ALU & FSM

Discussion 7

Huizhe Su

Content

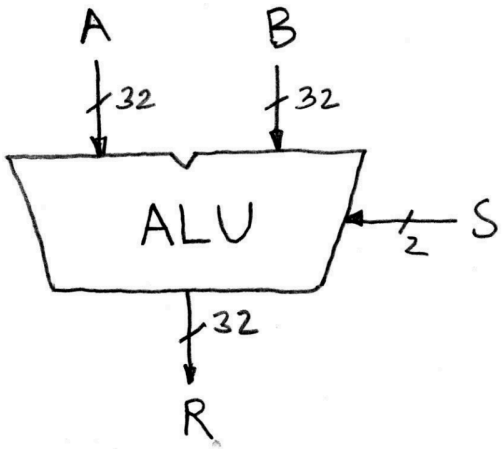
- Combinatorial Logic
  - Arithmetic and Logic Unit (ALU)
  - Logics & Circuits
- Sequential Logic
  - Timing Diagram
  - Time Calculation
- FSM

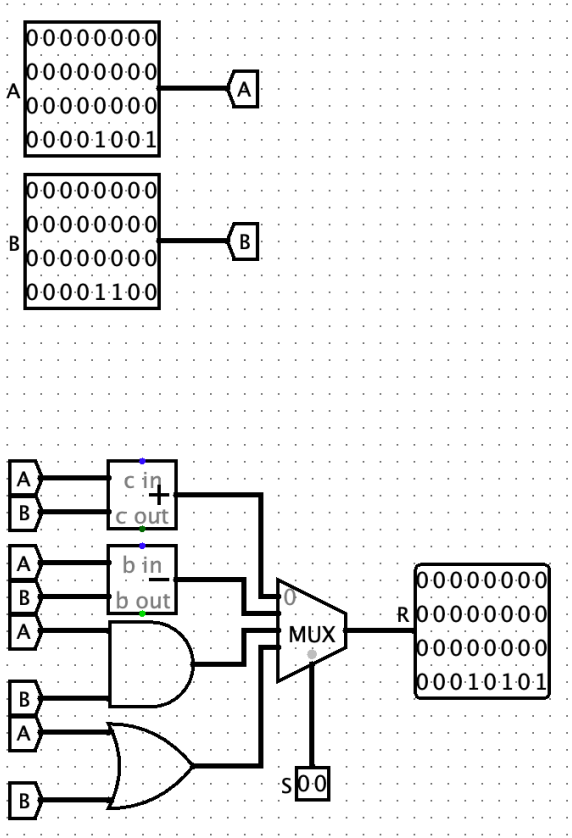
Combinatorial Logic

Arithmetic and Logic Unit (ALU)

Combinatorial Logic

- Arithmetic and logic operations
- Simple example:
  - Input: A, B (32-bit input)
  - Control: S (2-bit input)
  - Output: R (32-bit output)
- Functions:
  - R = A + B, when S = 00
  - R = A - B, when S = 01
  - R = A AND B, when S = 10
  - R = A OR B, when S = 11

