**Team04 Final Report**

R13943009 鄭至盛、R13943021 陳品睿

**Part1. Introduction**

We will focus on optimizing the workload defined in Lab 6 by leveraging specialized hardware and techniques to significantly enhance performance. The workload involves performing three key computations: Finite Impulse Response (FIR) filtering, quick sort, and matrix multiplication. To achieve this, we will integrate specialized Intellectual Property (IP) cores designed to handle each of these tasks efficiently. These IP cores will be used to accelerate the computations and their associated datapath operations, ensuring a faster and more optimized execution of the workload.

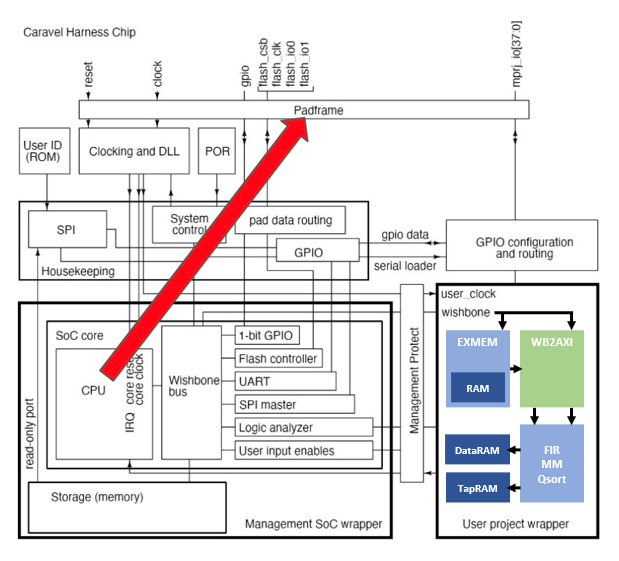
Our optimization strategy will incorporate concepts such as memory hierarchy design, which ensures data is stored and accessed efficiently to minimize latency and maximize throughput. We will also implement direct memory access (DMA) to facilitate fast data transfers between the memory and the IP cores, bypassing the CPU and reducing overhead. Additionally, firmware co-design will play a crucial role in orchestrating the interactions between hardware and software, enabling seamless integration and operation.

To accelerate the entire process, we will apply various hardware techniques and principles learned in previous labs (Lab 1 through Lab 4). These include the use of parallelism, pipelining, and other performance-enhancing methods, which are essential for achieving our optimization goals.

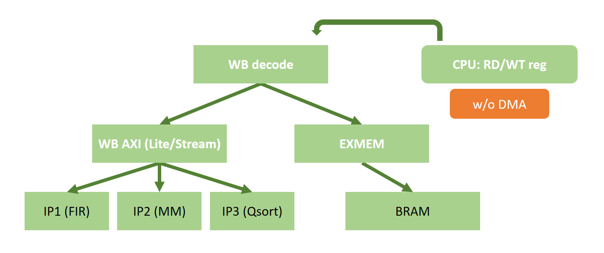
At the same time, we will incorporate communication functions into our design to ensure robust interaction between different components. This includes enabling UART transmission for data exchange and implementing an interrupt request (IRQ) mechanism to handle asynchronous events efficiently. By combining these techniques, our aim is to deliver a well-optimized and high-performance system capable of meeting the requirements of the defined workload.

**Part2. Architectures**

In our design, we mainly focus on the user project part of the caravel SoC, which is shown in the following figure:

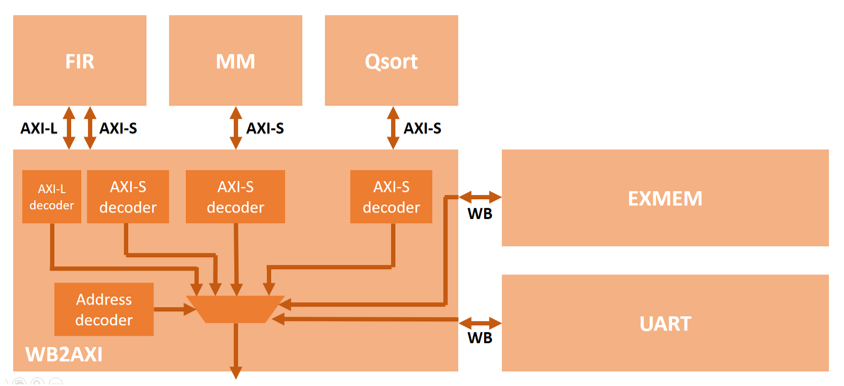


Our user project design consists of several key components, each tailored to meet the system's performance and functionality requirements. These components include a data transfer module responsible for efficient data movement, an execution module that manages memory operations, a UART communication module for facilitating serial communication, and three computation IP cores designed for specialized tasks. Together, these modules form a cohesive system that enables high-performance computation and seamless data management.



To evaluate the efficiency and effectiveness of our design, we have chosen computation cycles as our primary performance metric. This metric allows us to quantify the time required to execute computations and provides a clear indicator of the system's overall efficiency.

Besides the modules mentioned above, the basic design of bus protocol & controlling unit between wishbone and axi-interface is shown in the following figure:

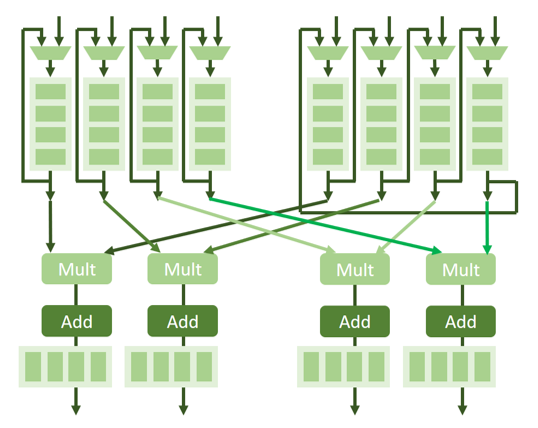


In the subsequent chapters, we will delve into the detailed designs of the various components that make up our system. This includes an in-depth exploration of the three computation IPs, each tailored to specific tasks and optimized for performance. Additionally, we will discuss the design of the SDRAM controller, a critical component responsible for managing memory access and ensuring data is retrieved and stored efficiently. Finally, we will focus on the direct memory access (DMA) module, which plays a pivotal role in facilitating high-speed data transfers between the memory and other components, bypassing the CPU to reduce overhead and improve throughput. By addressing these aspects in detail, we aim to provide a comprehensive understanding of the design choices and strategies implemented in our user project.

**Part3. Computations**

**Matrix multiplication unit**

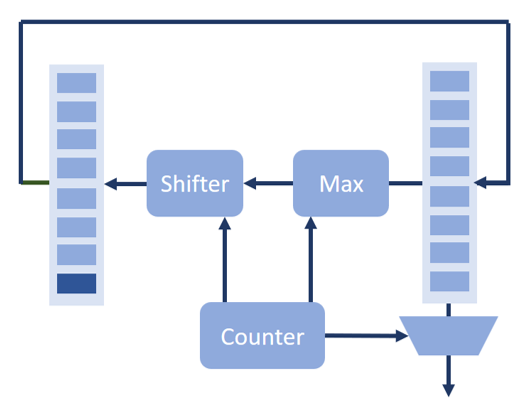
The design of our matrix multiplication unit is shown in the following figure:



It is comprised of an AXI-stream interface for data I/O, 3 of 4x4x32bit register files for matrices, 4 multipliers & adders for computation, and respective control logic for matrix multiplication. Our design requires at least 48 cycles for data transfer & 16 cycles for computation.

