CSE 2021
COMPUTER ORGANIZATION

HUGH CHESSER
CSE B 1012U

W7-M

Agenda

Topics:
1. Basics: Clock, Latches and Flip Flops
2. Sequential and Combinational Circuits

Patterson: Appendix C, Section 4.1, 4.2, 4.3

This Week in CSE 2021

Allowed aids for the midterm:

- Green Sheet
- 8.5 x 11 Cheat Sheet

WEEK	WEEK OF	Mon	Wed	Lab	Topic
1	Sep 05	-	✔	-	Overview of the course
2	Sep 12	✔	✔	-	Performance and Data Translation
3	Sep 19	✔	✔	A	Code Translation
4	Sep 26	✔	Quiz #1	B	Translating Utility Classes
5	Oct 03	✔	✔	C	Translating Objects
6	Oct 10	-	-	-	READING WEEK - No Classes
7	Oct 17	✔	Mid-term	D	Introduction to Hardware
8	Oct 24	✔	✔	Make-up Labs	Machine Language + Floating-Point
9	Oct 31	✔	✔	K	The CPU Datapath
10	Nov 07	✔	Quiz #2	L	The Single-Cycle Control
11	Nov 14	✔	✔	M	Pipelining
12	Nov 21	✔	✔	N	Caches
13	Nov 28	✔	Quiz #3	Make-up Labs	
14	Dec 05	✔	-	-	No lecture on Wednesday

Requirements for Cheat, Instruction Sheets (Midterm and Final Exam)

Cheat Sheets MUST:

- Be no larger than 8.5" x 11" in size
- Information MUST be original handwriting – NO printed or copied cheat sheets are permitted
- A single sheet, writing permitted on both sides

Green (MIPS Instruction) Sheets:

  - MUST be extracted from textbook
  - OR you may use a print out of the pdf version from the course web site

What is the difference between these two ALUs?