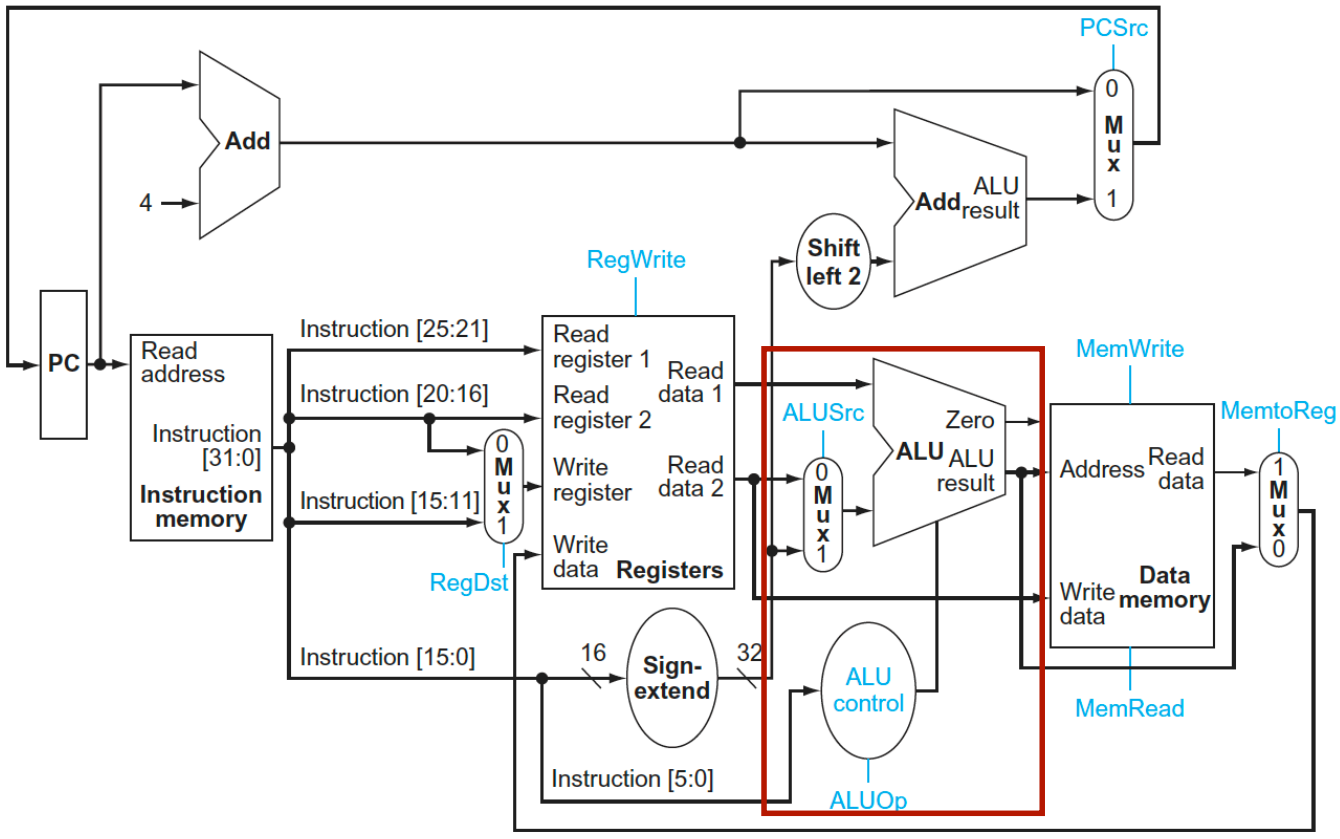




RISC-V ALU

Combinatorial Logic

- ALUSrc
- ALUOp/ALU control
- Zero



Logics & Circuits

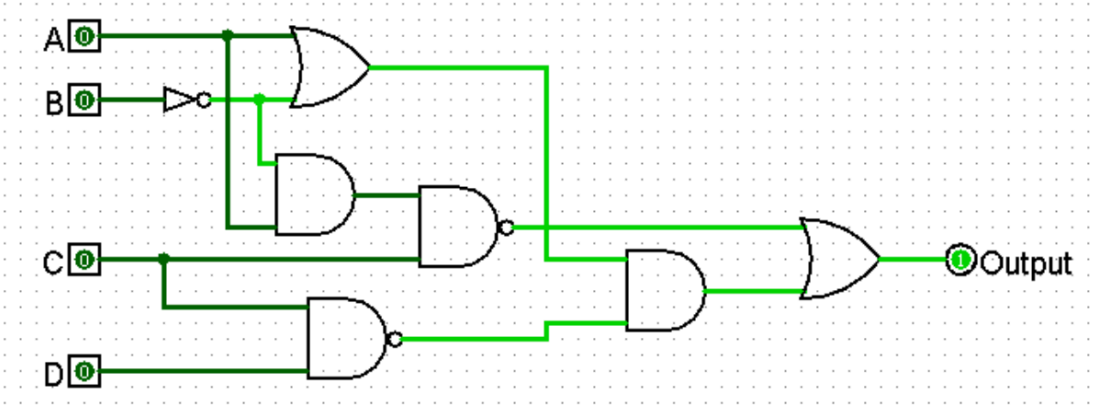
Combinatorial Logic

- Basic things you need to know:
  - The name of each logic gate
  - Write the boolean expression directly from the circuits
  - Simplify the circuits (by boolean expression or truth table)
  - Draw the circuit from the given boolean expression

Logics & Circuits - Example

Combinatorial Logic

(a) The circuit shown below can be simplified. Please **write** down the boolean expression that exactly corresponds to the circuit shown (no simplification). Then simplify this prepossession step by step, applying one rule at a time. Then **draw** the circuit according to the simplified boolean expression using the minimum number of **one-** or **two-input** logic gates.



