Digital Logic Systems

Recitation 11: Synchronous Circuits: Design, Simulation and Timing Analysis

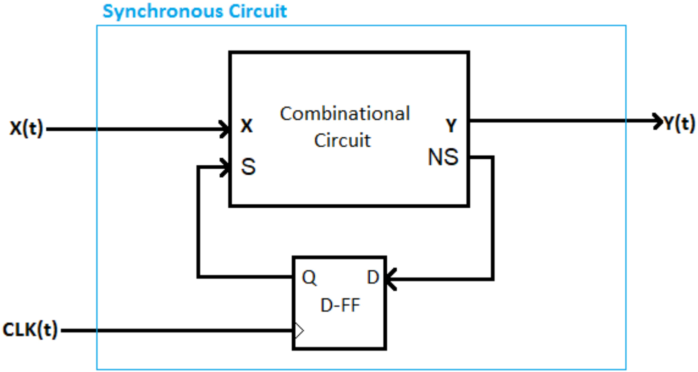
Guy Even   Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

May 27, 2019

Synchronous Circuits - Attributes

1. Consist of combinational circuits and flip-flops (FFs)
2. Can contain cycles, as long as the underlying combinational circuits are still acyclic.
3. Must contain a special CLK input, which is fed to all FFs



Synchronous Circuits - Time

1. We introduce **discrete time**
2. The time is dictated by a very special signal **CLK**$\in\{0,1\}$. This signal is given automatically, you don't have to worry about generating it. Just don't forget to connect it to the required elements (FFs).
3. Each **rising-edge** of CLK advances the time to the next clock cycle.
4. The inputs are now **time-dependent**: No more X, Y, Z. In synchronous circuits we deal with $X(t), Y(t), Z(t)$. The "user" of a synchronous circuit can change these inputs every clock cycle.
5. Don't confuse between the time indices and the string indices.

Example

- $X[i](t)$ is the value of the binary string X at index i at time t.
- $X[2](3)$ is the value of the binary string X at index 2 at time 3.

Synchronous Circuits Design

Step 1 - Functional design
Assume zero-delay and design a functionally correct circuit.

- Option 1: Ad-hoc design - use flip-flops and combinational logic.
- Option 2: Finite-State-Machine (FSM) design & synthesis.

Simulate the design with respect to zero delay model.

Step 2 - Timing Analysis
Remove the zero-delay assumption, don't think of correctness.
Calculate Min-Φ - the minimal clock cycle of your circuit.

The Zero Delay Model

Simplified model for specifying and simulating the functionality of circuits with flip-flops.

1. Transitions of all signals are instantaneous.
2. Combinational gates: $t_{pd} = t_{cont} = 0$.
3. Flip-flops satisfy:

$$t_{su} = t_{i+1} - t_i,$$
$$t_{hold} = t_{cont} = t_{pd} = 0.$$

4. This allows us to specify the functionality of a flip-flop in the zero delay model as follows:

$$Q(t+1) = D(t).$$

Clock enabled flip-flops (zero-delay model)

Definition

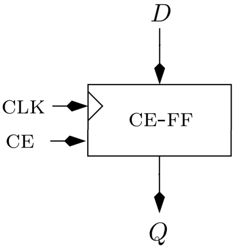
A clock enabled flip-flop is defined as follows.

Inputs: $D(t), \text{CE}(t) \in \{0,1\}$ and a clock CLK.

Output: $Q(t) \in \{0,1\}$.

Functionality:

$$Q(t+1) = \begin{cases} D(t) & \text{if } \text{CE}(t) = 1 \\ Q(t) & \text{if } \text{CE}(t) = 0. \end{cases}$$



Parallel Load Register (zero-delay model)

Definition

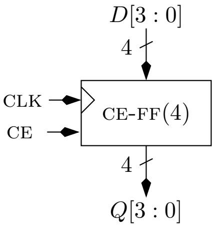
An n-bit *parallel load register* is specified as follows.

Inputs:
- $D[n-1:0](t)$,
- $\text{CE}(t)$, and
- a clock CLK.

Output: $Q[n-1:0](t)$.

Functionality: $$Q[n-1:0](t+1) = \begin{cases} D[n-1:0](t) & \text{if } \text{CE}(t) = 1 \\ Q[n-1:0](t) & \text{if } \text{CE}(t) = 0. \end{cases}$$

An n-bit parallel load register is simply built from n CEFFs.



Shift Register (zero-delay model)

Definition

A *shift register* of n bits is defined as follows.

Inputs: $D[0](t)$ and a clock CLK.

Output: $Q[n-1](t)$.

Functionality: $Q[n-1](t+n) = D[0](t)$.

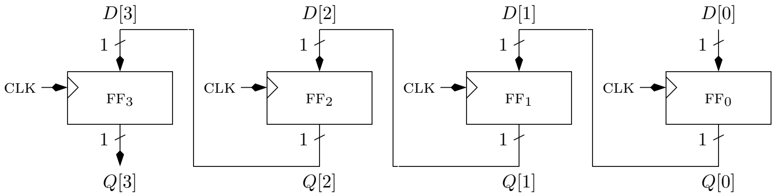
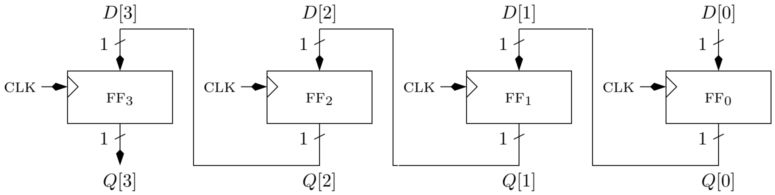


Figure: A 4-bit shift register. Functionality: $Q[3](t+4) = D[0](t)$

Shift Registers - simulation



t	$D[0]$	$Q[3:0]$
0	1	0000
1	1	0001
2	1	0011
3	0	0111
4	1	1110

Algorithm 1 SIM$(C, S_0, \{IN_i\}_{i=0}^{n-1})$ - An algorithm for simulating a synchronous circuit C with respect to an initialization S_0 and a sequence of inputs $\{IN_i\}_{i=0}^{n-1}$.

1. Construct the combinational circuit C' obtained from C by stripping away the flip-flops.
2. For $i = 0$ to $n - 1$ do:
   1. Simulate the combinational circuit C' with input values corresponding to S_i and IN_i. Namely, every input gate in C feeds a value according to IN_i, and every Q-port of a flip-flop feeds a value according to S_i. For every sink z in C', let z_i denote the value fed to z according to this simulation.
   2. For every Q-port S of a flip-flop, define $S_{i+1} \leftarrow NS_i$, where NS denotes the D-port of the flip-flop.

Example: Sequential Adder

Definition

A *sequential adder* is defined as follows.

Inputs: $A(t), B(t), reset(t)$ and a clock signal CLK, where $A(t), B(t), reset(t) \in \{0,1\}$.

Output: $S(t) \in \{0,1\}$.

Functionality: The *reset* signal is an initialization signal that satisfies:

$$reset(t) = \begin{cases} 1 & \text{if } t = 0, \\ 0 & \text{if } t > 0. \end{cases}$$

Then, for every $t \geq 0$,
$\langle A[t:0] \rangle + \langle B[t:0] \rangle = \langle S[t:0] \rangle \pmod{2^{t+1}}$.

Example: Sequential Adder (Ad-hoc design)

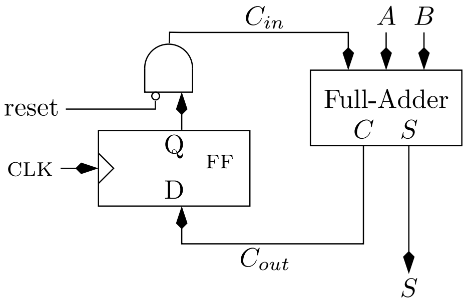
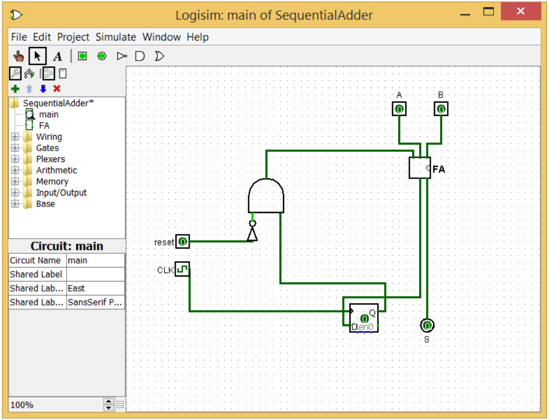


Figure: A synchronous circuit that implements a sequential adder.

Logisim Synchronous Simulation

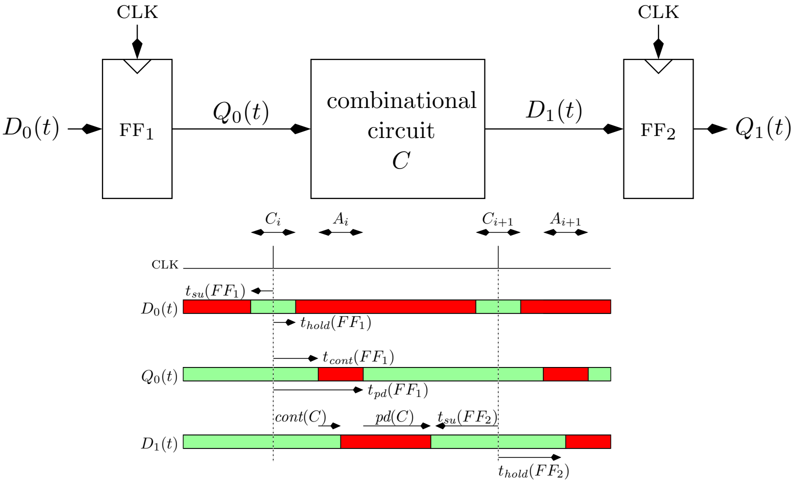
- Use components: "Wiring→Clock" ; "Memory→D Flip-Flop"
- One clock cycle consists of 1 rising edge and 1 falling edge
- One clock cycle = 2 "ticks"
- Logisim respects the Zero-Delay model
- Use "Simulate→Logging" to record the simulation log.



Minimum Clock Period in a Synchronous Circuit

Claim (Minimum Clock Period)

$$\Phi \geq t_{pd}(FF_1) + t_{pd}(C) + t_{su}(FF_2)$$



Ad-hoc design question

Design a synchronous circuit $S(n)$ according to the following specification. Let $n = 2^k$.

Input: n sequential inputs $\{X_i\}_{i=0}^{n-1}$, where each $X_i \in \{0,1\}^k$. Assume that the inputs are valid and stable from clock cycle 0 to clock cycle n.

Output: A single bit Y.

Functionality: The output Y should satisfy in every clock cycle $t \geq n$:

$$Y(t) = \begin{cases} 1 & \text{if all } \{X_i\}_{i=0}^{n-1} \text{ are distinct} \\ 0 & \text{otherwise.} \end{cases}$$

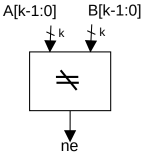
Note that n strings are distinct if no two are equal.

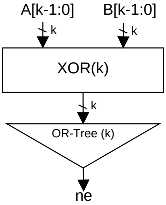
Your design should meet the following goals:

1. Number of flip-flops should be at most $n \cdot k + 1$.
2. The minimum clock period should be $O(k) = O(\log n)$.

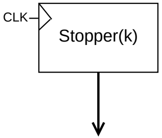
Auxiliary Circuits

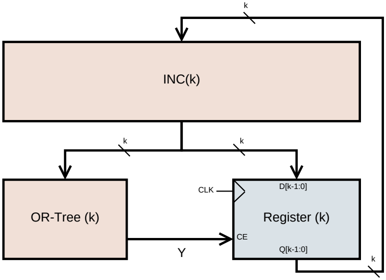
Not-Equal(k) - combinational circuit that outputs $ne = 1$ iff $A \neq B$



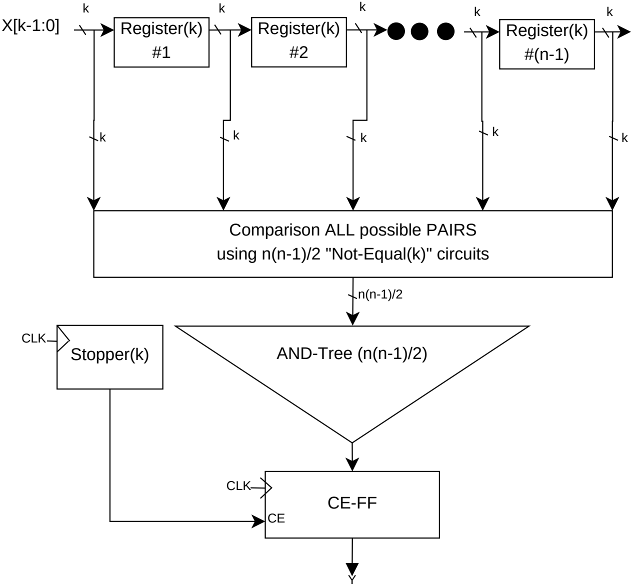


Stopper(k) - synchronous circuit that outputs $Y(t) = 1$ for $t \leq n - 1$ and then $y(t) = 0$ for $t \geq n$.



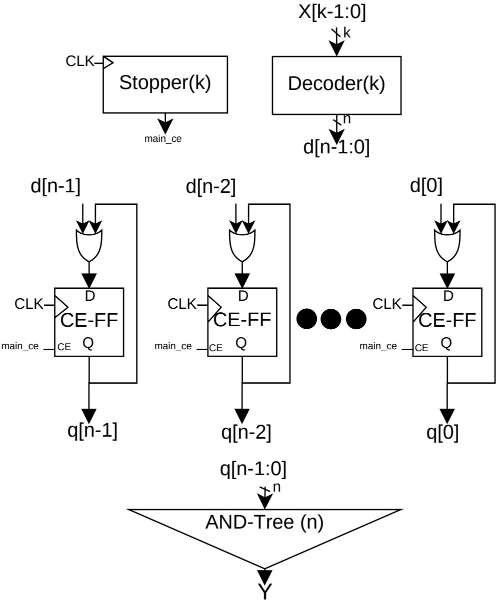


Naive Solution



Using $n \cdot k + 1$ flip-flops. Min-$\Phi = \Theta(log(n))$

Decoder-Based Solution



Using $n + k$ flip-flops. Min-$\Phi = \Theta(log(n))$