— The 1-bit adder is supplemented with AND and OR gates

— A multiplexer controls which gate is connected to the output
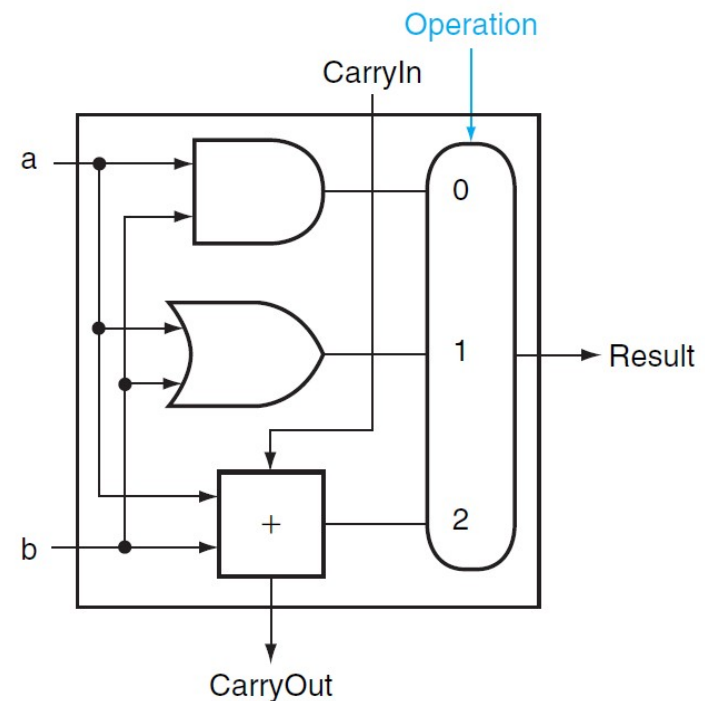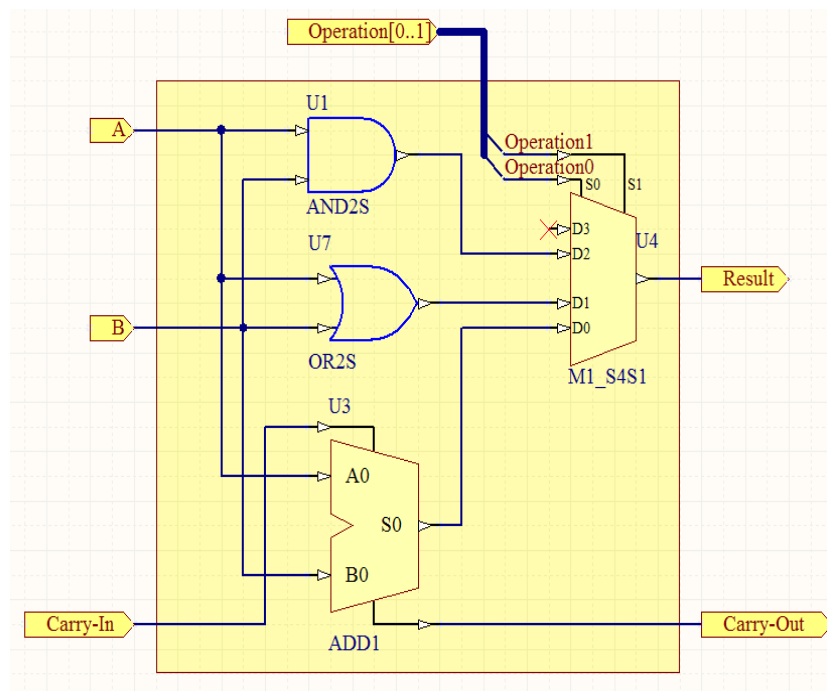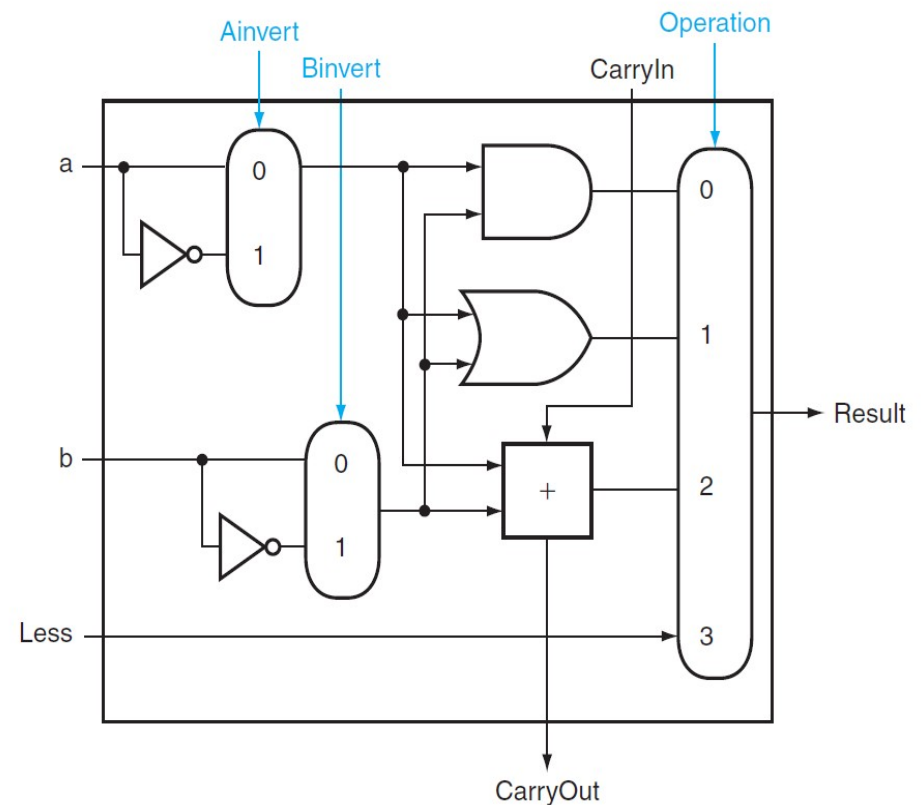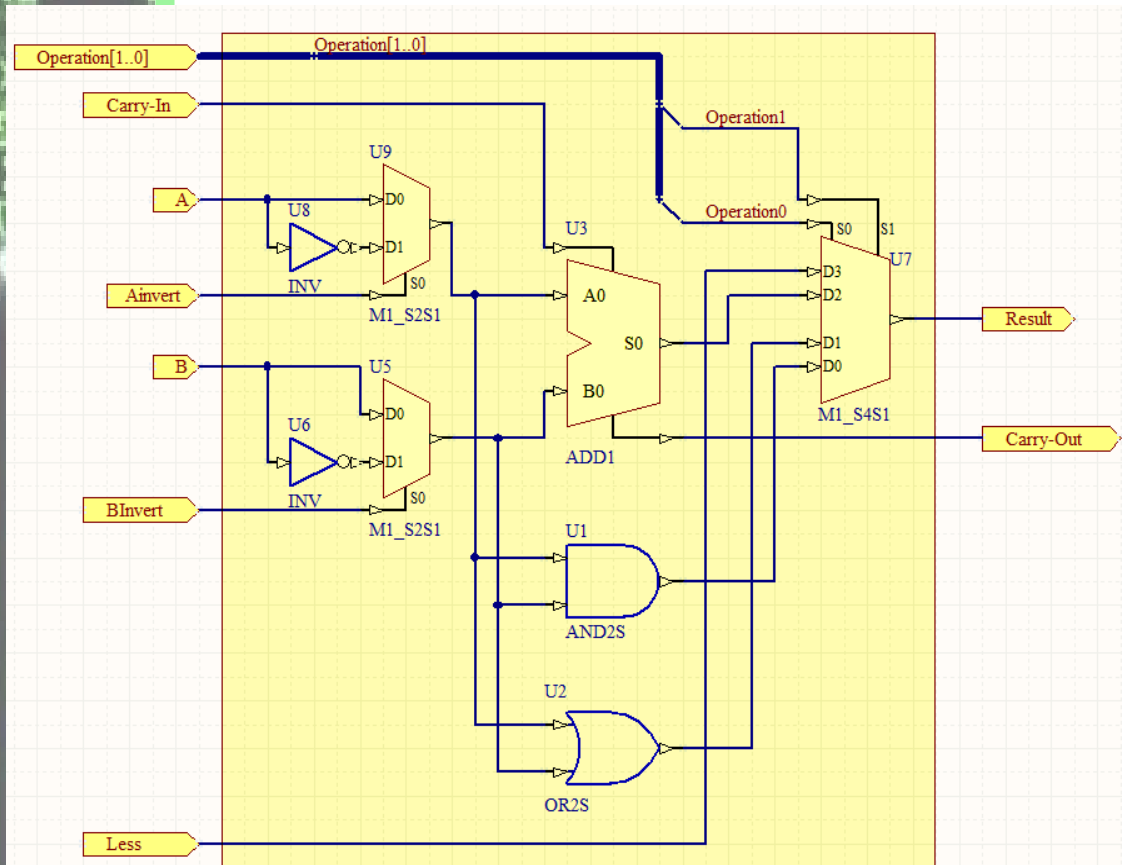