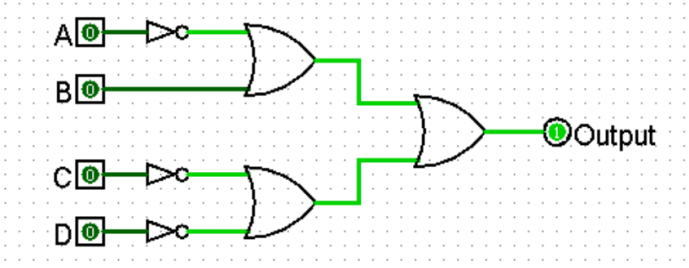
Logics & Circuits - Solution

Combinatorial Logic

Solution:

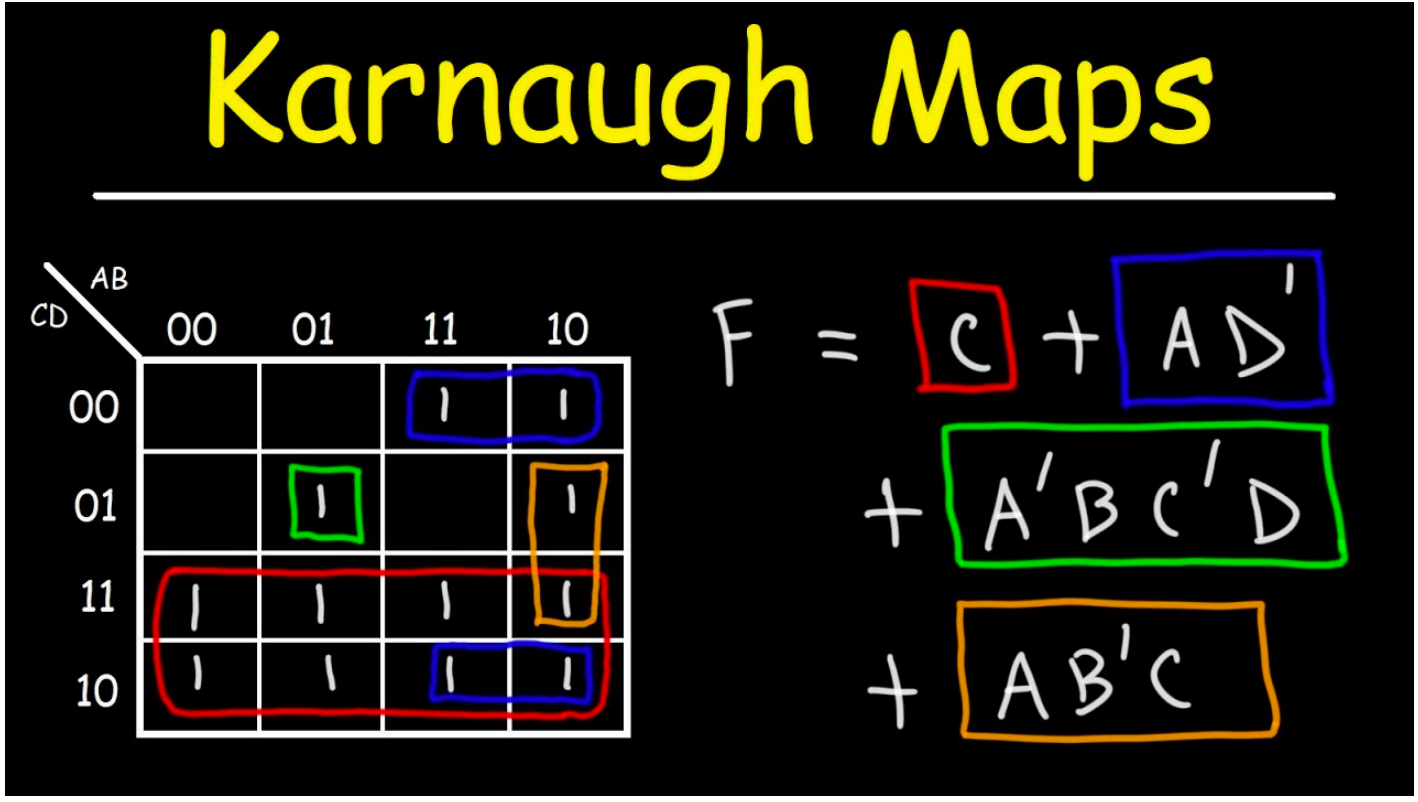
$$\begin{aligned}\overline{A\overline{B}C} + (A+\overline{B})(\overline{C}\overline{D}) &= \overline{A} + B + \overline{C} + (A+\overline{B})(\overline{C}+\overline{D}) \\ &= \overline{A} + B + \overline{C} + A\overline{C} + A\overline{D} + \overline{B}\,\overline{C} + \overline{B}\,\overline{D} \\ &= \overline{A} + B + \overline{C} + A\overline{D} + \overline{B}\,\overline{D} \\ &= \overline{A} + B + \overline{C} + \overline{D}\end{aligned}$$

