

MIPS Processor Design

Course: Computer Organization

Faculty: Pratik Trivedi

Group Number B 03

Group Members:

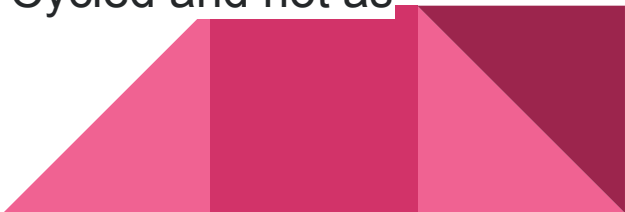
Ridham Shah (AU-1841007)

Harshil Mehta (AU-1841010)

Vidish Josh (AU-1841019)

Manav Patel (AU-1841036)

Motivation

- At one point (in the 90s) MIPS-derived processors were the best selling processors in the world, dwarfing sales of Intel x86 processors.
 - To understand the concepts of Microprocessor without interlocked pipelined stages. (MIPS)
 - To create a microprocessor and get awareness regarding assembly language.
 - To implement and learn the Verilog language to make MIPS processor.
 - Pondering upon, we have designed and added features to current MIPS processor.
 - Motivation for MIPS processor was to make a balanced processor having optimal facilities and hardware; not simple like Single Cycled and not as complex as CISC processors.
- 

SPECIFICATIONS

- This is an 8-Bit processor.
- Total number of instructions in this processor is - 28.
- The size of the address bit is 8.
- Type of Instruction Set Architecture(ISA) is - RISC.
- Speed-up factor of this processor is 5 (In comparison to Single cycle processor).
- The CPI for very large number of instructions is 1.



SPECIFICATIONS

- ❖ There are 3 types of instructions that can be operated in this processor:
 - J-Type:
 - These type of instructions have only 2 parts in their instructions. One for the opcode and the second part for the locations to jump to.
 - R-Type:
 - These type of instructions have 4 parts in them. One for the opcode and three for memory locations on which operation has to be done and store the result.
 - I-Type:
 - These type of instructions have 3 parts in them. One for the opcode and two for the immediate value to be moved and and the location where it has to be moved.

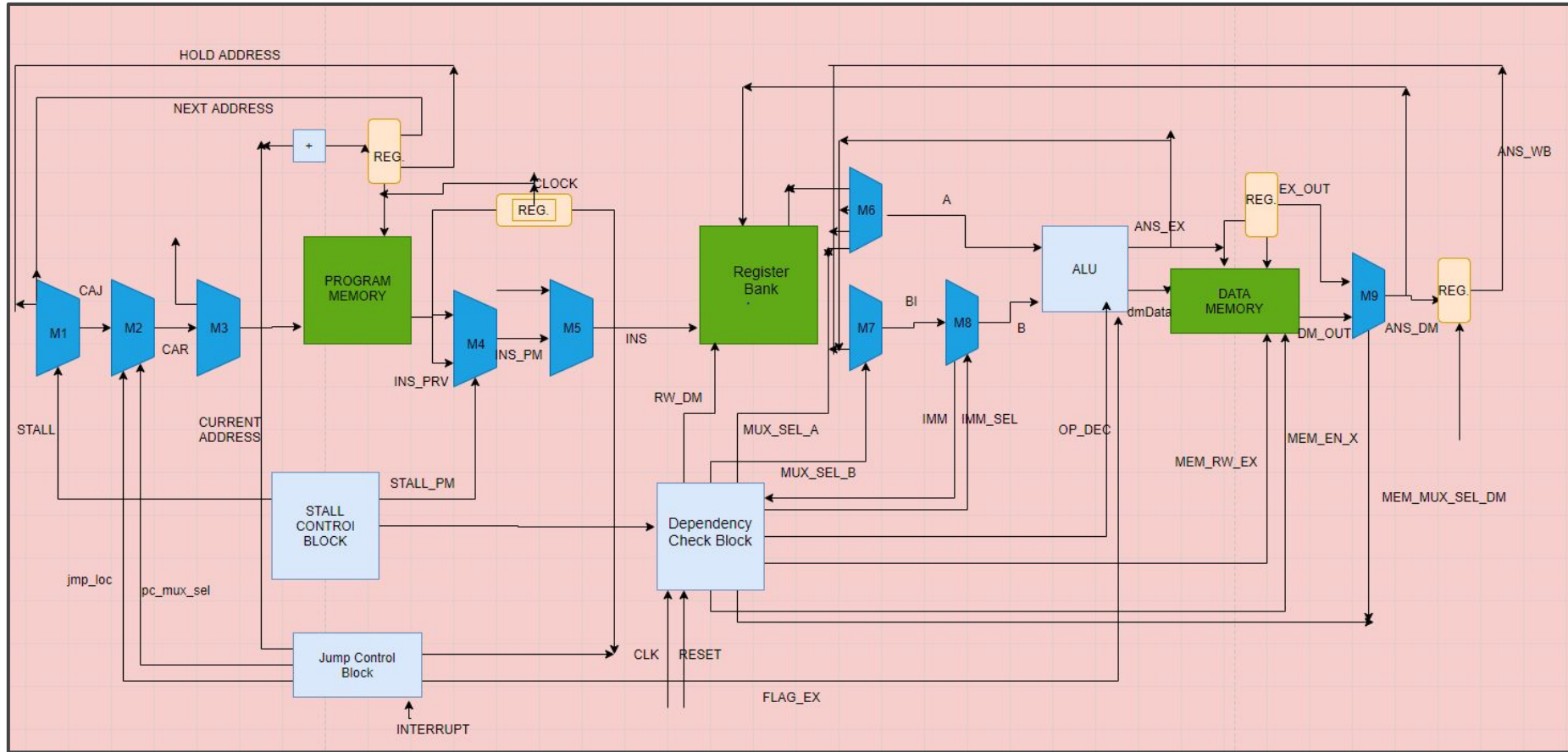


SPECIFICATIONS

- Type of Addressing Mode used to point to data locations are: Direct, Register Direct and Immediate Direct.
- The frequency of this processor is 1MHz.
- Total 32 registers are used.
- We have also added other opcodes for Wallace multiplication and parity checker.



MIPS ARCHITECTURE(BLOCK DIAGRAM)



COMPARISON OF MIPS WITH 8085

<u>Comparisons</u>	<u>8085</u>	<u>MIPS</u>
<u>Architecture</u>	Neumann	Harvard
<u>RISC/CISC</u>	Semi-CISC	RISC
<u>Pipelined</u>	No	Yes
<u>Registers</u>	6	32
<u>Address line</u>	16 Bit	20 Bit
<u>Instructions</u>	80	28
<u>Accumulator</u>	8 Bit	No
<u>Frequency</u>	Frequency is 3-6 Mhz	Frequency is 100 Mhz

COMPARISON OF MIPS WITH 8085(Cont.)

Different ALU Flags:

1. MIPS: flags are used as branch instructions.
2. 8085: flags are stored in a condition codes register (flip flop flags are used).

Temporary Registers to store Data:

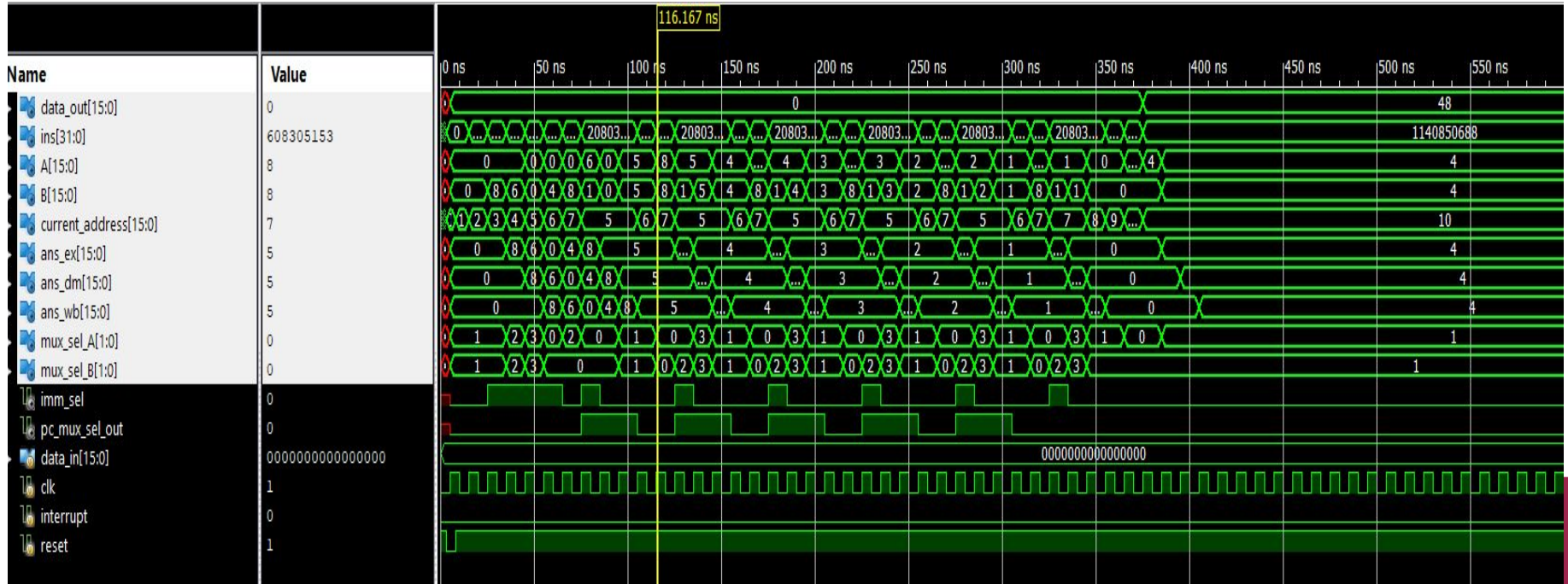
1. 8085: Temporary registers for Overflow and Negative to hold to data for logical operations while in MIPS they are not.



POST ROUTE RESULT OF ADDITION

[illegible]

POST ROUTE RESULT OF MULTIPLICATION



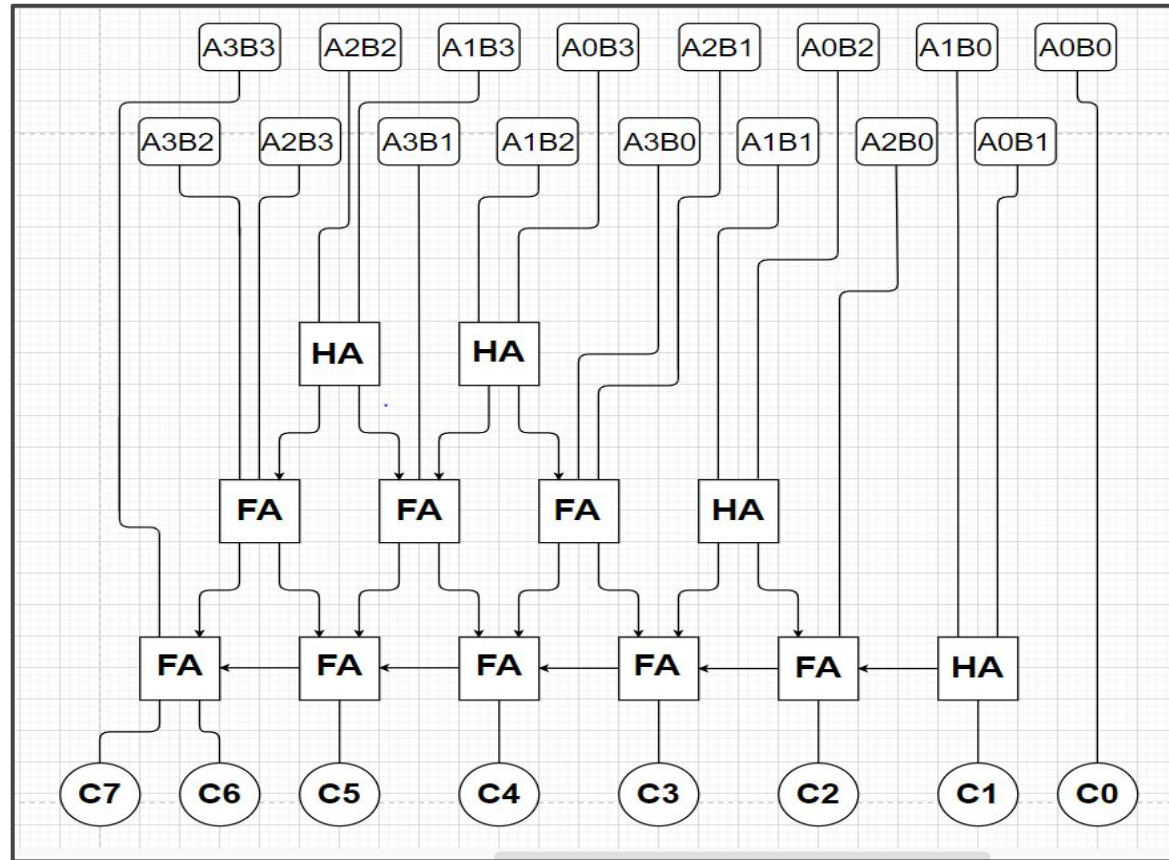
Advantages of Wallace Multiplication

1. Used in fast 3D computer graphics and floating point precision in higher level softwares.
2. Total per stage execution time is much less than compared to any other algorithms(Multiplicative).
3. As it uses Half-Adder and Subtractor, it does the operation in constant time.
4. It also takes optimum number of steps to compute answer for multiplication and sums up the at the last.
5. It takes partial products and shifts them and adds at last.
6. Easily pipelined and easily scheduled.

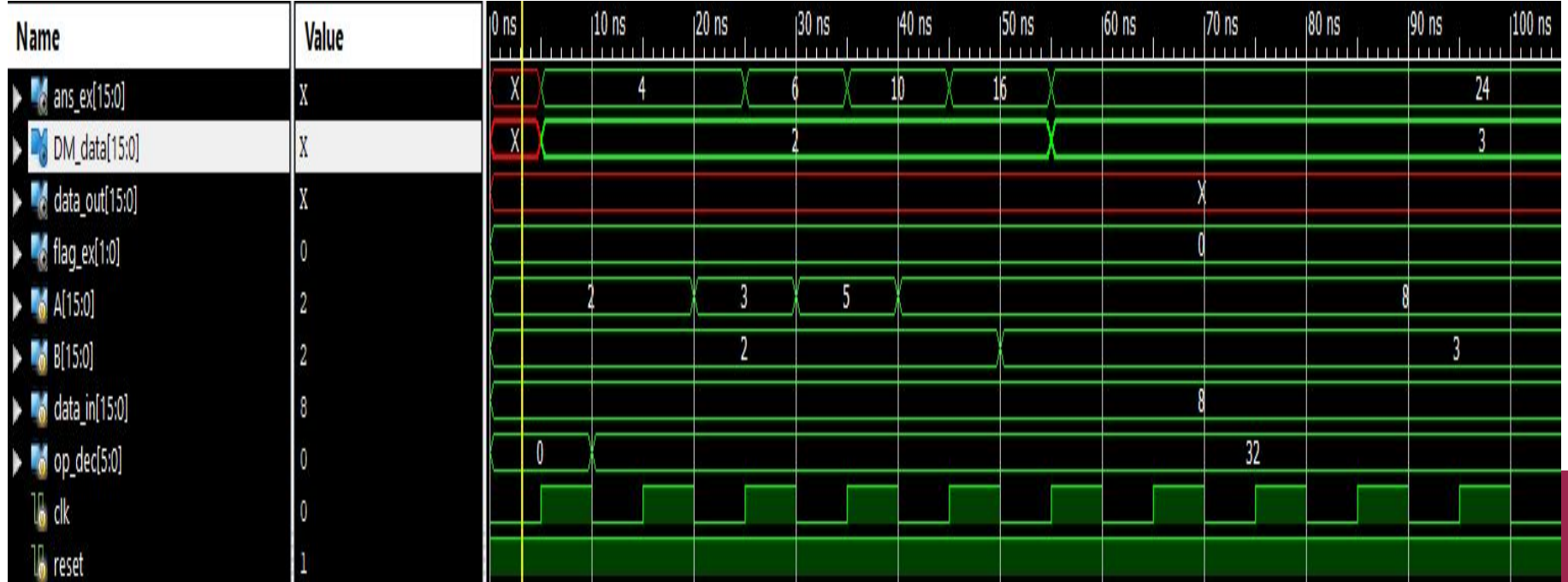
Binary multiplication

$$\begin{array}{r} \begin{array}{r} 1\ 1\ 0\ 1 \\ \times 1\ 0\ 1\ 1 \\ \hline 1\ 1\ 0\ 1 \\ 0\ 0\ 0\ 0 \\ 1\ 1\ 0\ 1 \\ \hline 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \end{array} \quad \begin{array}{l} (13)_{10} \text{ Multiplicand M} \\ (11)_{10} \text{ Multiplier Q} \\ \left. \begin{array}{l} 1\ 1\ 0\ 1 \\ 0\ 0\ 0\ 0 \\ 1\ 1\ 0\ 1 \end{array} \right\} \text{Partial products} \\ (143)_{10} \text{ Product P} \end{array}$$

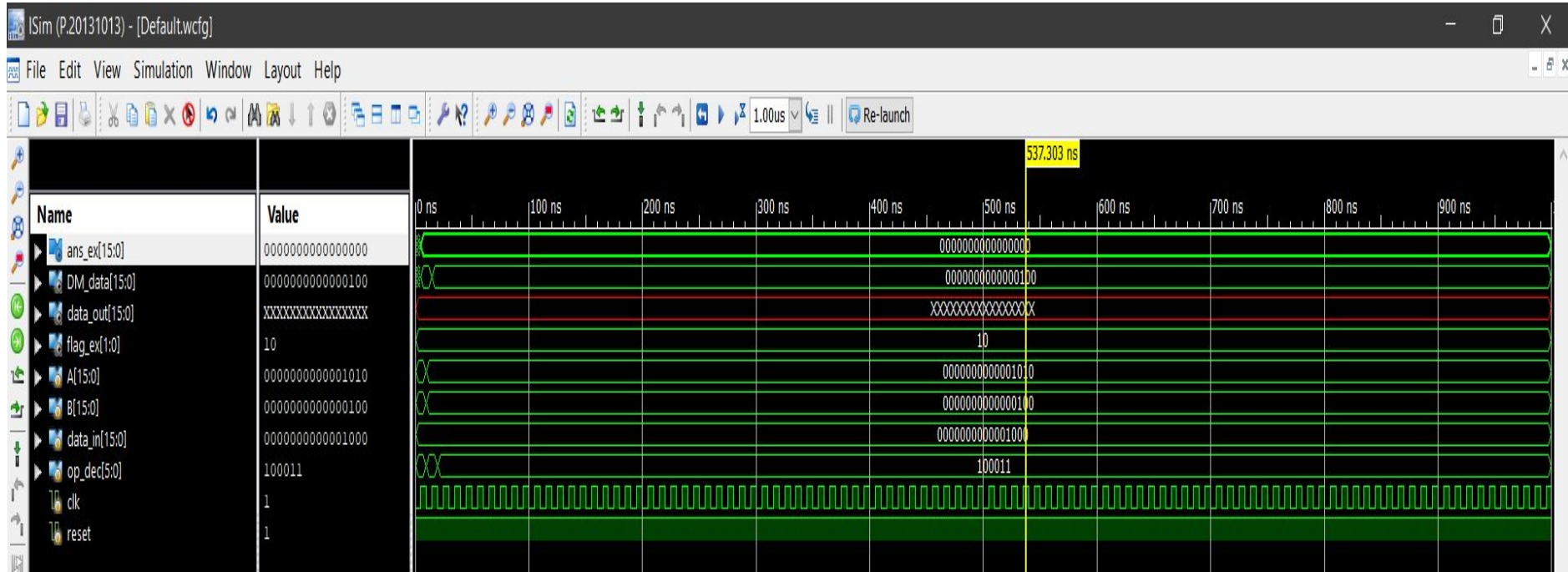
WALLACE MULTIPLIER BLOCK DIAGRAM



WALLACE MULTIPLIER



PARITY CHECKER:

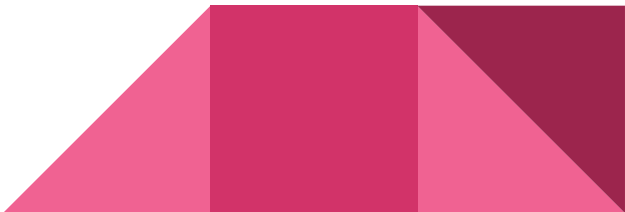


CHALLENGES FACED DURING DESIGN

- We faced difficulties during implementation of the code to get *multiplication* instructions. Addition instruction was pretty easy to implement.
- But during implementation of *multiplication* instructions, we could not get correct output and we found problems in jump control block and execution block. Those problems/errors were resolved after which we got the correct output.
- Another difficulty that we faced was during the implementation of our additional functionality to make *Undo Operation*.



CHALLENGES FACED DURING IMPLEMENTATION

- We decided to implement Undo Operation by making a Register that would work like a memory storage of the last instruction that was executed.
 - This register will store the address where the output of the last instruction was stored and the value at that location before it gets changed by an instruction. When Undo instruction is encountered, the hold_value is again written at the memory address.
 - This function for undo should work with the delay of one clock cycle than the rest of the components. That was difficult to implement. Also, the program malfunctioned when this module was inserted in it.
 - Also, Integration of all the blocks created separately was a major task as debugging was tough on finding any bugs.
- 

FEEDBACK/SUGGESTIONS OF MIPS PROCESSOR

- Using MIPS designing we can implement instructions with only 1 clock cycle delay rather than Single Cycle process. This designing was optimal and we understood some of the standard processor designs used in the industry.
- Faster ALU can be constructed using pipeline.
- Using pipeline the CPU work at higher clock frequency than RAM.
- Data hazard is almost solved using pipeline.
- Without it simple and cheaper system can be build.
- It increases instruction latency.
- It is hard to predict through-put.



THANK YOU

