Fetch:** Fetch inputs A and B for the MAC operation.
2. **Fuzzy Memoization Lookup:** Search the cache for a similar input pair using Hamming distance.
3. **Cache Hit:** If a similar input is found, reuse the cached result. If not, proceed to the next step.
4. **Approximate Multiplication:** Compute the product using the Lower-Part OR-based Approximate Multiplier
5. **Partial Product Reduction:** Use Approximate m:2 Compressors to reduce the partial products.
6. **Approximate Addition:** Accumulate the reduced partial products using the Lower-Part OR-based Approximate Adder
7. **Cache Update:** Store the result in the cache for future reuse.

The proposed AAUD-MAC is ideal for a Fuzzy Memoized FIR Filter used in image denoising noisy image pixels. Each pixel is passed through the FIR filter, where the MAC operations are accelerated using AAUD-MAC. Common pixel patterns are cached and reused, reducing redundant computations. The AAUD combines various approximation technique to deliver a highly efficient MAC unit suitable for energy-constrained application. Its hybrid design leverages approximate arithmetic and fuzzy memoization, achieving significantly improvements in power, area, and delay making it a viable solution for real-time, error-tolerant applications such as image processing.

3. Technical Progress to Date

Initial technical progress to date includes laying a foundation for the AAUD framework architecture. The following section delineates the work completed as yet.

Note: *Coding and testing have been implemented on Xilinx using Verilog on a Nexys-A7 FPGA development board*

3.1 Architecture Design for AAUD

The initial step involved in designing the AAUD architecture consists of developing an initial outline to visualise the framework, structure and functionality. The models below directed focus of subsequent stages of the project by specifying the design's requirements and necessary modules needed to construct the unit.

The diagrams below illustrate the implementations of approximate units that are to be integrated into the multiplier and adder units.

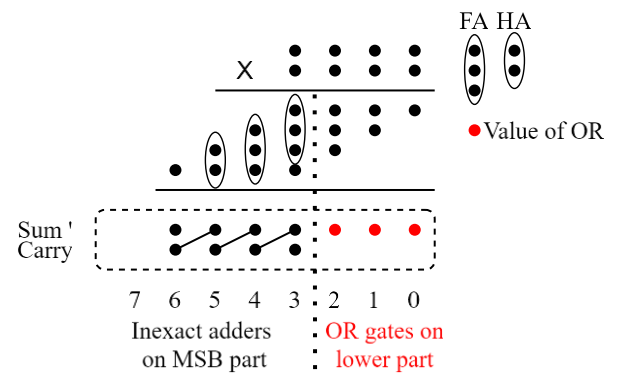


Fig 4 Approximate Multiplication Block

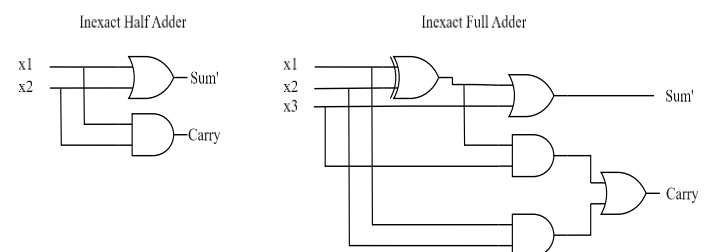


Fig 5 Approximate Half and Full Adders

3.2 Testing and Framework Design

The first stage was to design and code a pipelined Multiply-Accumulate (MAC) unit which will be a tool for benchmarking, to evaluate the feasibility and performance metrics such as speed, power consumption, and chip area of the AAUD design

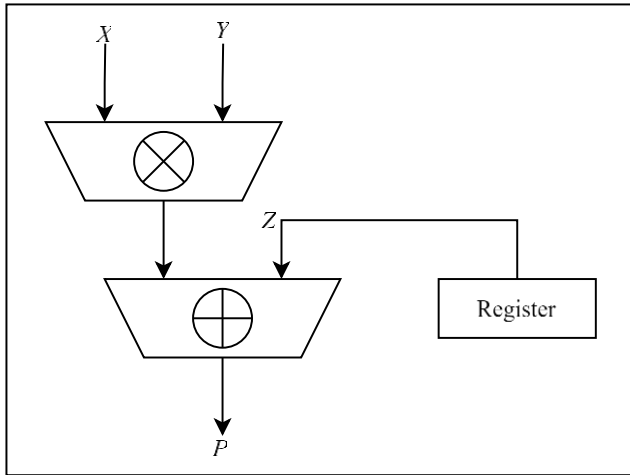


Fig 6 Traditional MAC Unit Design

Pipelined MAC Unit Implementation

Last year, significant modifications were necessary to create a fully functional pipelined MAC module. By utilizing online resources, a MAC module was developed, accompanied by a testbench to verify its operation and performance.

The testbench included tests to assess throughput by measuring the processing speed of 1000 inputs, accuracy to ensure the correctness of each operation, and Vivado analytics to estimate the chip area and the hardware resources used. Power consumption was also measured to evaluate the design's energy efficiency.

3.3 AAUD Architecture Development and Testing

Developing further from the initial MAC design, significant contributions to the design and implementation of the AAUD have been made. The design can be summarised into 3 main stages.

1. **Approximate Multipliers** where a design for Partial Product generation, approximate 3:2, 4:2 and 5:2 compressors, Dadda multipliers for MSB half, OR-bit logic for LSB half, and approximate adders is being evaluated.
2. **Approximate Adders** where approximate adder techniques such as Carry Propagated Addition (CPA), Carry Lagged Addition (CLA), and other Carry Chains are being considered as design choice.
3. **Fuzzy Memoization** is being considered as an optional implementation for RISC-V architecture. Further research is required; however, useful papers have been acquired that delineate the implementation process. A suitable use for this is being considered, however, possibilities in the image processing area are promising.

For the first two sections initial design and coding development is being progressed, however, due to it being in very early stages it will not be included in the report as simulation testbenches are still in progress.

3.4 Milestones

As by the submission date of this report the following milestones have been achieved:

1. Basic framework has been laid by performing benchmarking of a pipelined MAC unit.
2. Conceptual design of AAUD architecture has been implemented.
3. Initial implementation of AAUD has begun, in very early stages.

These achieved milestones lay a promising structure for the subsequent phase in the project. The following milestones and progression can be found in the subsequent section or in the Gantt Chart in the respective appendix.

4. Project Management

The primary objective of this project is to design and implement an Arithmetic Unit using Approximate Computing techniques, specifically a 32-bit Multiply-Accumulate (MAC) unit, for integration into a RISC-V System on Chip (SoC). This will involve utilizing the Artix Nexus 7 FPGA and the Xilinx Vivado Design Suite for design, simulation, and testing. The planned work is organized into several phases, each corresponding to specific tasks and milestones, which will be carried out according to the updated project timeline outlined in the accompanying Gantt chart, Figure 1 and Figure 2.

Following the literature review, the next phase has been the focus on the design of the 32-bit MAC unit. A detailed block diagram has been developed, considering the selected Approximate Computing techniques and the target performance metrics, including power consumption, speed, and accuracy. The expected performance of the design will be calculated, and peer reviews will be conducted to refine the design. Once the design is finalized, the implementation will proceed using Hardware Description Languages (Verilog/VHDL), followed by initial debugging and testing. This phase will mark the completion of **Milestone 2**.

In the subsequent phase, simulation and verification will take place. Test benches will be developed using the Xilinx Vivado Design Suite to conduct a series of simulations. These simulations will evaluate the functionality and performance of the MAC unit, including power consumption and accuracy, under the conditions specified by the design. The design will undergo further debugging and refinement based on simulation results, with the final performance metrics documented. This will lead to the completion of **Milestone 3**.

The next major phase will involve integrating the MAC unit into the RISC-V architecture. This phase will include participation in relevant labs to better understand the implementation strategies for embedding the MAC unit into the RISC-V processor. Once integrated, the system will undergo debugging and initial testing to verify that the MAC unit functions correctly within the RISC-V SoC framework. Successful integration and testing will result in the completion of **Milestone 4**.

Finally, a thorough evaluation and analysis phase will be conducted to assess the final design. This will involve creating a series of test cases to evaluate power consumption, processing speed, and accuracy, as well as conducting error testing to evaluate the impact of Approximate Computing on the system's

output. The results will be compared to existing models to assess the advantages and trade-offs of using Approximate Computing techniques.

Throughout the entire project, comprehensive documentation will be maintained, detailing the design process, test results, and performance evaluations. This will culminate in the preparation of a final report, which will reflect the design and implementation process, key findings, and recommendations for future work in Approximate Computing.

A detailed Gantt chart outlining the project timeline, milestones, and progress will be included as an appendix to track the project's completion. The chart will serve as a tool to monitor progress and make necessary adjustments against the initial plan to ensure that the project remains on schedule.

References:

- [1] A. Gupta and K. Suneja, "Hardware Design of Approximate Matrix Multiplier based on FPGA in Verilog," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2020, pp. 496-498, doi: 10.1109/ICICCS48265.2020.9121004. keywords: {Performance evaluation;Computational modeling;Approximate computing;Hardware;Software;Table lookup;Hardware design languages;approximate computing;matrix multiplier;Verilog;Field Programmable Array (FPGA);Look Up Tables (LUTs)},
- [2] Y. Guo, X. Chen, Q. Zhou and H. Sun, "Power-Efficient and Small-Area Approximate Multiplier Design with FPGA-Based Compressors," 2024 IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, Singapore, 2024, pp. 1-5, doi: 10.1109/ISCAS58744.2024.10558590. keywords: {Image coding;Accuracy;Power demand;System performance;Circuits;Approximate computing;Hardware;Approximate computing;FPGA-based compressor;Low-power circuit},
- [3] T. Nomani and M. Mohsin, "A Novel Approximate Adder Design Methodology with Single LUT Delay for Fault-tolerant FPGA-based Systems," 2019 Second International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT), Karachi, Pakistan, 2019, pp. 1-6, doi: 10.1109/INTELLECT47034.2019.8955460. keywords: {Approximate computing;Adders;FPGA;Image processing applications;Fault-tolerant systems},
- [4] Y. Ono and K. Usami, "Approximate Computing Technique Using Memoization and Simplified Multiplication," 2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), JeJu, Korea (South), 2019, pp. 1-4, doi: 10.1109/ITC-CSCC.2019.8793369. keywords: {Energy consumption;Frequency modulation;Approximate computing;Field programmable gate arrays;Error analysis;Gray-scale;Approximate Computing;Image Processing;Field Programmable Gate Array (FPGA)},
- [5] S. Ullah, S. S. Murthy and A. Kumar, "SMAproxLib: Library of FPGA-based Approximate Multipliers," 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2018, pp. 1-6, doi: 10.1109/DAC.2018.8465845. keywords: {Table lookup;Field programmable gate arrays;Libraries;Adders;Performance gain;Delays;Viterbi algorithm;Approximate Computing;Multipliers;Adders;FPGAs;Optimization;Area;Latency;Design Space Exploration},
- [6] H. Nakahara and T. Sasao, "A deep convolutional neural network based on nested residue number system," 2015 25th International Conference on Field Programmable Logic and Applications (FPL), London, UK, 2015, pp. 1-6, doi: 10.1109/FPL.2015.7293933. keywords: {Table lookup;Field programmable gate arrays;Convolution;Kernel;Neural networks;Clocks;Dynamic range},
- [7] W. Liu, F. Lombardi and M. Schulte, "Approximate Computing: From Circuits to Applications [Scanning the Issue]," in *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2103-2107, Dec. 2020, doi: 10.1109/JPROC.2020.3033361. keywords: {Special issues and sections;Approximate computing;Computer architecture;Artificial intelligence;Approximation algorithms},
- [8] A. M. Dalloo, A. Jaleel Humaidi, A. K. Al Mhdawi and H. Al-Raweshidy, "Approximate Computing: Concepts, Architectures, Challenges, Applications, and Future Directions," in *IEEE Access*, vol. 12, pp. 146022-146088, 2024, doi: 10.1109/ACCESS.2024.3467375. keywords: {Approximate computing;Accuracy;Transistors;Surveys;Silicon;Power demand;Machine learning;Deep learning;Statistical analysis;Neuromorphic engineering;Approximate computing;approximate programming language;approximate memory;circuit-level;approximate machine learning;deep learning;approximate logic synthesis;statistical and neuromorphic computing;cross layer and end-to-end approximate computing},
- [9] A. Gorantla, R. Kothapalli and T. L. Spandana, "Developments of Approximate Computing: From Algorithm Level to System Level," 2022 International Conference on Computing, Communication and Power Technology (IC3P), Visakhapatnam, India, 2022, pp. 52-56, doi: 10.1109/IC3P52835.2022.00020. keywords: {Measurement;Power demand;Costs;Approximate computing;Very large scale integration;Approximation algorithms;Delays;approximate computing;low power;very large scale integration},
- [10] J. Han, "Approximate Computing with Approximate Circuits: Methodologies and Applications ESWEEK 2017 Tutorial." Accessed: Dec. 10, 2024. [Online]. Available: https://www.ece.ualberta.ca/~jhan8/publications/esweek17_tutorial_ac_part2.pdf
- [11] P. Yadav, A. Pandey, M. R. K., R. P. K.J., V. M.H. and N. K. Y.B., "Low Power Approximate Multipliers With Truncated Carry Propagation for LSBs," 2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS), Windsor, ON, Canada, 2018, pp. 500-503, doi: 10.1109/MWSCAS.2018.8624067. keywords: {Adders;Logic gates;Delays;Transistors;Power dissipation;Power demand;CMOS technology;Approximate Multiplier;Carry generation;Accuracy;PDP},