One possible circuit:



Karnaugh map (not covered in lecture)

Combinatorial Logic



Sequence Logic

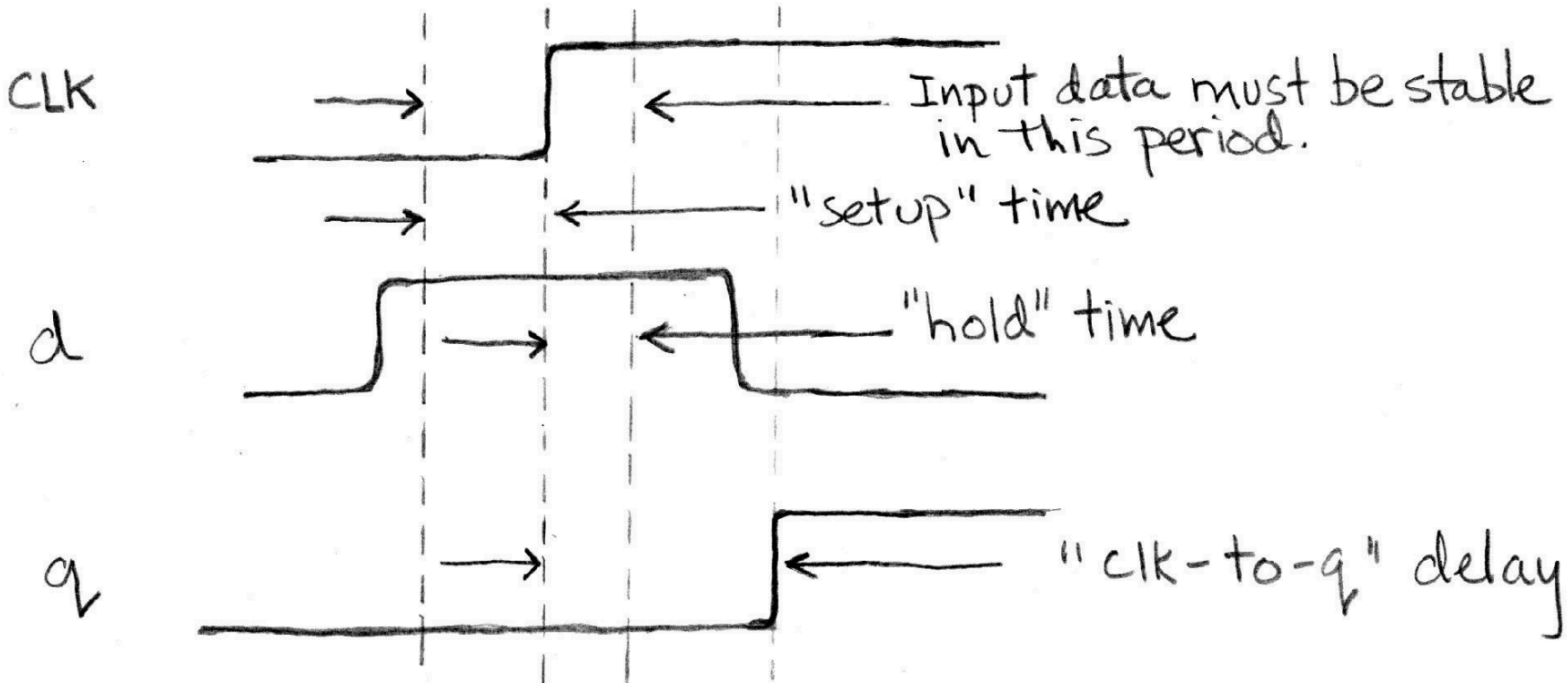
Timing diagram

Sequential Logic

- Sequential Logic can remember their inputs even after the inputs change
- State changes are synchronized by clock.
- Some terminology:
  - Clk-to-q: (t_prop) is the time for a signal to propagate through a flip-flop.
  - Propagation delay: (t_combinational) is the longest delay for any combinational logic (which by definition is surrounded by two flip-flop)
  - Setup time: (t_setup) is the time before the rising clock edge that the input to a flip-flop must be valid.
  - Hold time: The minimum time during which the input must be valid after the clock edge.