FIGURE C.5.6 A 1-bit ALU that performs AND, OR, and addition (see Figure C.5.5).

1-bit ALU with AND, OR, and Addition capability

SLT (and Nor) Logic Added

— The 1-bit adder is supplemented with AND and OR gates

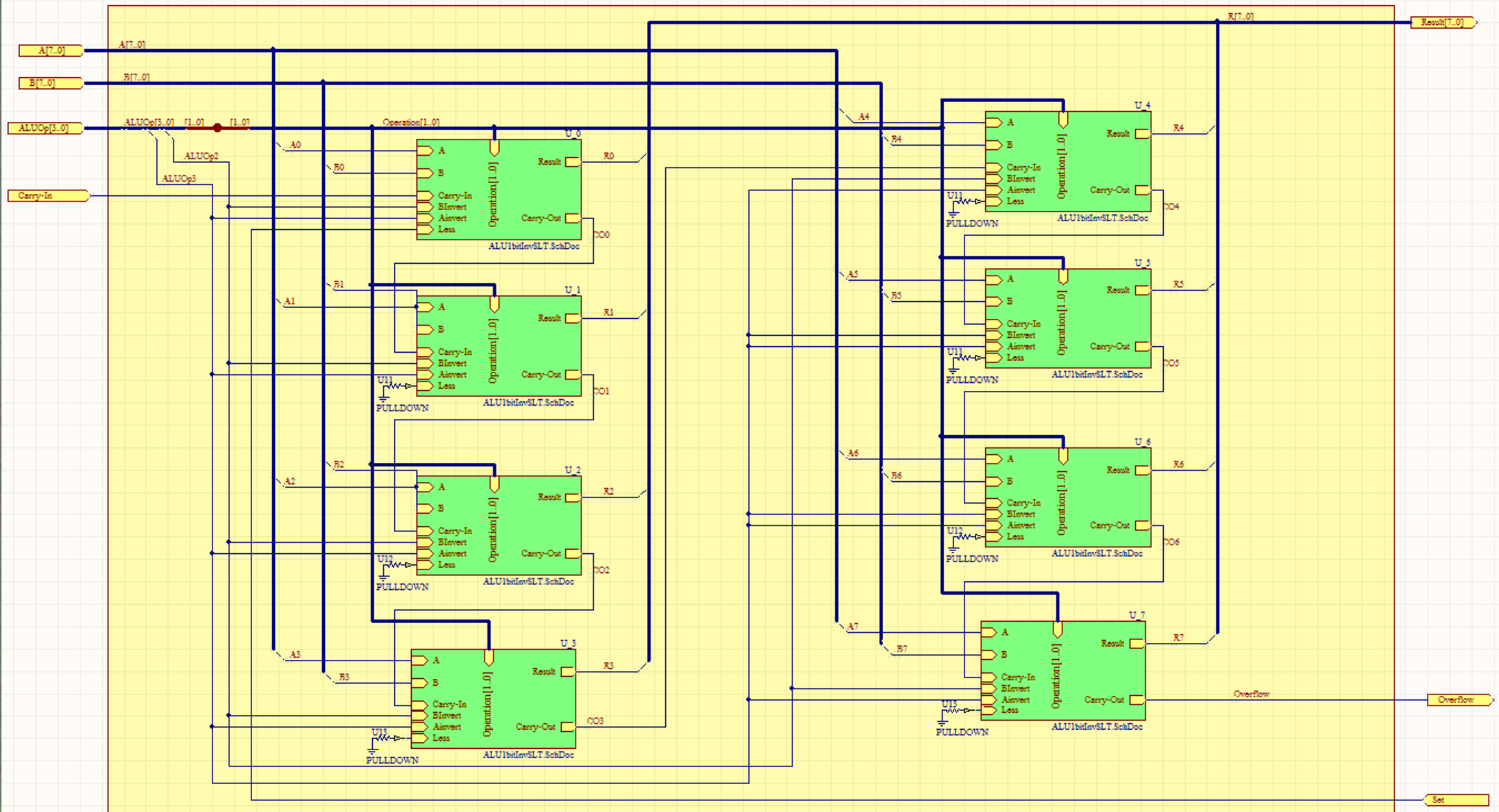— A multiplexer controls which gate is connected to the output

From Last time...

32-bit ALU w/ And, OR, Nor, Add, Subtract, SLT, and Equality Test

ALU Control Lines (ALUOp[3..0])				Result
Ainvert	Binvert	Carry In	Operation	
0	0	0	$0 = (00)_{two}$	AND (a·b)
0	0	0	$1 = (01)_{two}$	OR (a+b)
1	1	0	$0 = (00)_{two}$	NOR ($\bar{a}\cdot\bar{b}$)
0	0	0	$2 = (10)_{two}$	Add sum(a,b)
0	1	1	$2 = (10)_{two}$	Subtract (a - b)
0	1	1	$3 = (11)_{two}$	SLT if (a < b) Result0 = 1
0	1	1	$2 = (10)_{two}$	Test Equality Zero = 1 if (a < b)

32-bit ALU w/ And, OR, Nor, Add, Subtract, SLT, and Equality Test
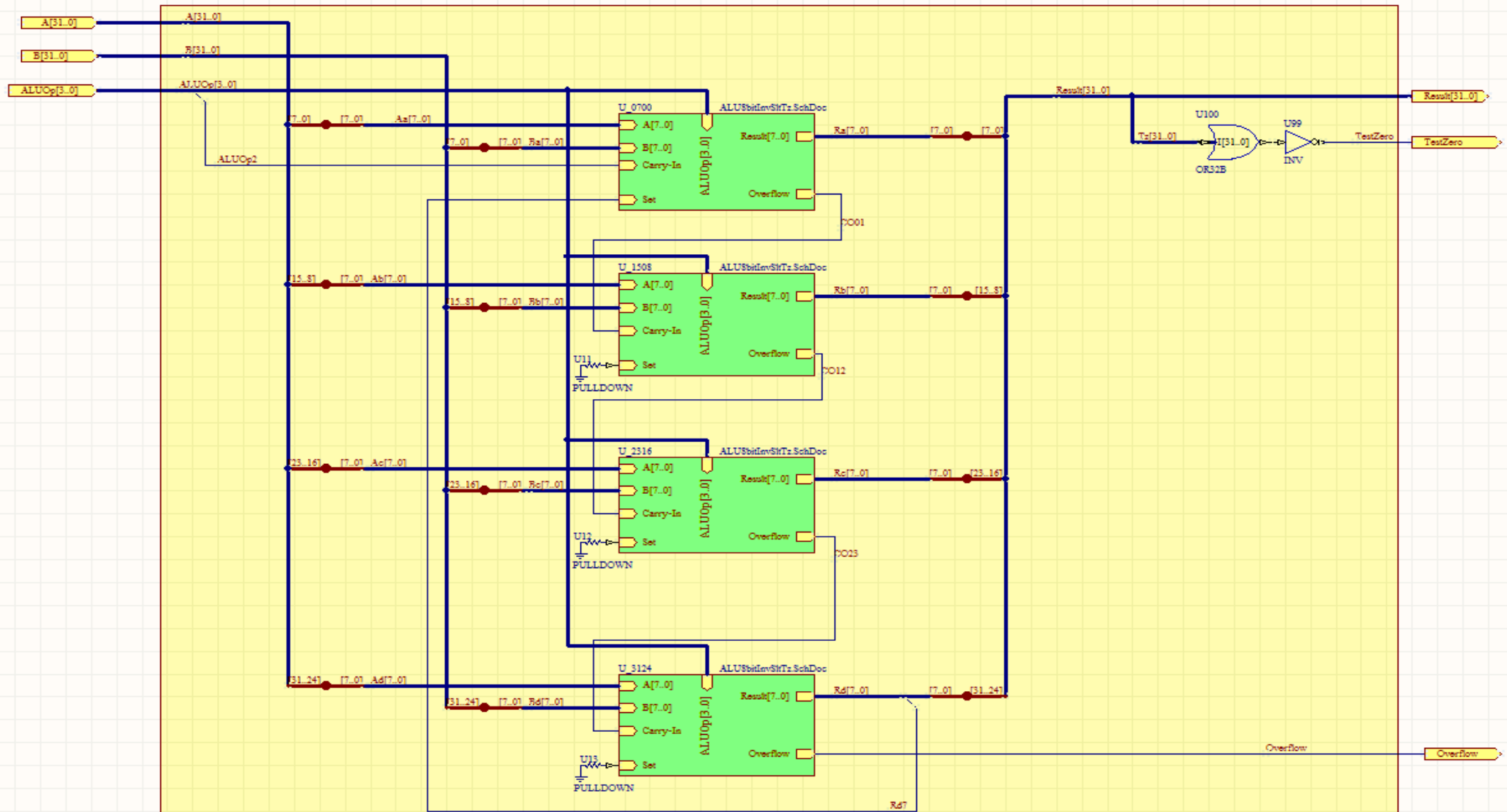
Overview (1)

