Design and Layouting of 2-bit Vector-array co-processing unit

Ajay Kumar M – 13EC102
Gururaj K – 13EC120
B Shravan Kumar – 13EC209

Contents

Abstract

Clock Frequency of CPUs are reaching their Limits. In this scenario performance gain can only be achieved through processing multiple data parallely. Two classical approaches to parallel processing namely Array processing and Vector processing have their own pros and cons. This project aims at combining the advantages of both Array processing (faster) and Vector processing (area efficient). As a proof of concept here we Design and Layout 2-bit Vector-array processing ALU.

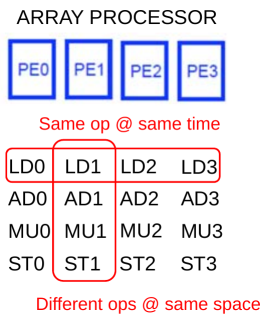
0.1 INTRODUCTION

0.1.1 ARRAY PROCESSING

In this approach, the idea is to process multiple data elements using multiple ALUs simultaneously. Therefore during each instruction only one of the functional units is active. The advantages of this approach are

- High Speed
- Simple Control Unit

The main disadvantage of this approach is that it is **not Area Efficient**



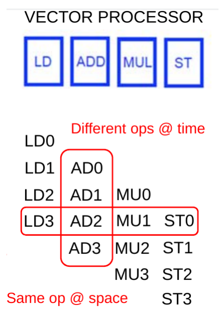
Plot of time vs space over different instructions

0.1.2 VECTOR PROCESSING

In this approach, the idea exploit the different functional of a single ALU to simultaneously process multiple data elements over multiple clock cycles in a pipelined fashion. The advantages of this approach are

- Area Efficient
- Improved performance

The disadvantage of this approach is speed improvement is not as pronounced as in Array processing.



Plot of time vs space over different instructions

0.1.3 COMBINED APPROACH

The idea is to use the pipelined approach as in case of Vector processing, exploiting the parallelism between functional units but instead of processing a single element, functional units are designed to process suitable number of data elements simultaneously as in case of array processing. In this way, the new design has better performance than vector processor and also is more area efficient than an array processor.

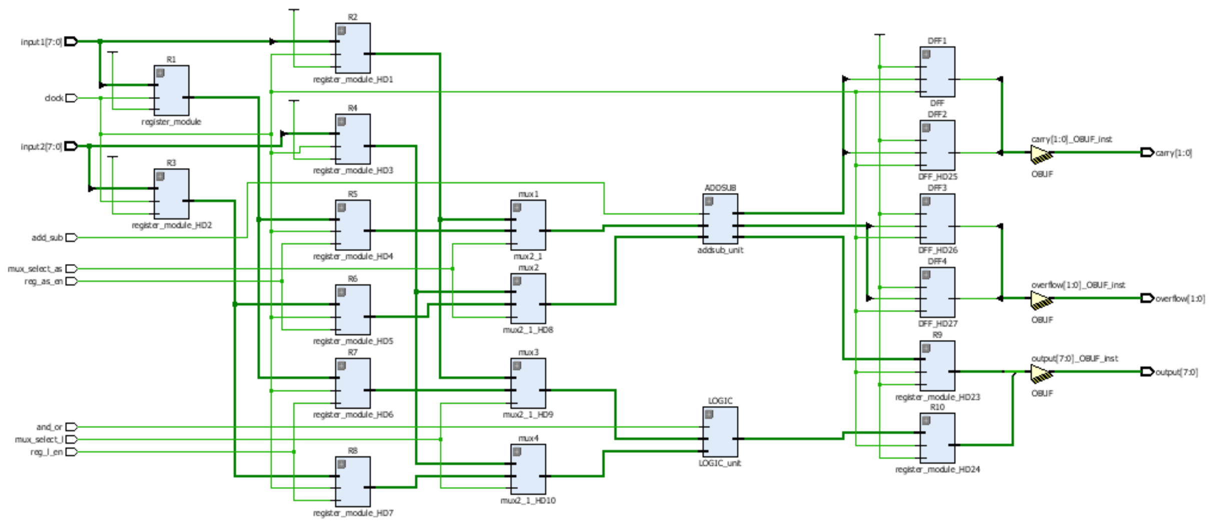
0.2 DESIGN

The following are the specifications for which the Design was made

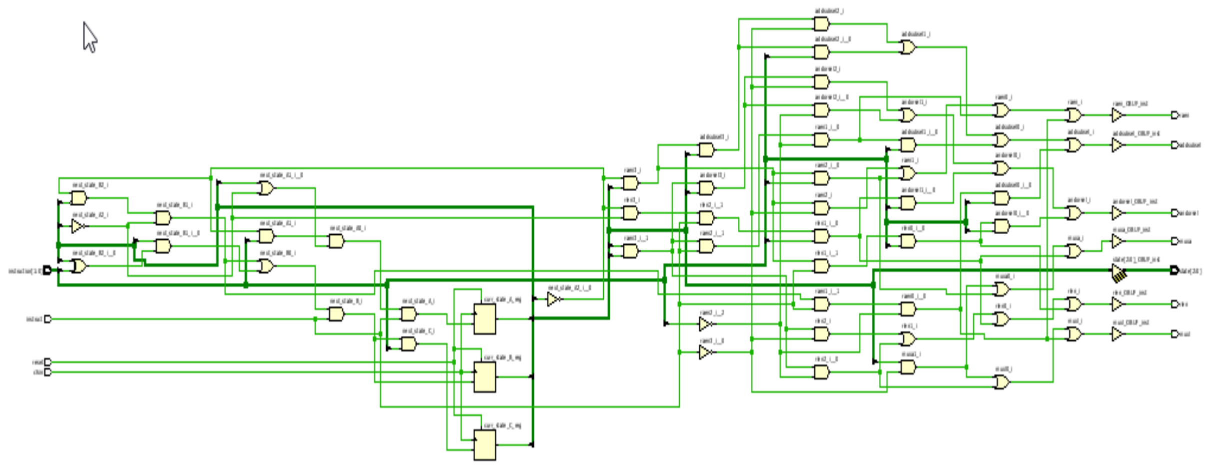
- The word length is 2 bits
- Each vector register contains 4 independent words
- The instruction set has only 4 valid instructions
  - ADD
  - SUBTRACT
  - LOGICAL AND
  - LOGICAL OR
- There are 2 functional units, one for arithmetic instructions and one for logical instructions
- Each functional unit has the can process 2 words simultaneously
- There is additional **NO OPERATION** instruction useful for stalling the pipeline

0.2.1 SCHEMATIC

The Design was done using **VHDL**.



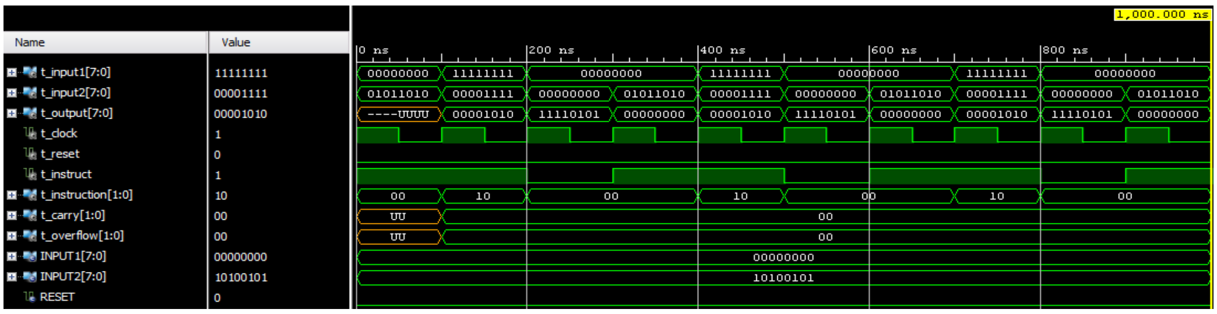
SCHEMATIC OF THE DATAPATH



SCHEMATIC OF THE CONTROLPATH

0.2.2 SIMULATIONS

The simulations of the Design and testing was done in **XILINX VIVADO**.
The Design was tested by generating **Test Vectors for all combinations of INSTRUCTIONS**.



Name	Value
t_input1[7:0]	11111111
t_input2[7:0]	00001111
t_output[7:0]	00001010
t_clock	1
t_reset	0
t_instruct	1
t_instruction[1:0]	10
t_carry[1:0]	00
t_overflow[1:0]	00
INPUT1[7:0]	00000000
INPUT2[7:0]	10100101
RESET	0

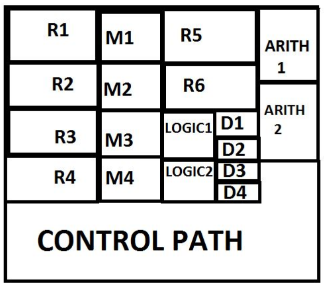
Sample Timing Simulation for a random case

0.3 LAYOUT

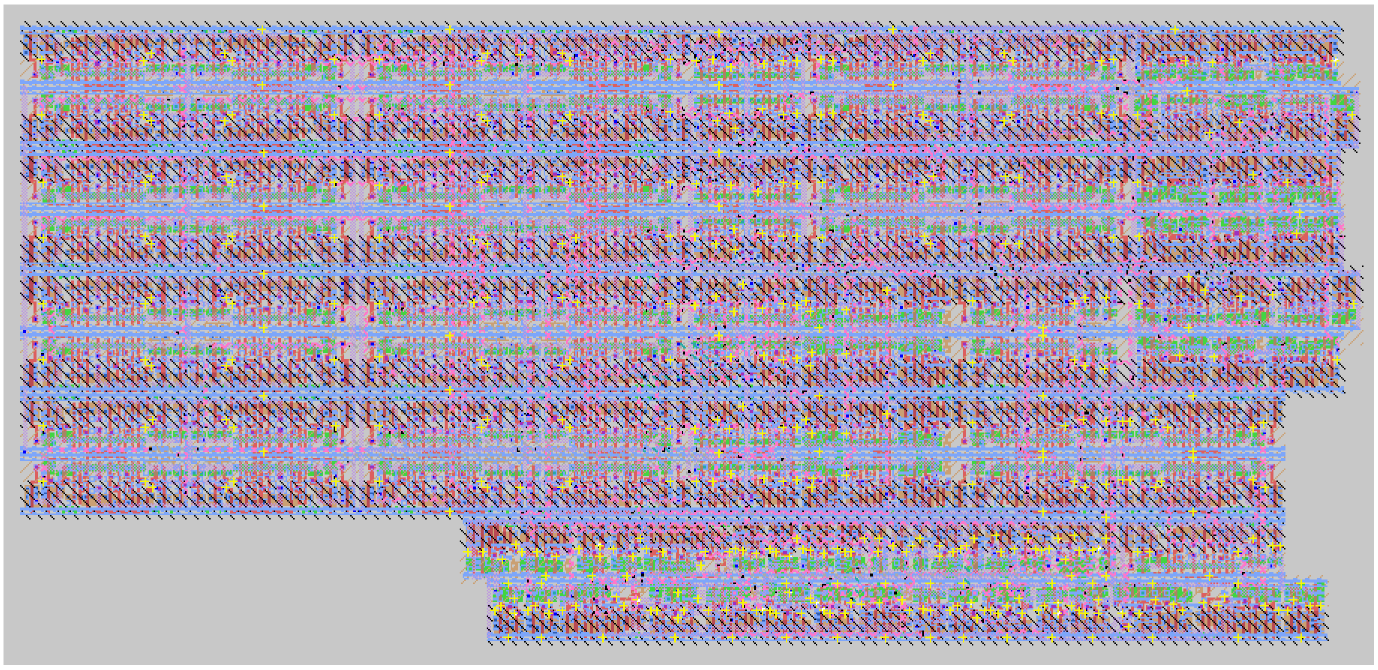
Layouting was done using **MAGIC**.

0.3.1 FLOORPLAN

Floorplanning was done before layouting to optimize for AREA.



0.3.2 LAYOUT IMAGE



0.4 MAJOR SPECIFICATIONS

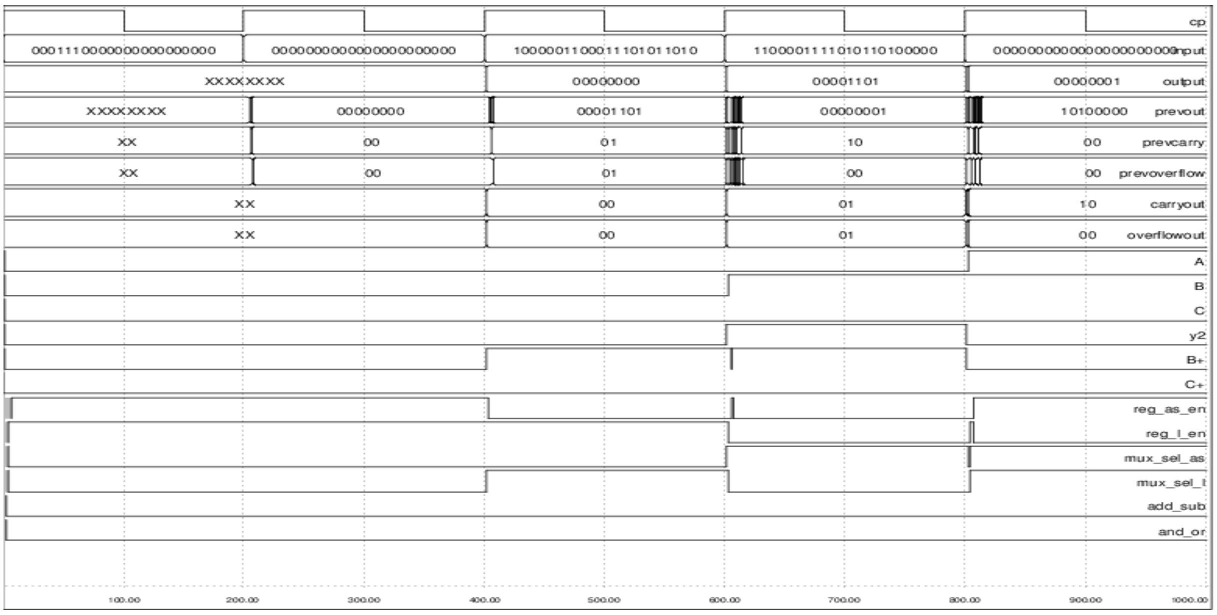
0.4.1 DELAY OF BLOCKS

BLOCK	DELAY
D Flip Flop	0.217ns
Multiplexer	0.107ns
Arithmetic Sum	0.785ns
Arithmetic Carry	0.841ns
Arithmetic Overflow	1.02ns
Logical AND	0.273ns
Logical OR	0.436ns

- Total Nodal Capacitance = 1.773 pF
- Critical Path Delay = 3.63 ns
- Maximum Operating Clock Frequency = 275.4MHZ

0.5 VERIFICATION

The LAYOUT was extracted and was simulated using **IRSIM**.
The extracted netlist was tested against all combinations of INSTRUCTIONS and results were displayed in **ANALYSER** window.



Sample Timing Simulation for a random case

The obtained results match exactly with the expected results got from VHDL Simulation.

0.6 FUTURE SCOPE

Generating Test Vectors from Instructions can be automated. C Program which decodes the instructions and generates appropriate Test Vectors can be written. The generated Test Vectors can be automatically passed to IRSIM using scripting and the results can be obtained and processed into desired form. In this way we can include programmability.

This Design can be extended to wider word lengths and more functional units without disturbing the underlying assumptions.

0.7 REFERENCES

1. *http://www.eecs.berkeley.edu/ krste/thesis.pdf*

2. *Vector processing by David A. Patterson and Jan Rabaey*

3. *Roger Espasa, Mateo Valero, James E. Smith, "Vector Architectures: Past, Present and Future",2008*

4. *C. Kozyrakis, D. Patterson. "Overcoming the Limitations of Conventional Vector Processors",in ISCA,2003*

5. *Bharat Bhushan Agarwal, Sumit Prakash Tayal, "Computer Architecture and parallel processing",2001*

6. *LY Bucher and M.L.Simmons. Performance Analysis of Supercomputers in Vector and Parallel Processor Architecture,Applications.National Laboratory Report LA-UR-85-1805,1985*