Timing diagram

Sequential Logic

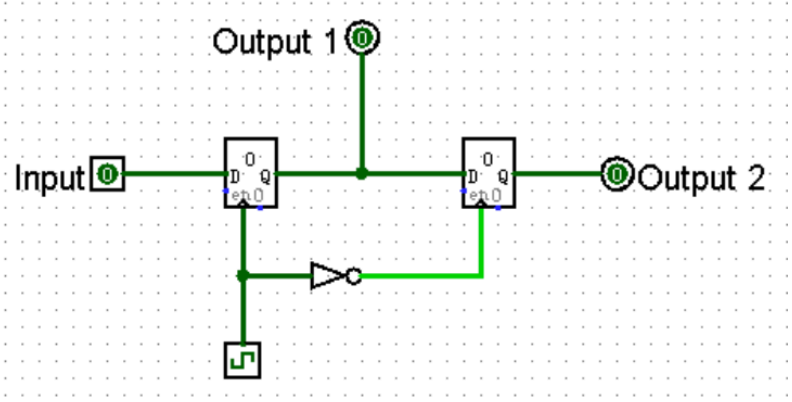


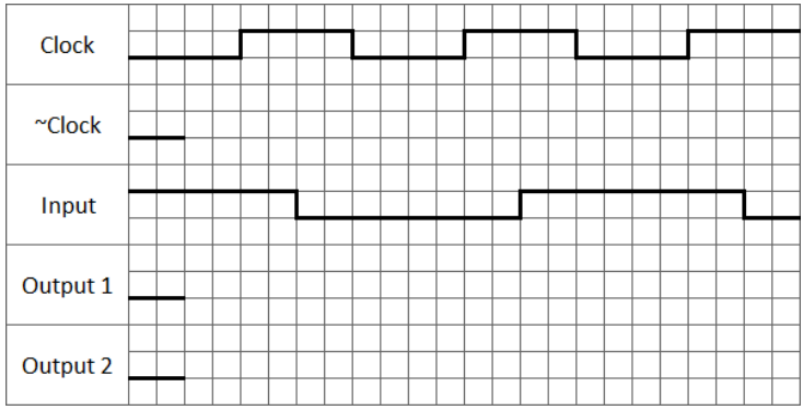
Timing diagram - example

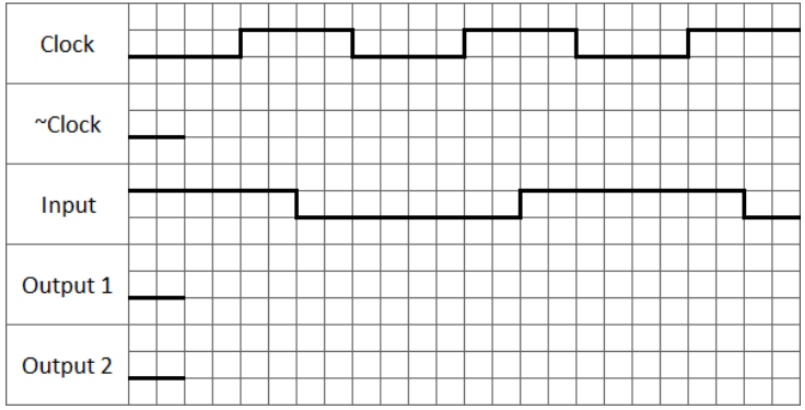
Sequential Logic

10. **SDS**

(a) Draw the Timing Diagram for the circuit below. NOT gates have a 2 ns propagation delay. For each register, the clk-to-q delay is 2 ns and setup time is 2 ns. The clock period is 8 ns, and each grid in the following diagram is a unit of 1 ns. The initial values of clock and output are given in the diagram. Use any of the two empty graphs to put in your answer (so you can re-do it). Clearly **mark your final answer** if you use more than one graph!