Goal: Implement a subset of core instructions from the MIPS instruction set, given below

Category	Instruction	Example	Meaning	Comments
Arithmetic and Logical	**add**	add $s1,$s2,$s3	$s1 ← $s2+$s3	
	subtract	sub $s1,$s2,$s3	$s1 ← $s2-$s3	
	and	add $s1,$s2,$s3	$s1 ← $s2&$s3	**& => and**
	or	or $s1,$s2,$s3	$s1 ← $s2|$s3	**| => or**
	slt	slt $s1,$s2,$s3	If $s1 < $s3, $s1←1 else $s1←0	
Data Transfer	**load word**	lw $s1,100($s2)	$s1 ← Mem[$s2+100]	
	store word	sw $s1,100($s2)	Mem[$s2+100] ← $s1	
Branch	**branch on equal**	beq $s1,$s2,L	if($s1==$s2) go to L	
	unconditional jump	j 2500	go to 10000	

Overview (2)

For each instruction, the first two steps are the same

Step 1:   Based on the address in the program counter (PC), fetch instruction from memory

Step 2:   Read 1 or 2 registers specified in the instruction

Steps 3 and 4 vary from one instruction to another

Step 3:   Perform the arithmetic operation specified by the instruction

   load/store word (sw/lw): add offset to $s2

   (add/sub/and/or): appropriate operation is performed on $s2, $s3

   (beq/slt): compare $s2 and $s3 (requires $s2 - $s3)

   jump (j): calculate address

Step 4:   Complete the instruction
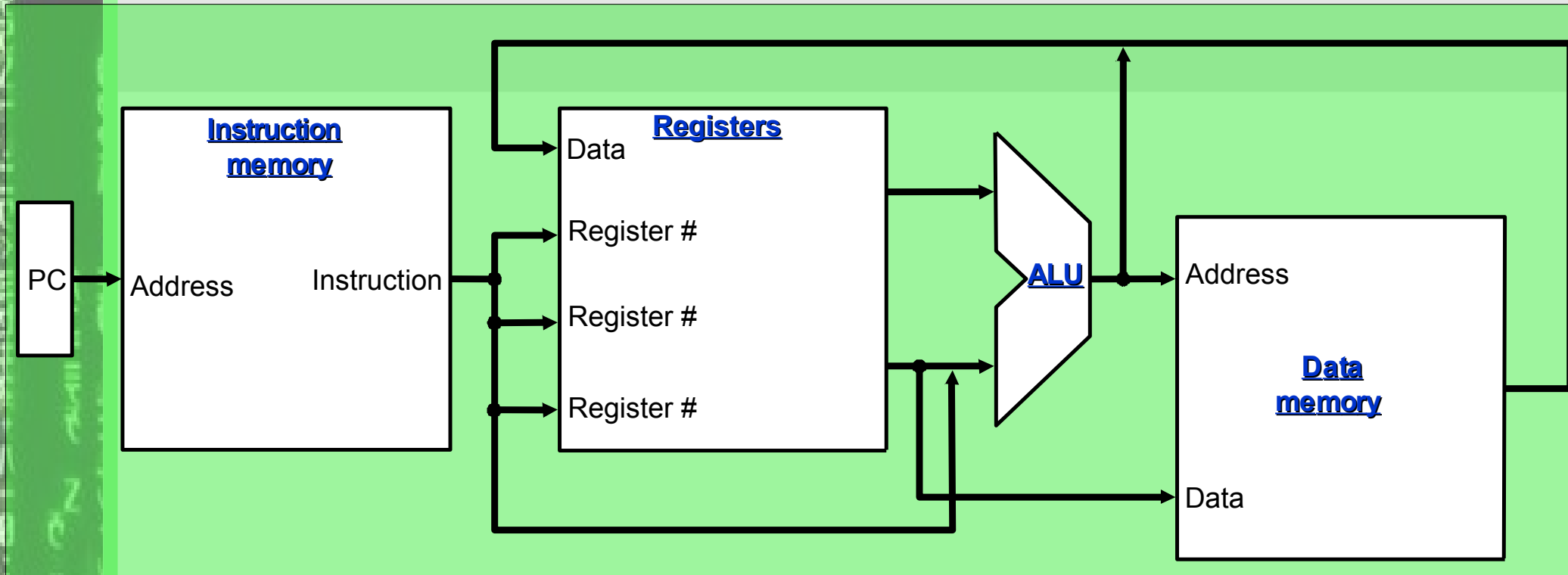
   sw: write data into memory

   lw: read data from memory

   add/sub/and/or: store result in $s1

   beq/j: jump to the appropriate instruction

We will now focus on the hardware implementation of each of these instructions

Abstract view of the Implementation

1. Program counter provides the instruction address
2. Instruction is fetched from instruction memory based on address in the PC
3. Register numbers are specified by the instruction
4. ALU computes an arithmetic result or address of memory
   - Arithmetic operation: Result is saved in a register
   - Data transfer: Data is extracted from data memory & transferred to a register or vice versa

Basics: Sequential vs. Combinational Circuits (1)

Digital circuits can be classified into two categories

1. Combinational Circuits:
   — Output depends only on the current input
   — Same set of inputs will always produce the same output
   — Consist of AND, OR, NOR, NAND, and NOT gates
   — Common examples are adder circuits and ALU

2. Sequential Circuits:
   — Output depends on the current input and state of the circuit
   — Same set of inputs can produce completely different outputs
   — Consist of memory elements such as flip-flops and registers in addition to combinational circuits
   — Examples are traffic signals and street lights

Datapath and control circuits of an ALU use sequential circuits.

Basics: Clocks (2)

1. Clock provides a periodic signal oscillating between low and high states with fixed cycle time.
2. Clock frequency is inverse of clock cycle time. What is the cycle time for 1GHz clock?

Falling edge

Clock period   **Rising edge**
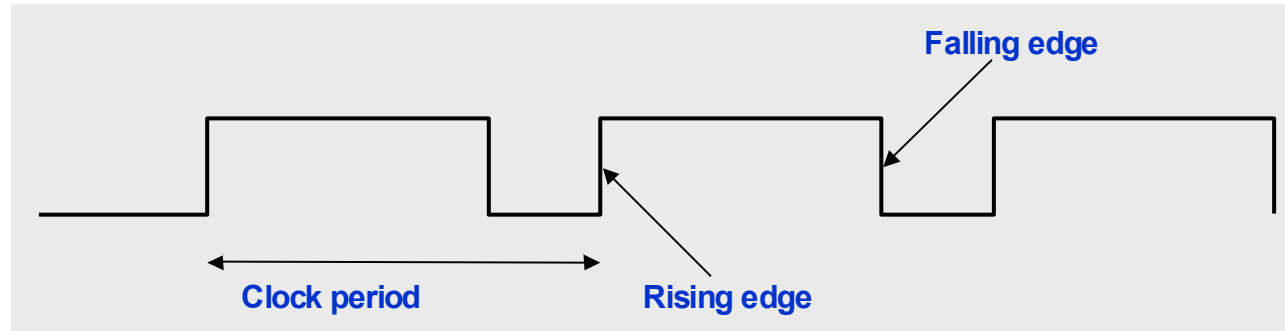
3. Clock controls when the state of a memory element changes.
4. We assume falling edge-triggered clocking implying that the state changes only at the falling edge. When is the state element 2 modified in the following circuit?
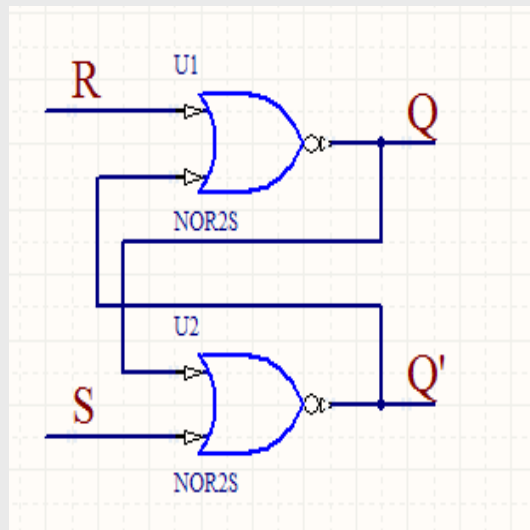
State element 1 → Combinational logic → State element 2

Clock cycle
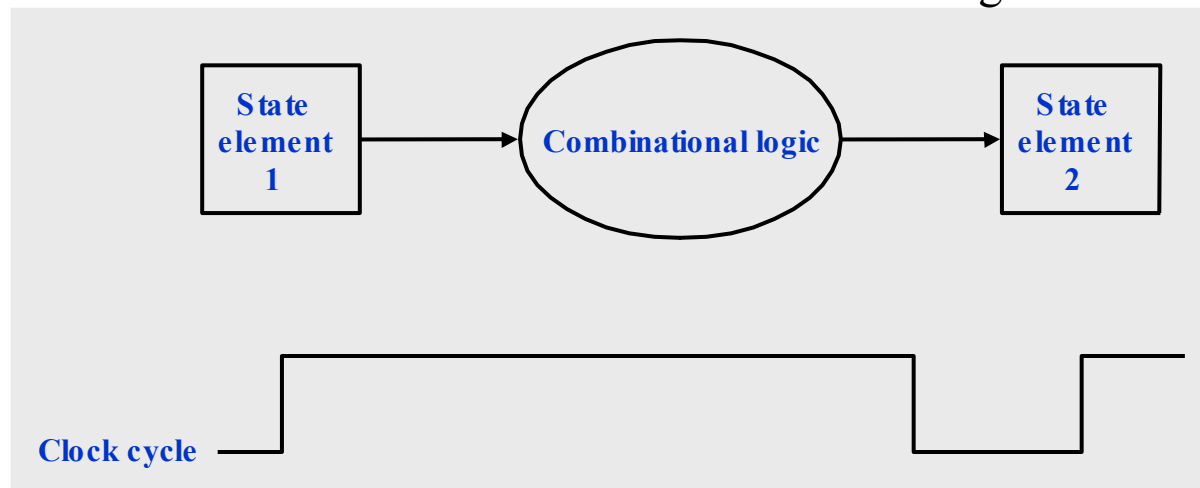
Basics: RS Latch (3)

Simplest memory elements are Flip-flops and Latches

— In clocked latches, state changes whenever input changes and the clock is asserted.

— In flip-flops, state changes only at the trailing edge of the clock
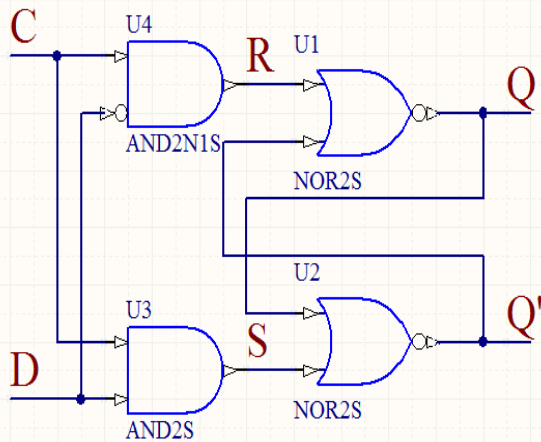
RS Unclocked Latch

Logic Diagram

Inputs		Outputs		Comments
S	R	Q	$Q'=\bar{Q}$	
		0	1	**Initial Condition**
1	0	1	0	
0	0	1	0	**After S = 1, R = 0**
0	1	0	1	
0	0	0	1	**After S = 0, R = 1**
1	1	0	0	**Undefined**

Function Table

— For a RS-latch: output Q = 1 when S = 1, R = 0 (set condition)

output Q = 0 when S = 0, R = 1 (reset condition)

Basics: Clocked D Latch (4)

1. For a D-latch: output Q = 1 when D = 1 (set condition)

   output Q = 0 when D = 0 (reset condition)
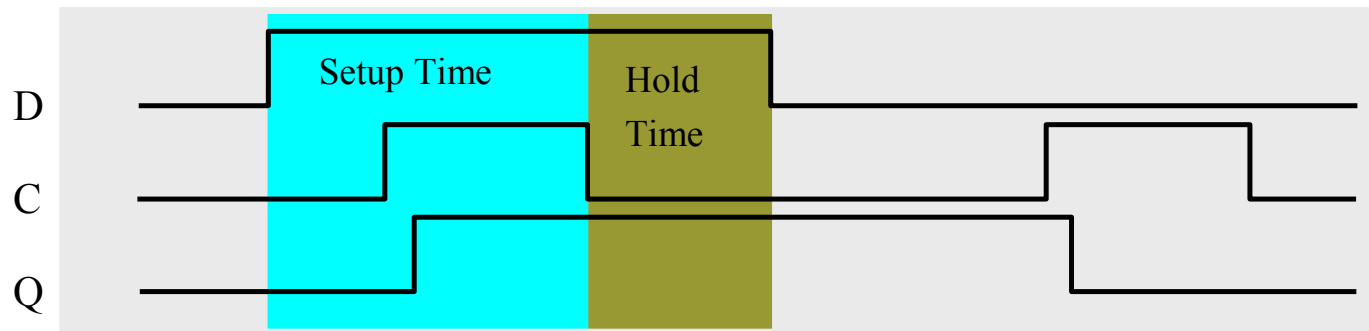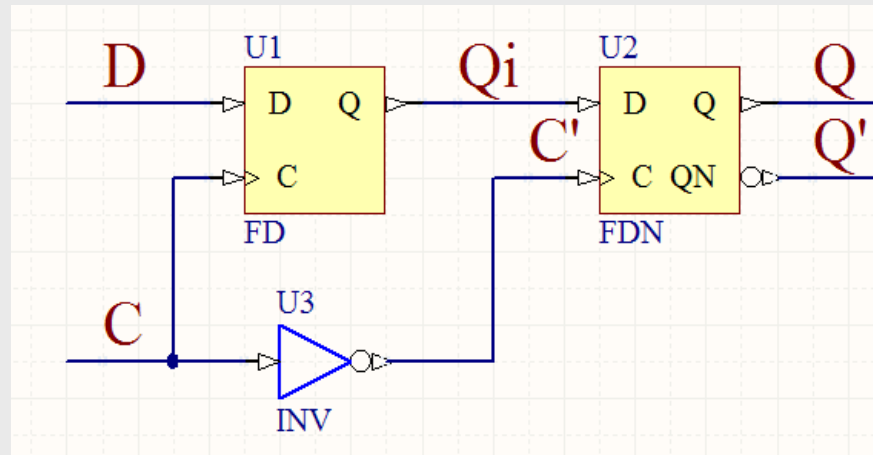

Logic Diagram

Inputs		Outputs		Comments
C	D	Q	$Q'=\bar{Q}$	
0	**X**	**Unchanged**		
1	**0**	**0**	**1**	**Reset**
1	**1**	**1**	**0**	**Set**

Function Table

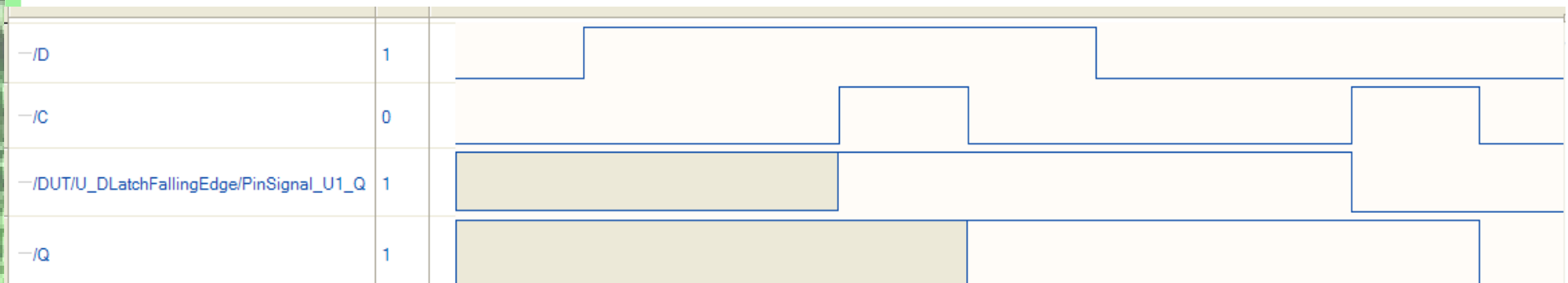2. D Latch requires clock to be asserted for output to change

Basics: Falling Edge Triggered D flip-flop (5)

Logic Diagram

Output Q follows D but changes only at the falling edge

Basics: 32-bit Registers (6)

Falling edge triggered D flip-flops can be combined to form a register