**Sorting unit**

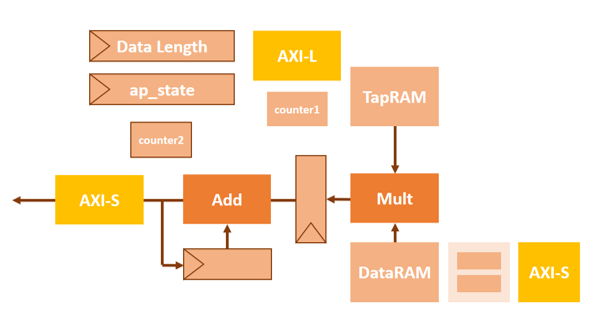
The design of our sorting unit is shown in the following figure:



It is comprised of an AXI-stream interface for data I/O, 1 of 10x32bit register files for matrices, an 10-elements comparator & masked shifter, and respective control logic for bubble sort. Our design requires at least 10 cycles for data transfer & 10 cycles for sorting.

**FIR unit**

The design of our FIR unit is shown in the following figure, mainly follows Lab4:



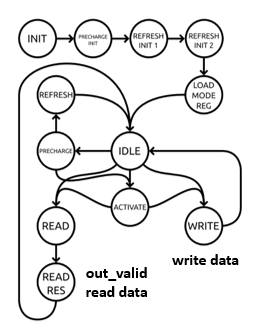
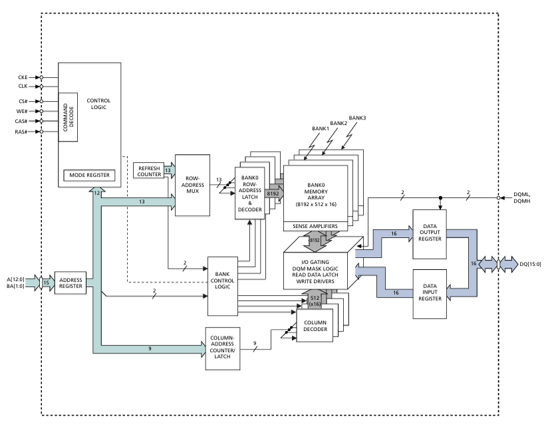
It is comprised of an AXI-stream interface for data I/O & AXI-lite interface for configuration, 2 of 11-elements BRAMs for tap & data, a multiplier & adder for pipelined FIR filter computation, and respective control logic for FIR filter. Our design requires at least 12 cycles for data transfer & computation with the decoupled data RAM controller design.

With the design mentioned above, the overall performance is shown in the table:

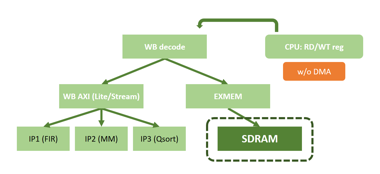
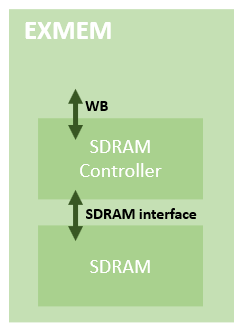
|  |  |  |  |
| --- | --- | --- | --- |
|  | **FIR** | **MM** | **Sort** |
| **Cycles** | 39712 | 21055 | 7043 |
| **Improvements** | - | - | - |

**Part4. SDRAM**

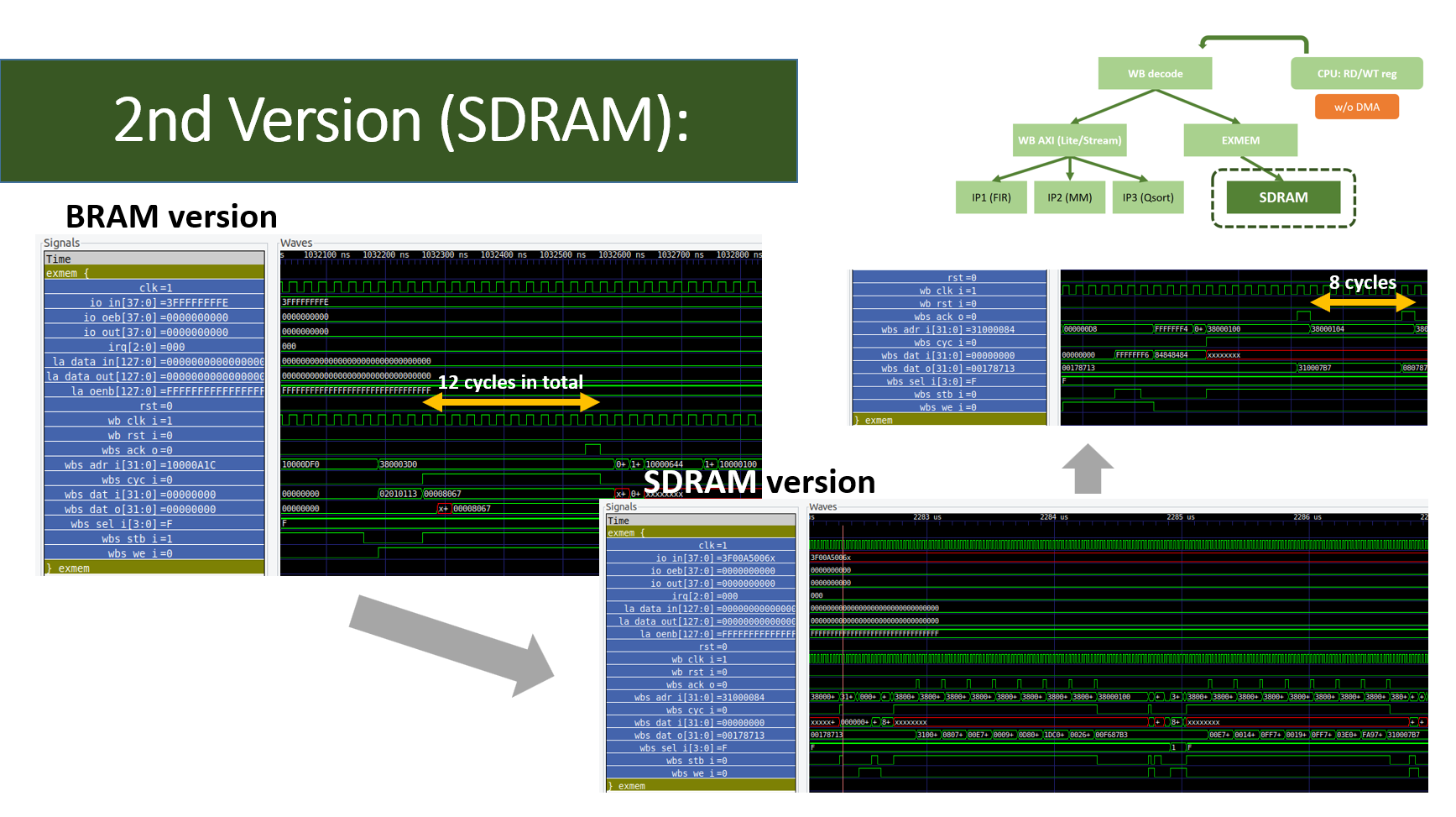
The basic design of caravel SoC includes a 10T BRAM in exmem module. To reduce the average memory access time, we decide to follow the example mojo design in Lab SDRAM with the behavior model of Micron MT48LC64M4A2 modules with 4x4 banks of 16MB RAMs.



With the introduction of SDRAM, the new design is shown in the following figure:

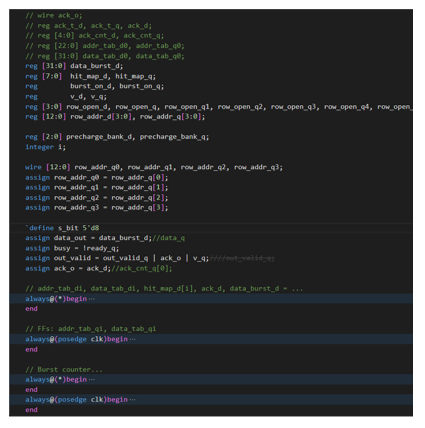
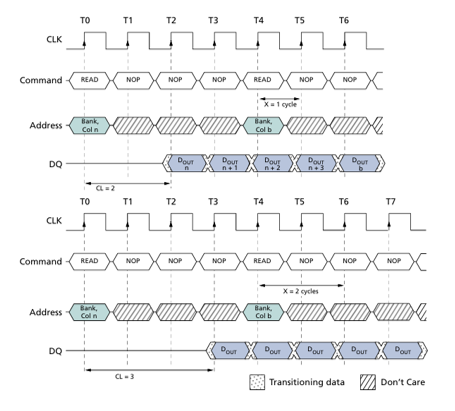
 

With the basic 7T SDRAM module, the overall performance can be improved by about 20%, the waveform & the performance is shown:

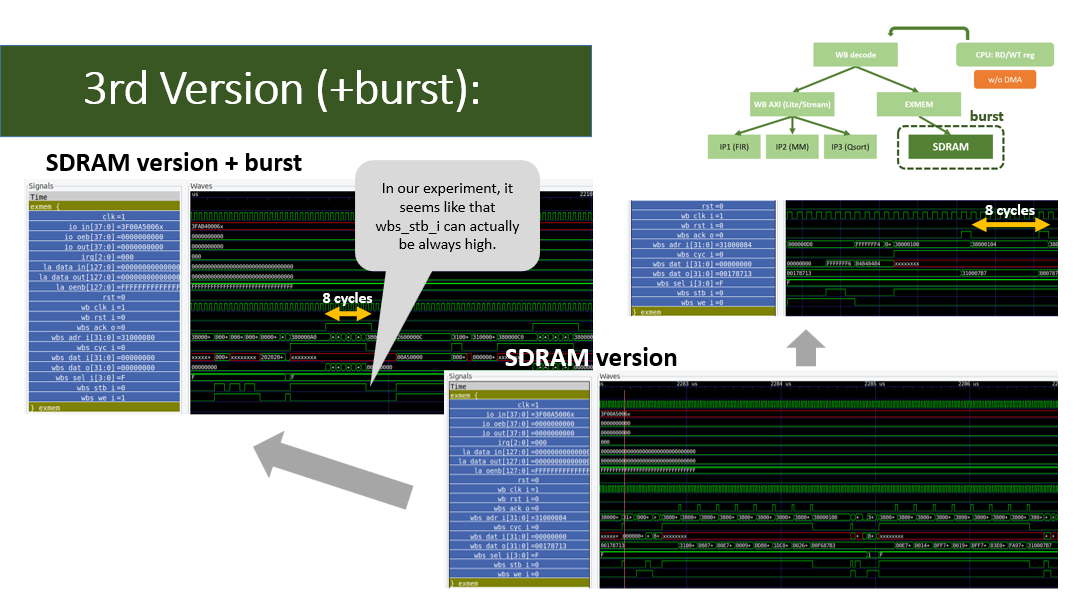


|  |  |  |  |
| --- | --- | --- | --- |
|  | **FIR** | **MM** | **Sort** |
| **Cycles (Basic)** | 39712 | 21055 | 7043 |
| **Cycles (SDRAM)** | 31033 | 16496 | 5638 |
| **Improvements** | 21.8% | 21.6% | 19.9% |

Additionally, the SDRAM can support faster access with 8 data in a single burst. The corresponding behavior model modification & design spec is shown:

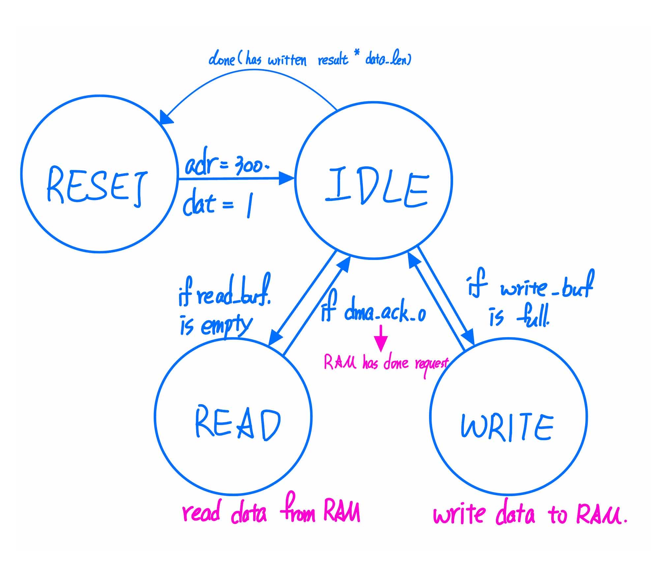
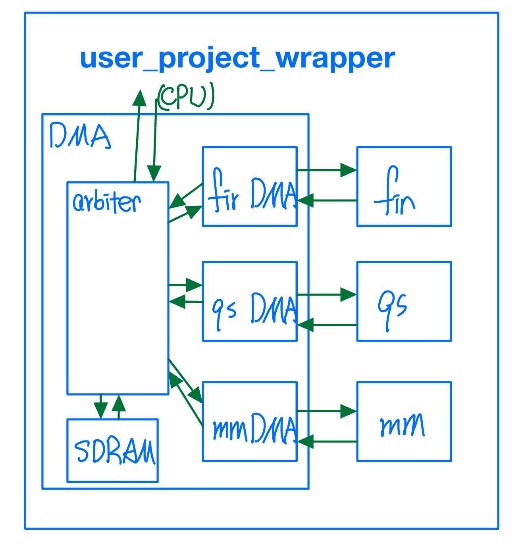
In our design, we didn’t provide a prefetch buffer design for CPU instruction since the wb\_stb\_i can be active between consecutive instructions, which might be modified as the future work of caravel SoC.



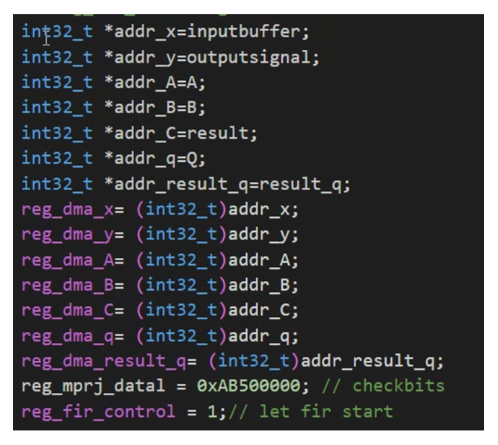
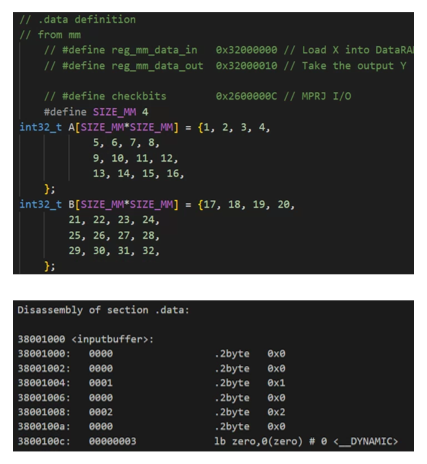
|  |  |  |  |
| --- | --- | --- | --- |
|  | **FIR** | **MM** | **Sort** |
| **Cycles (SDRAM)** | 31033 | 16496 | 5638 |
| **Cycles (w/ burst)** | 16804 | 8977 | 3363 |
| **Improvements** | 45.8% | 45.6% | 40.4% |

**Part5. DMA**

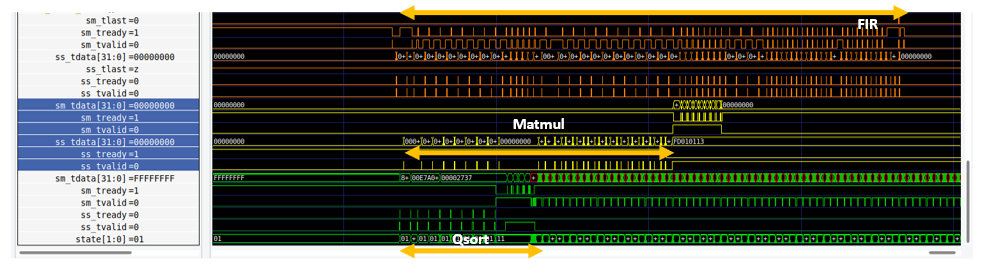
After introducing the design of SDRAM, the bottleneck of the total time is the data transfer time of writing one-by-one with firmware. Hence, we implement the DMA design with the following architecture:

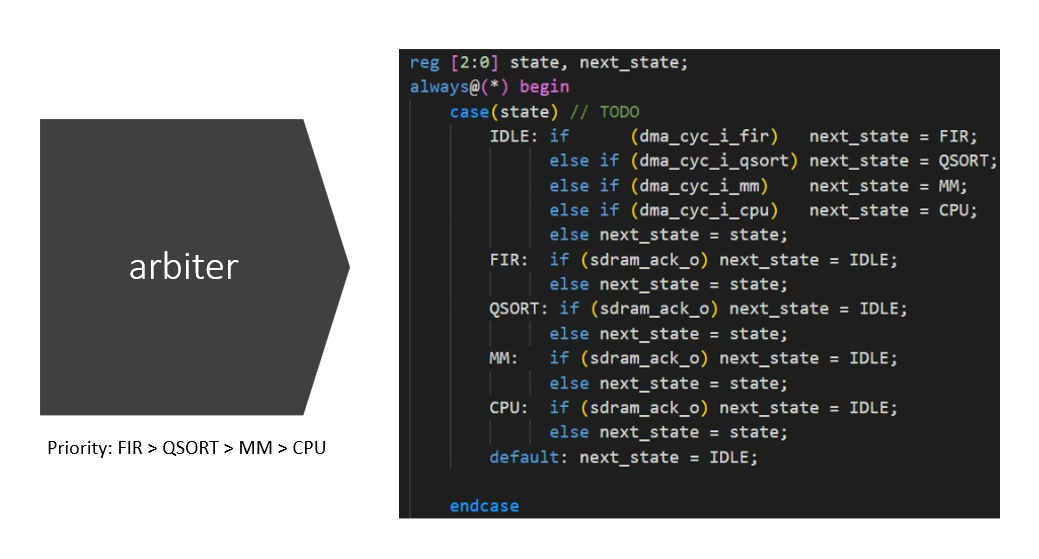


With DMA, the modified firmware includes array initialization & address assignment:



To further enhance the efficiency, the arbiter is also introduced to enable the 3 IPs to compute simultaneously, the design provides a simple logic to uniformly provide data to different modules.





|  |  |  |  |
| --- | --- | --- | --- |
|  | **FIR** | **MM** | **Sort** |
| **Cycles (w/ burst)** | 16804 | 8977 | 3363 |
| **Cycles (DMA)** | **3984 (in total)** | | |

**Part6. Conclusion**

In the design process, we’ve learned many techniques in SoC design from this project, covering: memory controller, firmware, and architecture design, not to mention further topics such as memory hierarchy, interleaving design, coherency, out-of-order prefetching, etc.

In conclusion, we successfully optimized the workload by dividing the process into two distinct parts, focusing on computation as the core aspect of our efforts. By accelerating the FIR filtering, sorting, and matrix multiplication operations through hardware implementations, we achieved significant performance improvements compared to the software-based approach. The timing metrics, which began when any of the three operations started and ended upon their completion, showcased the efficiency of our hardware-accelerated design.

By applying hardware techniques such as DMA & SDRAM, we effectively reduced the computational time and enhanced the overall system performance. This project demonstrates the advantages of transitioning from software to hardware optimization in achieving faster execution and more efficient workload handling and showcases the importance of the co-design of software & hardware in SoC design nowadays. The results highlight the importance of leveraging hardware acceleration to address complex computational tasks, paving the way for further enhancements and optimizations in future projects.