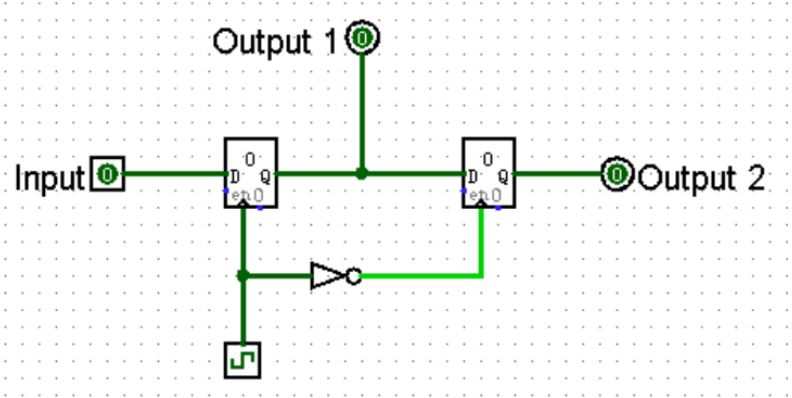


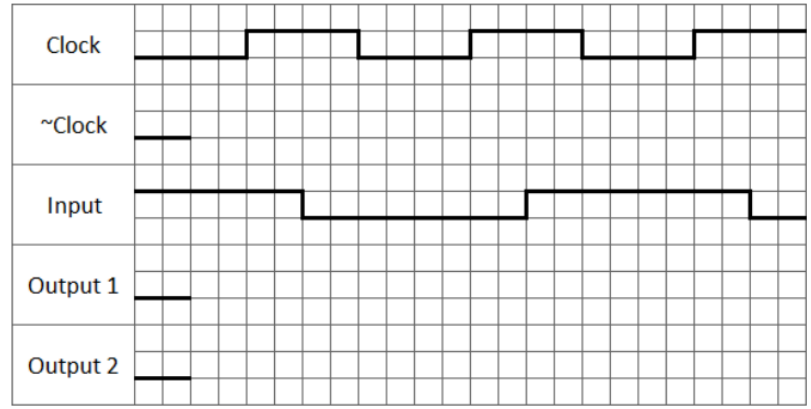
Timing diagram - solution

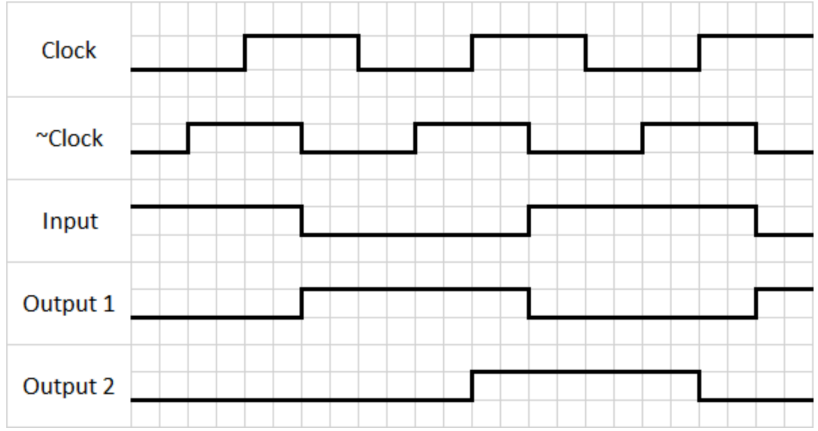
Sequential Logic

10. **SDS**

(a) Draw the Timing Diagram for the circuit below. NOT gates have a 2 ns propagation delay. For each register, the clk-to-q delay is 2 ns and setup time is 2 ns. The clock period is 8 ns, and each grid in the following diagram is a unit of 1 ns. The initial values of clock and output are given in the diagram. Use any of the two empty graphs to put in your answer (so you can re-do it). Clearly **mark your final answer** if you use more than one graph!