Appendix A

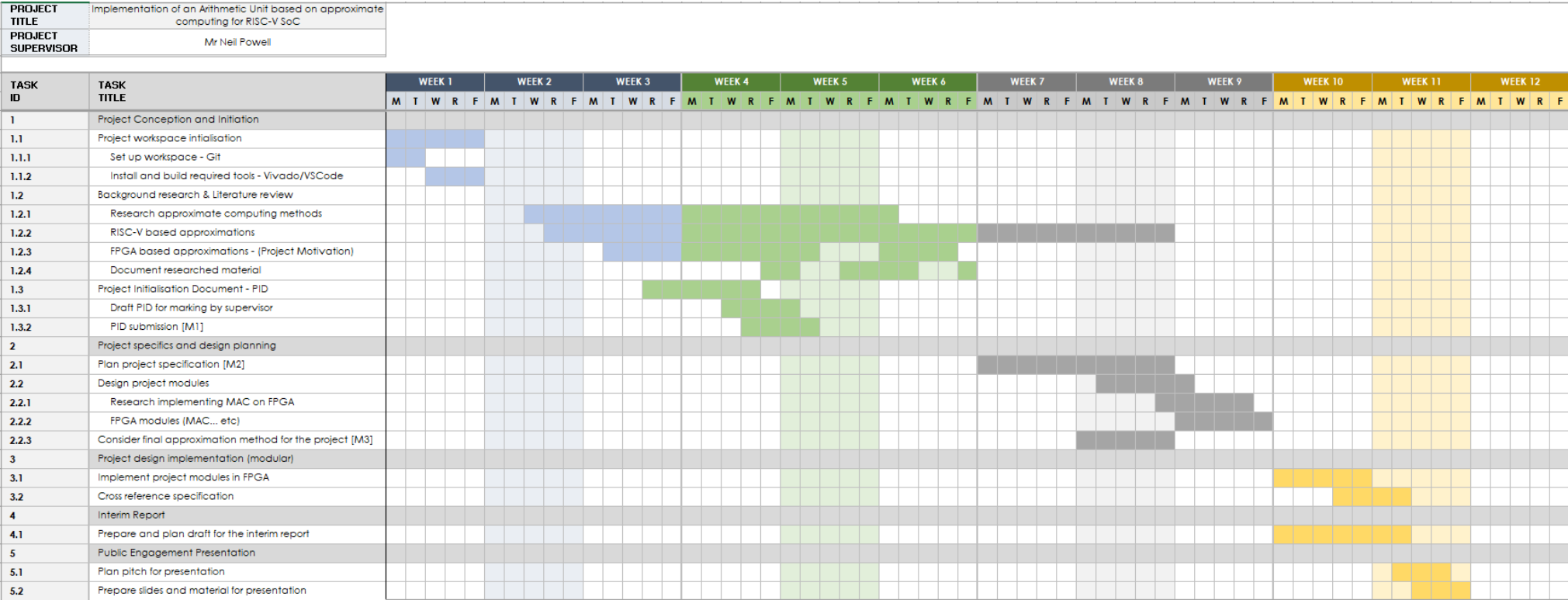


Fig 7a: Gantt Chart for Semester 1

EEE380/381/480 Interim Report



Fig 7b: Gantt Chart for Semester 2

- M1 – Milestone 1 for producing a final process initiation document for submission
- M2 – Milestone 2 for planning a detailed project specification
- M3 – Milestone 3 for choosing an approximation method to be implemented for the project
- M4 – Milestone 4 for implementing the chosen approximation method in FPGA

Appendix B

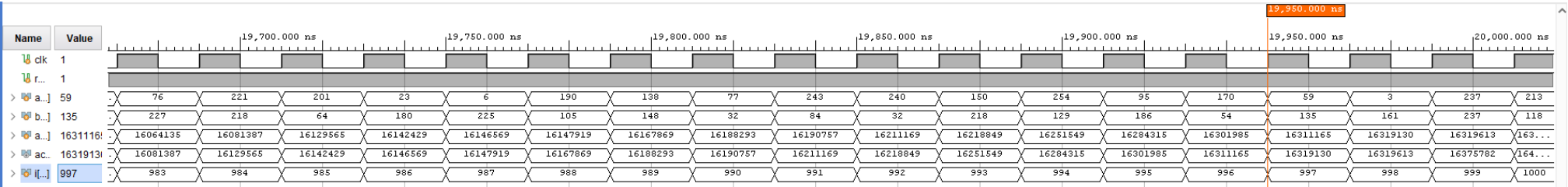


Fig 8: Waveform for Pipelined MAC 1000 inputs