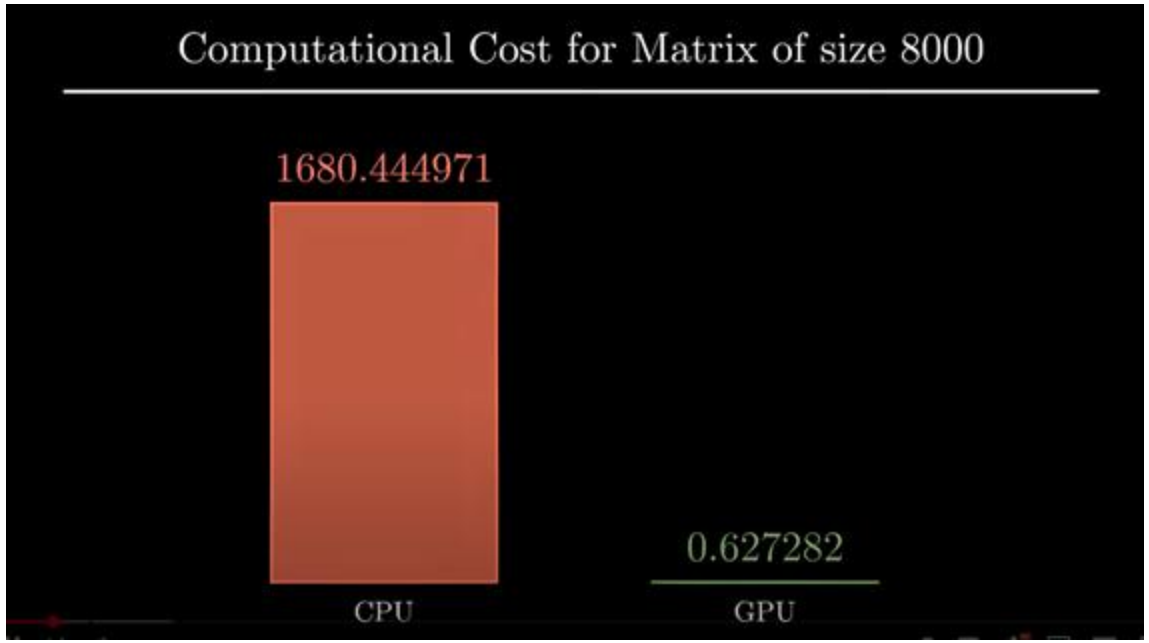
ESE 4730/5730, Chips-design

Computing: Digital 8-bit 6x6 matrix multiplier engine

Team 8: Shun Yao, Yue Zhang

Importance of Matrix Multiplier Engine

- Matrix multiplication is a fundamental computation in various fields, including **artificial intelligence**, **machine learning**, **image processing**, and **cryptography**.
- By implementing the matrix multiplication engine at the hardware level, we can optimize key performance factors such as power consumption, processing speed, and silicon area efficiency.



Project Goals

Efficient Matrix Computation: Develop a CMOS-based 8-bit 6×6 matrix multiplier engine, balancing speed, power efficiency, and silicon area.

Optimized Data Handling: Use SRAM-Array for efficient read/write, parallel dot product computation, and SIPO shift registers for seamless data flow.

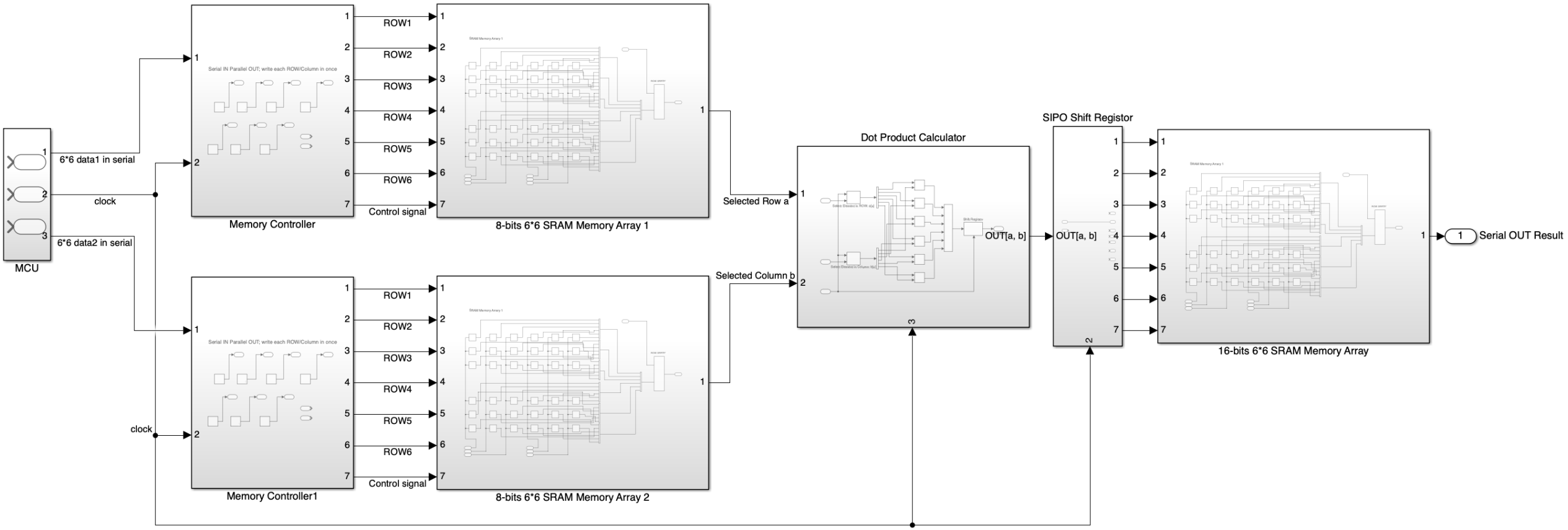
Low-Latency Processing: Minimize compute time with structured memory control and high-speed arithmetic.

Scalability: Ensure adaptability for integration into larger digital systems.

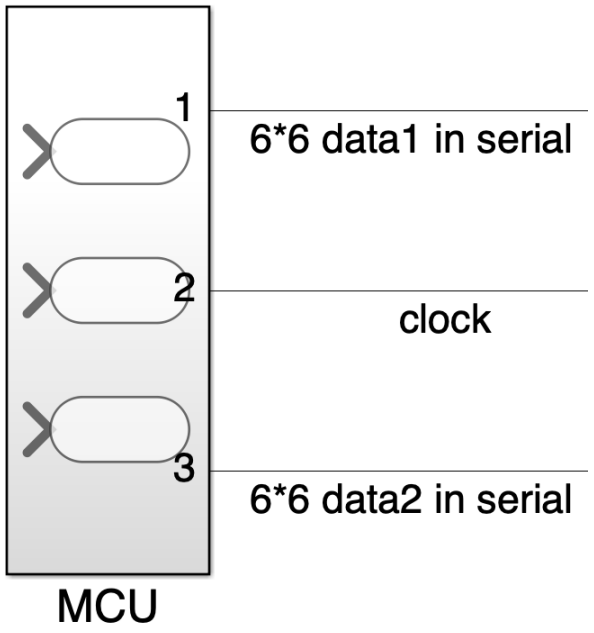
Verification & Simulation: Perform functional verification and power analysis; Implement layout design with DRC, LVS and LEX and hardware validation in 25FALL.

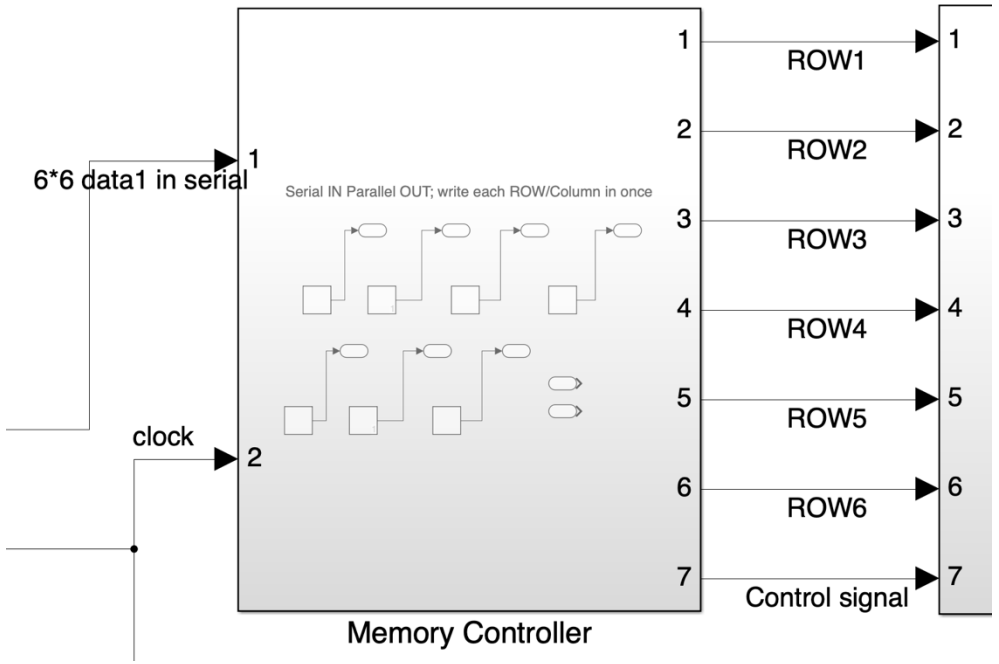
Top Cell Block Diagram

Block	VCC	Speed	Power Consumption
Memory	1.8V	15ns	2mW
Memory Controller		15ns	2mW
Register		4ns	0.1mW
Dot-Product Claculator		720ns	18mW



Detailed Block Diagram Explanation





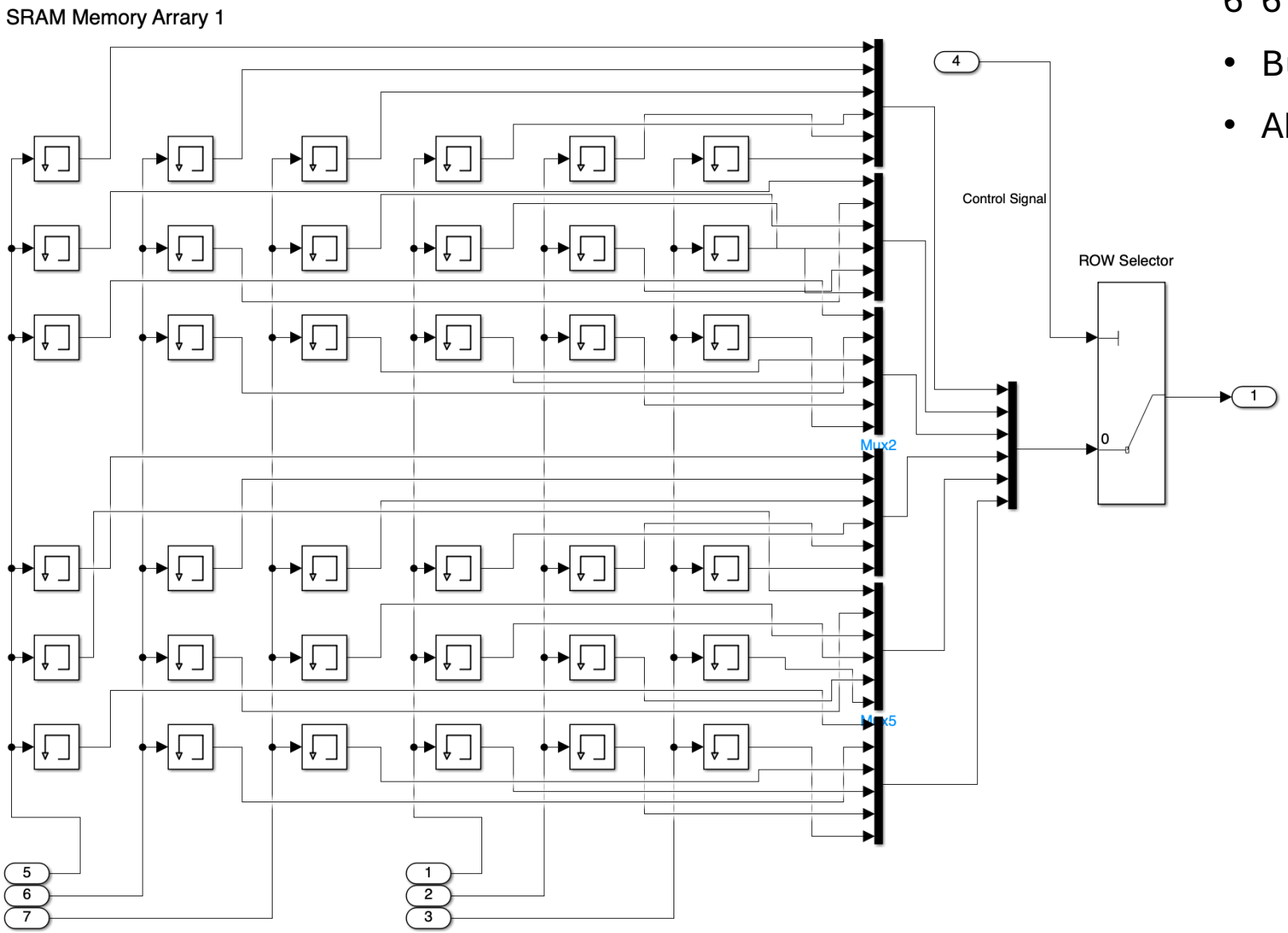
MCU (ESP32, VDD=3.3V)

- Sending 36 8-bit data in serial to the two memory controllers using SPI protocol
- Generate a Clock signal for the chip (150MHZ)

Memory Controller

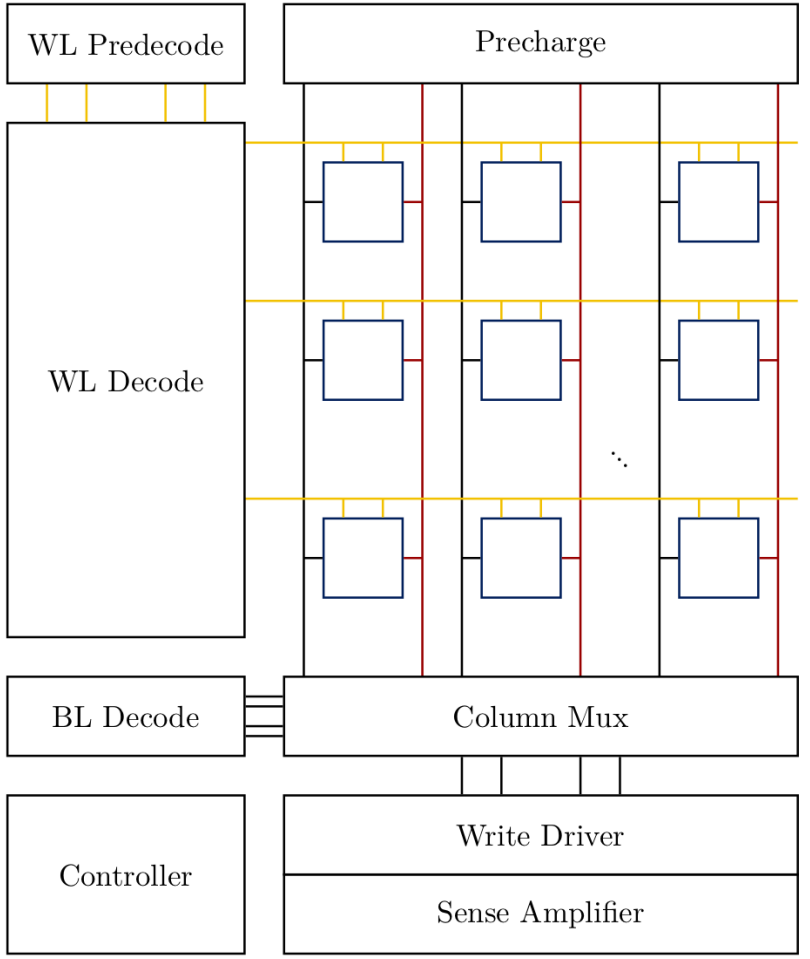
- Use 6 SIPO shift register to divide data steam from MCU into 6 arrays, write the Memory cell ROW by ROW 6 times.
- Generate a Control signal to select ROW/Column from memory array.

Memory Array Block Diagram Explanation



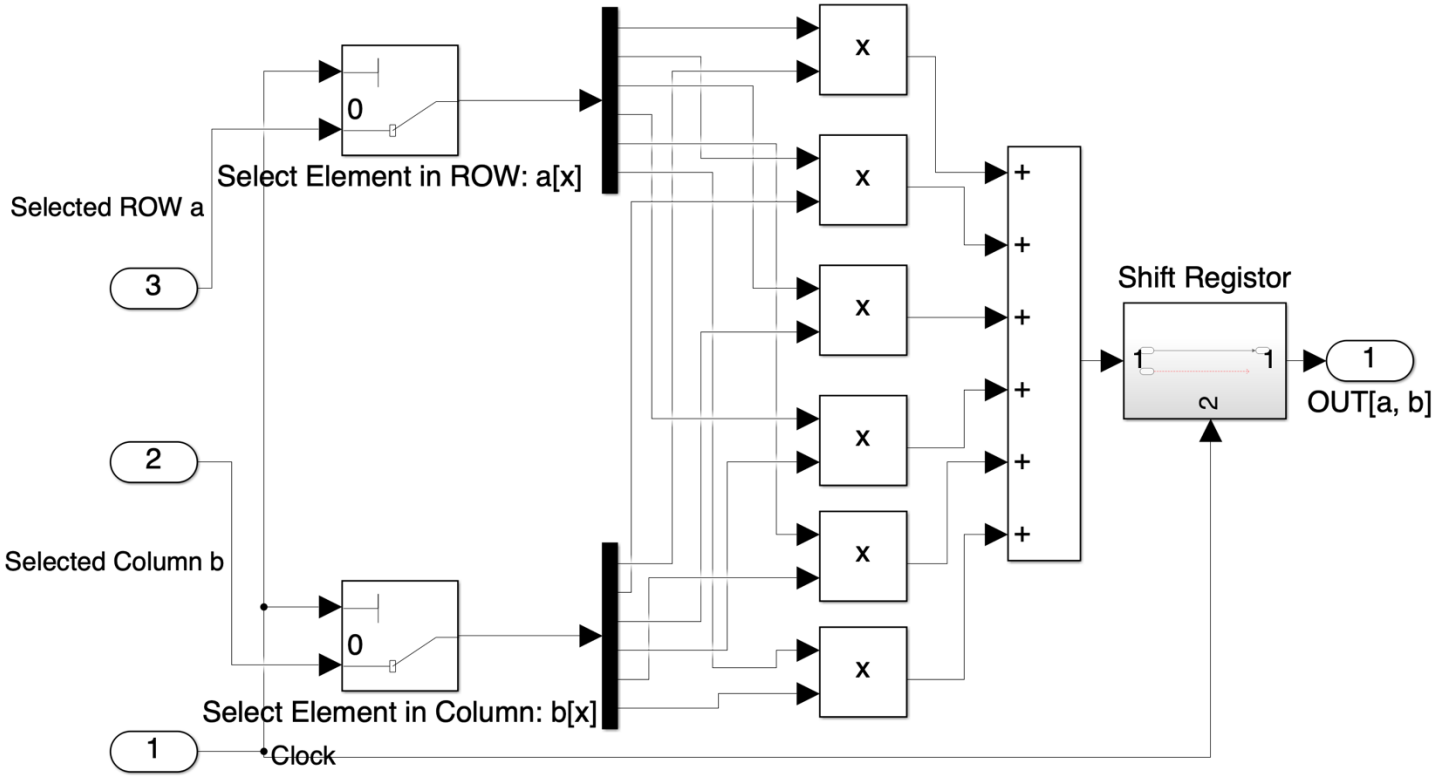
6*6 SRAM Memory Array

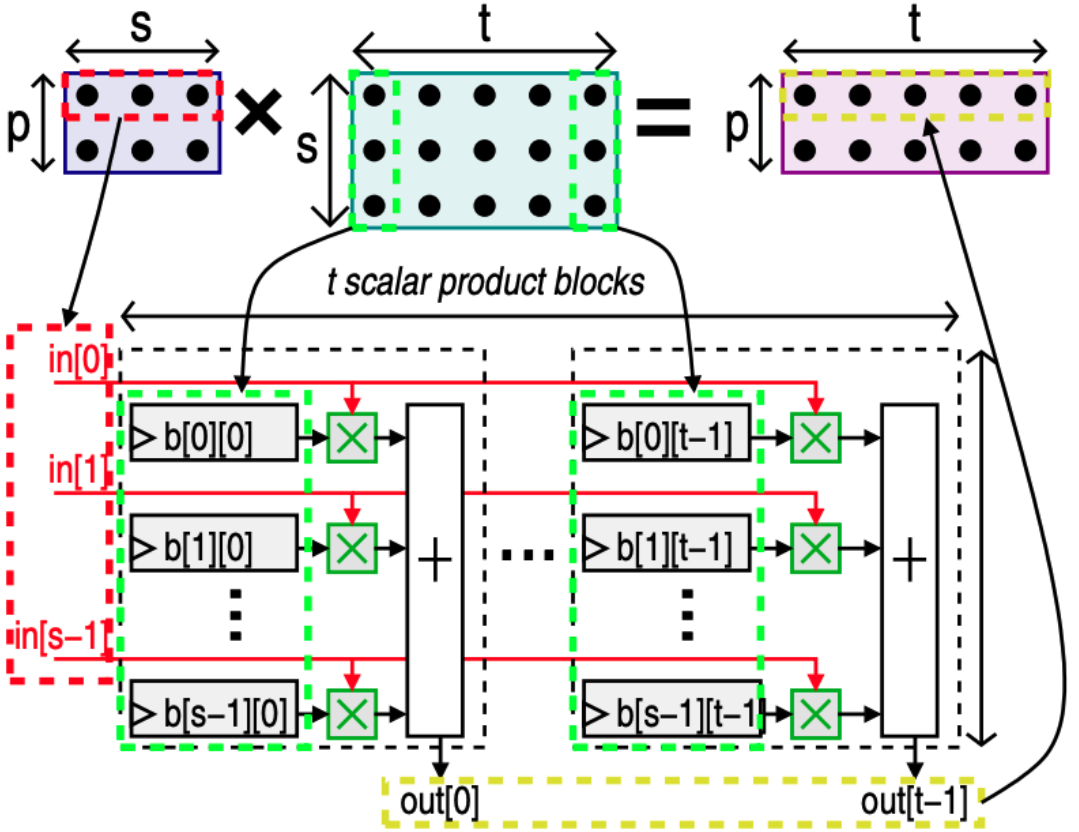
- Built with 36 6-T SRAMs
- Able to read/write data once a ROW/Column



Dot Production Block Diagram Explained

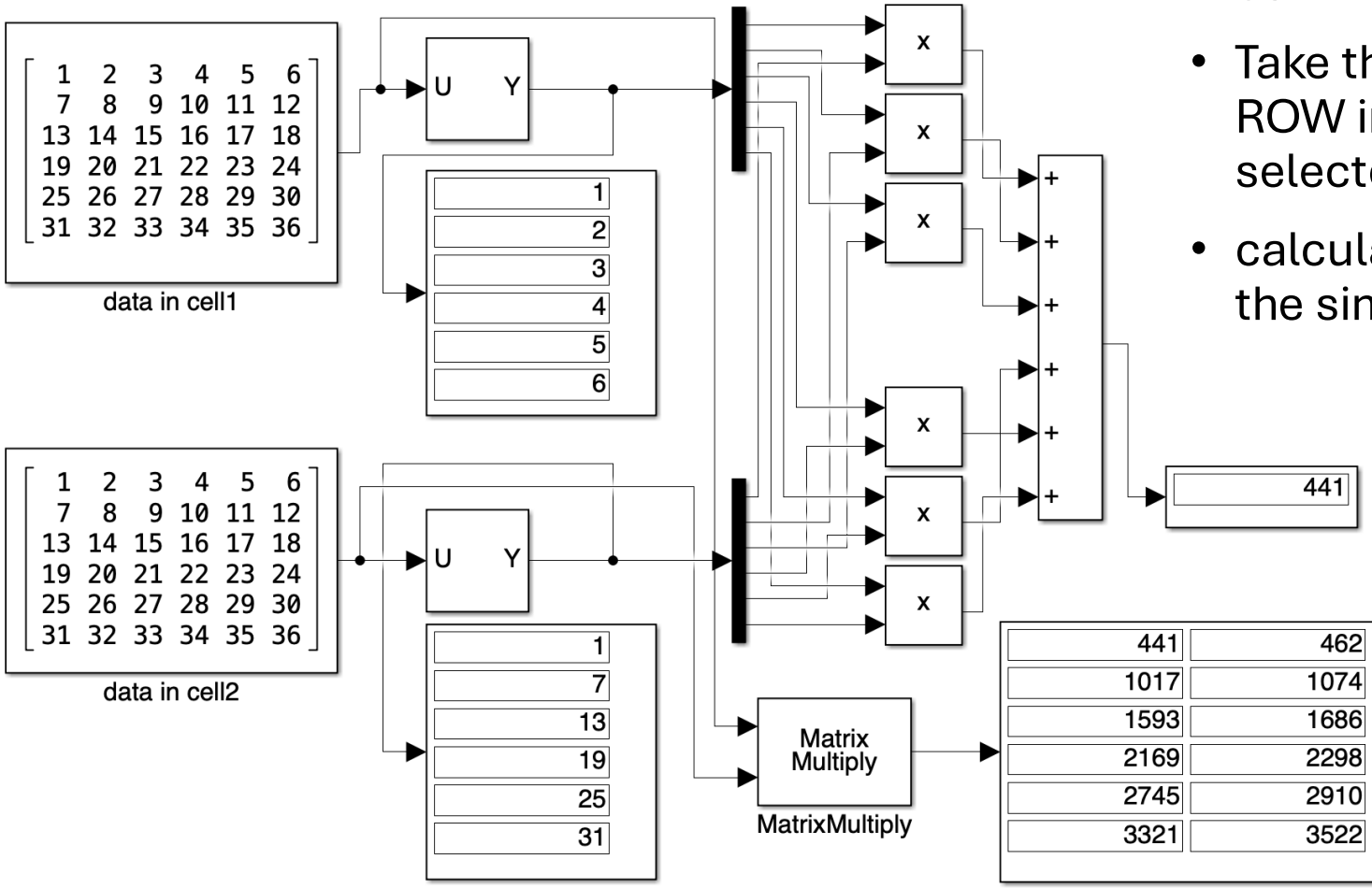
- The Dot Production of matrix can be divided into two stages: **Multiplication** and **Sum**
- Every calculation cycle would read one ROW (a) from Array1 and one Column (b) from Array2
- **Multiply** corresponding element together to get six values (eg. a[0]b[0], a[1]b[1]), **Sum** the six values to get one element of the output Matrix OUT[a, b]
- Repeat the calculation for every ROW/Column in Array1 and Array2, store the result in a 16-bit Memory Array





Simulation Result for Dot Production

- To verify the calculation algorithm, the ‘Matrix Multiply’ module is used
- The correct result for dot production in cell1 and cell2 have shown in the table
- Take the value OUT[0, 0] for example, the first ROW in cell1 and first Column in cell2 have been selected
- calculation result is 441, which is matches with the simulated result



441	462	483	504	525	546
1017	1074	1131	1188	1245	1302
1593	1686	1779	1872	1965	2058
2169	2298	2427	2556	2685	2814
2745	2910	3075	3240	3405	3570
3321	3522	3723	3924	4125	4326

Timeline

Task	Start Date	End Date	Assigned To	Task Description
Design Confirmation	2.26	3.5	Both	Determine overall architecture, design specifications, and performance goals before starting.
Memory Sche and Pre-layout Simulation	3.6	3.13	Shun	Schematic design of the memory unit and run pre-layout simulations to verify design correctness (power, delay).
Memory Layout and Post-layout Simulation	3.14	3.21	Shun	Finished the layout and do DRC/LVS/PEX, Perform post-layout simulations to validate performance.
Memory-Array Sche (36 & 64 Byte) and Pre-Layout Simulation	3.22	3.25	Shun	Schematic design of the memory array and run pre-layout simulations to verify design correctness.
Memory-Array Layout and Post-layout Simulation	3.26	3.28	Shun	Finished the layout and do DRC/LVS/PEX, Perform post-layout simulations to validate performance.
Memory Controller Sche and Pre-layout Simulation	3.6	3.13	Yue	Schematic design of the memory controller and run pre-layout simulations to verify design correctness.
Memory Controller Layout and Post-layout Simulation	3.14	3.21	Yue	Finished the layout and do DRC/LVS/PEX, Perform post-layout simulations to validate performance.
Register Schematic and Pre-Layout Simulation	3.22	3.24	Yue	Schematic design of the register and run pre-layout simulations to verify design correctness.

Timeline

Task	Start Date	End Date	Assigned To	Task Description
Register Layout and Post-Layout Simulation	3.25	3.29	Yue	Finished the layout and do DRC/LVS/PEX, Perform post-layout simulations to validate performance.
Dot-Product Sche and Pre-Layout Simulation	3,30	4.6	Yue	Schematic design of the dot-Product and run pre-layout simulations to verify design correctness.
Dot-Product Layout and Post-Layout Simulation	4.7	4.14	Yue	Finished the layout and do DRC/LVS/PEX, Perform post-layout simulations to validate performance.
State Machine Sche and Pre-Layout Simulation	3.29	4.6	Shun	Schematic design of the state machine and run pre-layout simulations to verify design correctness.
State Machine Layout and Post-Layout Simulation	4.7	4.14	Shun	Finished the layout and do DRC/LVS/PEX, Perform post-layout simulations to validate performance.
Top-Cell Sche and Pre-Layout Simulation	4.15	4.16	Both	Schematic integration for the 6x6 matrix multiplier and run pre-layout simulations to verify design correctness.
Top-Cell Layout and Post-Layout Simulation	4.17	4.22	Both	Finished the whole layout and do DRC/LVS/PEX, perform post-layout simulations to validate 6x6matrix multiplier performance.
Check and Ready for Tape-Out	4.23	4.30	Both	Check each block (schematic, layout, performace), DRC/LVS clean, performance and ready for tape-out.

Measurement Plan

- **Functional verification:**

Basic Arithmetic Correctness Test: Send matrix data to the input of the chip through the MCU and compare the calculation results with the theoretical calculation values to verify whether the matrix multiplication is correct using logic analyzer.

- **Performance Evaluation:**

Computation Latency Measurement: Use a high-precision chronograph to record the computational latency from the input matrix to the output result and compare it to simulation data to evaluate whether the logic optimization is working as expected.

Maximum Operating Frequency Test: Gradually increase the clock frequency to test the stability of the chip at different frequencies and determine its maximum reliable operating frequency.

Power Consumption Measurement: Use a power analyzer to measure the operating power.

- **Typical Application Scenarios**

Artificial Intelligence & Machine Learning: Acceleration of small matrix operations in Convolutional Neural Networks (CNNs) to improve AI computational efficiency.

Digital Signal Processing (DSP): For image processing, edge detection, filtering, and other scenarios to increase processing speed.

Embedded Computing & Low Power Applications: Fast matrix computation in IoT devices to improve energy efficiency and reduce computing resources.