Time Calculation

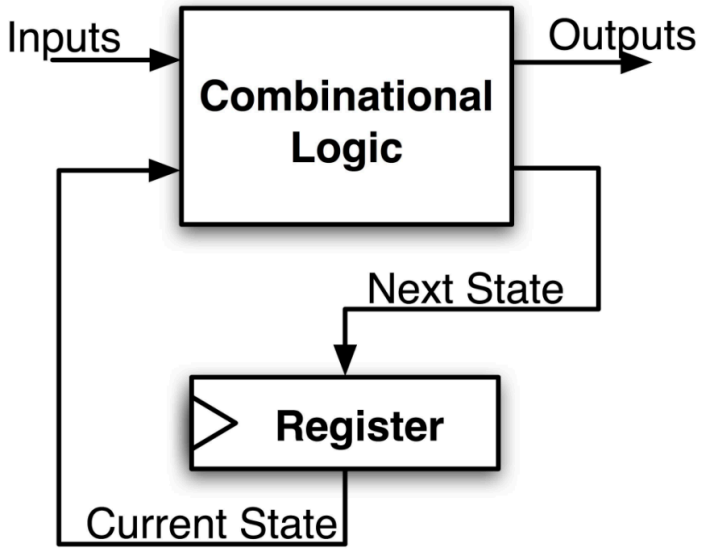
Sequential Logic

- (Setup time violation) In a SDS, a system the clock period (or cycle time) must be at least as large as

$$t_{\text{prop}} + t_{\text{combinational}} + t_{\text{setup}}$$

- (Hold time violation) In a feedback SDS (circuits contain register to register propagation), the hold time must not exceed:

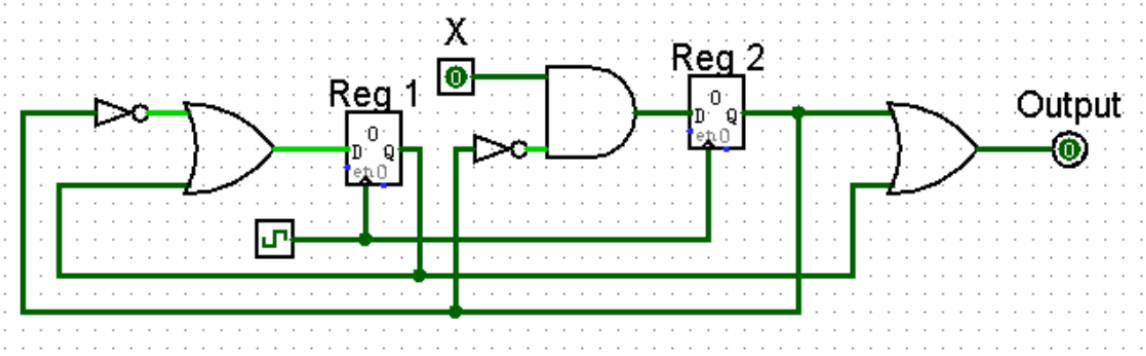
$$t_{\text{prop}} + t_{\text{combinational}}$$



Time Calculation - example

Sequential Logic

Consider the following circuit. Assume the clock has a frequency of 50 MHz, all gates have a propagation delay of 6 ns, X changes 10 ns after the rising edge of clk, Reg1 and Reg2 have a clk-to-q delay of 1 ns.



- What is the longest possible setup time such that there is no setup time violations?
- What is the longest possible hold time such that there is no hold time violations?

Time Calculation - solution

Sequential Logic

Solution:

The clock period is $\frac{1}{50\times10^6}s = 20\ ns$.

Reg1 longest possible setup time: the path is *the output of* $Reg2 \rightarrow NOT \rightarrow OR$, with a delay of 1 ns + 6 ns + 6 ns =13 ns. So 20 - 13 = 7 ns.

Reg2 longest possible setup time: the path is X *changes* $\rightarrow AND$, with a delay of 10 ns + 6 ns = 16 ns. So 20 - 16 = 4 ns.

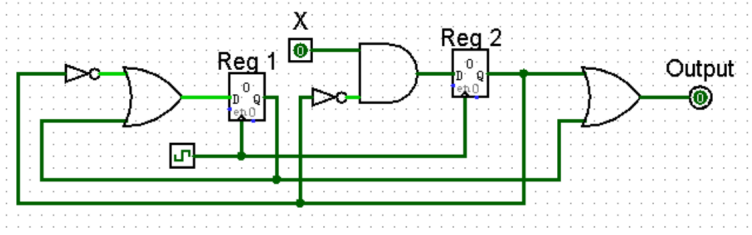
So longest setup time: min(7 ns, 4 ns) = 4 ns.

Reg1 longest possible hold time: the path is *the output of* $Reg1 \rightarrow OR$, with a delay of 1 ns + 6 ns = 7 ns.

Reg2 longest possible hold time: the path is *the output of* $Reg2 \rightarrow NOT \rightarrow AND$, with a delay of 1 ns + 6 ns + 6ns = 13 ns.

