Lab03 – Pipelined Arithmetic and Logic Unit

Dragomir MILOJEVIC, v3.1

1 Objectives

Design simple Arithmetic & Logic unit (ALU). Implement hierarchical design with multiple ALUs. Implement non-pipelined & pipelined computation, derive performance parameters & compare.

2 Exercise

2.1 Simple ALU

Design Arithmetic & Logic Unit (ALU) able to perform logic and arithmetic computations on two operands `A` and `B` coded on 8 bits (`A[7:0], B[7:0]`) and and additional input `Cin`. The result is a 8-bit vector `Dout[7:0]`. Overflow could be signaled by the ALU using an output `Cout` (bonus point). Desired operation is chosen using 3-bits instruction opcode. Possible operation and opcode association is illustrated in Table 1. Feel free to adapt. Note that the Table suggest operation with signed operands. Complement addition/subtraction simply invert the input operand.

S2	S1	S0	Cin	Function	Operation
0	0	0	x	$A + B$	OR logic
0	0	1	x	$A \cdot B$	AND logic
0	1	0	x	$A \oplus B$	XOR
0	1	1	x	$\overline{(A \oplus B)}$	XNOR
1	0	0	0	A	Transfer
1	0	0	1	$A + 1$	Increment
1	0	1	0	$A + B$	Addition
1	0	1	1	$A + B + 1$	Addition/incrementation
1	1	0	0	$A + \overline{B}$	A plus C1B
1	1	0	1	$A - B$	A minus B
1	1	1	0	$\overline{A} + B$	C1A plus B
1	1	1	1	$B - A$	Subtraction

Table 1: Operation description

The ALU top-level interface is shown in Figure 1.

Write circuit model in Verilog using arithmetic and logic operators. Note that the ALU will be implemented as combinatorial circuit, so inputs/outputs will NOT being registered. We assume ALU will be operating with a Register File, implemented elsewhere.

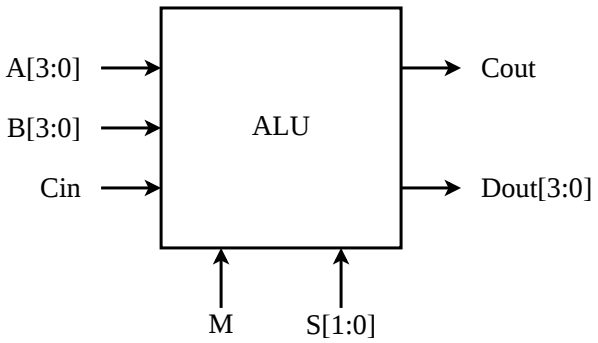

Figure 1: ALU interface

2.2 ALUs in series

Implement a top-level design with 3 ALUs connected in series. The output of 1st ALU will be connected to one operand of the second ALU; the other operator for 2nd ALU should be considered as 3rd input word at top-level. Finally, the output of the second ALU will be connected as first operand of the 3rd ALU. The other operand for 3rd ALU should be considered as 4th input word at top-level.

Write a test-bench for the circuit above to simulation addition of 4 vectors:

$$D[i] = A[i] + B[i] + C[i] + D[i]$$

Sum $A[i] + B[i]$ will be computed by ALU1, sum $(A[i] + B[i]) + C[i]$ will be computed by ALU2, and finally $((A[i] + B[i]) + C[i]) + D[i]$ will be computed by ALU3.

Implement the circuit and derive minimum period (maximum frequency).

2.3 Pipelined ALUs

Using the previous model, insert FF stages where appropriate to enable pipelined execution. Compare performance parameters between pipelined and non-pipelined version. What can you conclude?

Discuss different options for the FF insertion (e.g. at module or top-level). What are the trade-offs?

3 Questions to prepare

1. How did you deal with the overflow?
2. Where did you put the FF, module or top-level?
3. What is the frequency difference?