So longest hold time: min(7 ns, 13 ns) = 7 ns.

Consider the following circuit. Assume the clock has a frequency of 50 MHz, all gates have a propagation delay of 6 ns, X changes 10 ns after the rising edge of clk, Reg1 and Reg2 have a clk-to-q delay of 1 ns.



FSM

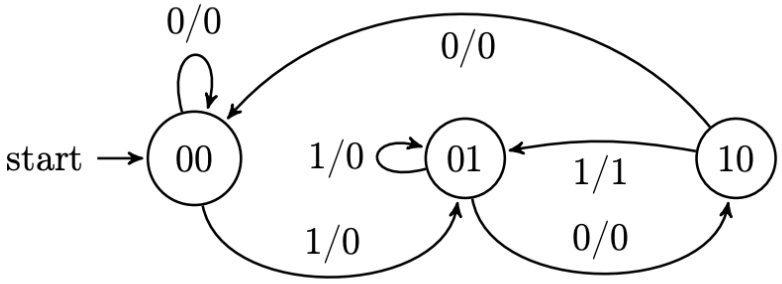
FSM

FSM

- A convenient way to conceptualize computation over time
- Start at a state, given an input, follow some edge to another
- With combinational logic and registers, any FSM can be implemented in hardware
- edge: input/output

Example 1

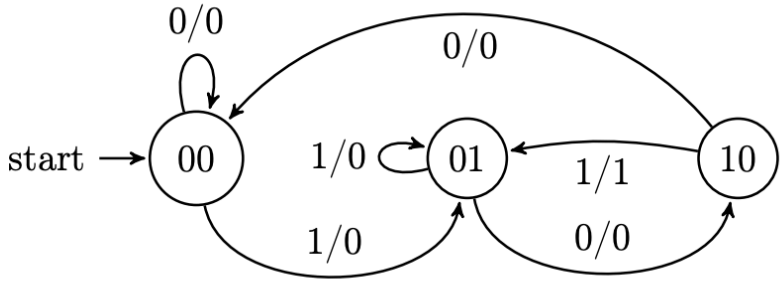
FSM



state bit1	state bit0	input	next state bit1	next state bit0	output
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			

Example 1 - solution

FSM

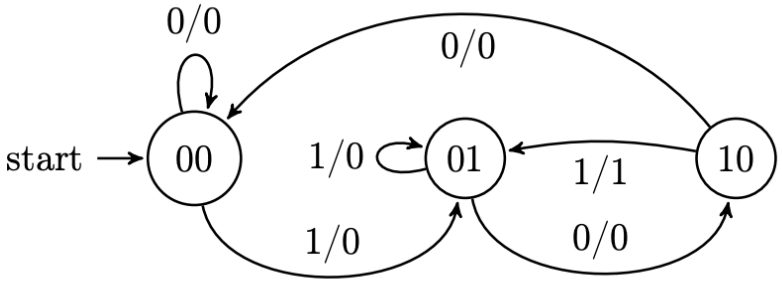


Solution:

state bit1	state bit0	Input	next state bit1	next state bit0	Output
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	0	1	1

Example 2

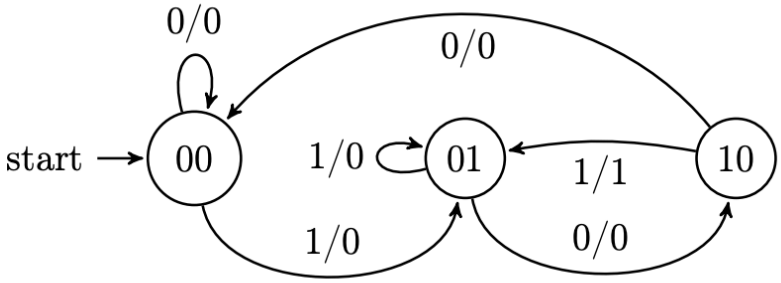
FSM



- Given input: 0100101010
- What is the output? What does this FSM describe (when it outputs 1)?

Example 2 - solution

FSM



- Given input: 0100101010
- What is the output?
- Output: 0000001010
- Whenever the FSM receives 101

Example 3

FSM

- Draw a FSM that outputs 1 when it receives two or more successive ‘0’.

Example 3 - solution

FSM

- Draw a FSM that outputs 1 when it receives two or more successive ‘0’.

Solution: