Digital Logic Systems

Recitation 12: Synchronous Circuits: Finite State Machines, Analysis and Synthesis. ISA of DLX

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

June 3, 2019

Finite State Machines

Definition

A *finite state machine* (FSM) is a 6-tuple $\mathcal{A} = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$, where

- Q is a set of *states*.
- Σ is the alphabet of the input.
- Δ is the alphabet of the output.
- $\delta : Q \times \Sigma \to Q$ is a *transition function*.
- $\lambda : Q \times \Sigma \to \Delta$ is an *output function*.
- $q_0 \in Q$ is an *initial state*.

The Canonic Form of a Synchronous Circuit

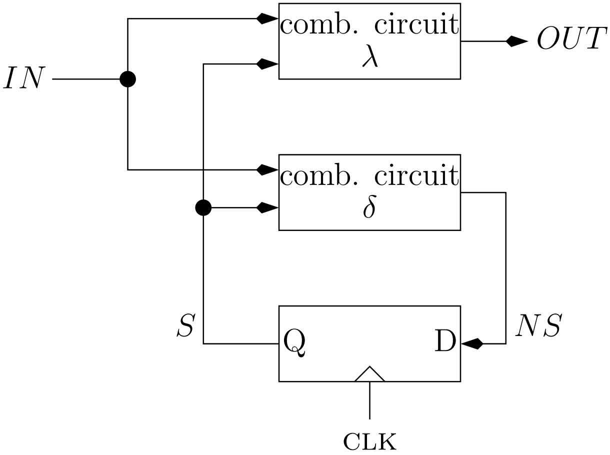


Figure: A synchronous circuit in canonic form.

Synthesis and Analysis

Two tasks are often associated with synchronous circuits:

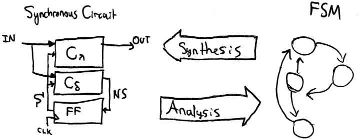
1. **Analysis**: given a synchronous circuit S, describe its functionality by an FSM.
2. **Synthesis**: given an FSM $\mathcal{A}$, design a synchronous circuit S that implements $\mathcal{A}$.

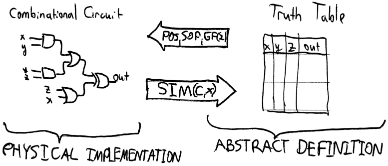
Synthesis and Analysis

Two tasks are often associated with synchronous circuits:

1. **Analysis**: given a synchronous circuit S, describe its functionality by an FSM.
2. **Synthesis**: given an FSM $\mathcal{A}$, design a synchronous circuit S that implements $\mathcal{A}$.

This is somewhat similar to what we already did:





Analysis

The task of analyzing a synchronous circuit S is carried out as follows.

1. Define the FSM $\mathcal{A} = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$ as follows.
   1. The set of states is $Q \triangleq \{0,1\}^k$, where k denotes the number of flip-flops in S.
   2. Define the initial state q_0 to be the initial outputs of the flip-flops.
   3. $\Sigma = \{0,1\}^\ell$, where ℓ denotes the number of input gates in S.
   4. $\Delta = \{0,1\}^r$, where r denotes the number of output gates in S.
   5. Transform S to a functionally equivalent synchronous circuit $\tilde{S}$ in canonic form. Compute the truth tables of the combinational circuits λ and δ. Define the Boolean functions according to these truth tables.

Synthesis

Given an FSM $\mathcal{A} = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$, the task of designing a synchronous circuit S that implements $\mathcal{A}$ is carried out as follows.

1. Encode Q, Σ and Δ by binary strings. Formally, let f, g, h denote one-to-one functions, where

$$
\begin{aligned}
f &: Q \to \{0,1\}^k \\
g &: \Sigma \to \{0,1\}^\ell \\
h &: \Delta \to \{0,1\}^r.
\end{aligned}
$$

2. Design a combinational circuit C_δ that implements the (partial) Boolean function $B_\delta : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^k$ defined by

$$B_\delta(f(x), g(y)) \triangleq f(\delta(x,y)), \text{ for every } (x,y) \in Q \times \Sigma.$$

Synthesis (cont.)

3. Design a combinational circuit C_λ that implements the (partial) Boolean function $B_\lambda : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^r$ defined by

$$B_\lambda(f(x), g(z)) \triangleq f(\lambda(x,z)), \text{ for every } (x,z) \in Q \times \Sigma.$$

4. Let S denote the synchronous circuit in canonic form constructed from k flip-flops and the combinational circuits C_δ for the next state and C_λ for the output.

Synthesis - how to encode the states?

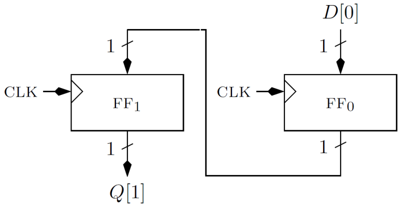
Let S denote the synchronous circuit in canonic form constructed from k flip-flops and the combinational circuits C_δ for the next state and C_λ for the output.

The description of the encoding step leaves a great deal of freedom. Since $|\{0,1\}^k| \geq |Q|$, it follows that $k \geq \log_2 |Q|$, and similar bounds apply to ℓ and r. However, it is not clear that using the smallest lengths is the best idea. Certain encodings lead to more complicated Boolean functions B_δ and B_λ. Thus, the question of selecting a "good" encoding is a very complicated task, and there is no simple solution to this problem.

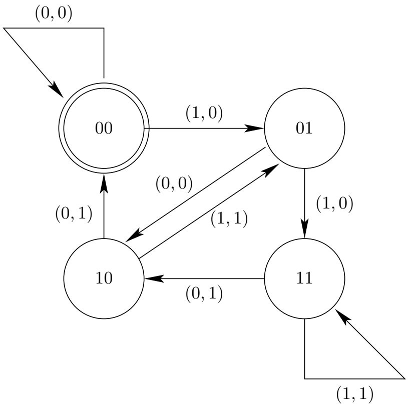
Analysis Question

Analyze the circuit shift-register(2)

i.e. draw a state diagram of the corresponding FSM.



(a) Shift-Reg(2)



(b) FSM(Shift-Reg(2)) Which is an important graph called De Bruijn Graph

Synthesis Question

Design a synchronous circuit S that satisfies the following:

Input: $x(t), y(t) \in \{0,1\}$, for every clock cycle t.

Output: $EQ(t), LT(t), GT(t) \in \{0,1\}$, for every clock cycle t.

Functionality: Let $X_t \triangleq \langle x(t), \ldots, x(0)] \rangle$, $Y_t \triangleq \langle y(t), \ldots, y(0) \rangle$

For every clock cycle $t \geq 0$:

$$EQ(t) = \begin{cases} 1, & \text{if } X_t = Y_t, \\ 0, & \text{otherwise}. \end{cases}$$

$$LT(t) = \begin{cases} 1, & \text{if } X_t < Y_t, \\ 0, & \text{otherwise}. \end{cases}$$

$$GT(t) = \begin{cases} 1, & \text{if } X_t > Y_t, \\ 0, & \text{otherwise}. \end{cases}$$

1. **FSM(S)** - Define the finite state machine $FSM(S) = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$ that satisfies the specification.
2. **Synthesize S** - Implement S by synthesizing $FSM(S)$.

FSM - Understand the required behavior

5	4	3	2	1	0	t	
0	0	1	1	1	0	x(t)	Inputs
0	1	1	0	1	0	y(t)	
14	14	14	6	2	0	$X_t \triangleq \langle x(t), ..., x(0)]\rangle$	Interpretation
26	26	10	2	2	0	$Y_t \triangleq \langle y(t), ..., y(0)\rangle$	
0	0	0	0	1	1	EQ(t)	Outputs
1	1	0	0	0	0	LT(t)	
0	0	1	1	0	0	GT(t)	

- Observation 1: The inputs are streamed from the LSB $\Rightarrow$MSB

FSM - Understand the required behavior

5	4	3	2	1	0	t	
0	0	1	1	1	0	x(t)	Inputs
0	1	1	0	1	0	y(t)	
14	14	14	6	2	0	$X_t \triangleq \langle x(t), \ldots, x(0)] \rangle$	Interpretation
26	26	10	2	2	0	$Y_t \triangleq \langle y(t), \ldots, y(0) \rangle$	
0	0	0	0	1	1	EQ(t)	Outputs
1	1	0	0	0	0	LT(t)	
0	0	1	1	0	0	GT(t)	

- Observation 1: The inputs are streamed from the LSB ⇒MSB
- Observation 2: Outputs at time t depend on inputs at time t.

FSM - Understand the required behavior

5	4	3	2	1	0	t	
0	0	1	1	1	0	x(t)	Inputs
0	1	1	0	1	0	y(t)	
14	14	14	6	2	0	$X_t \triangleq \langle x(t), ..., x(0)]\rangle$	Interpretation
26	26	10	2	2	0	$Y_t \triangleq \langle y(t), ..., y(0)\rangle$	
0	0	0	0	1	1	EQ(t)	Outputs
1	1	0	0	0	0	LT(t)	
0	0	1	1	0	0	GT(t)	

- Observation 1: The inputs are streamed from the LSB ⇒MSB
- Observation 2: Outputs at time t depend on inputs at time t.
- What will our circuit need to "memorize" to be able to determine the output at time t?

FSM - Understand the required behavior

5	4	3	2	1	0	t	
0	0	1	1	1	0	x(t)	Inputs
0	1	1	0	1	0	y(t)	
14	14	14	6	2	0	$X_t \triangleq \langle x(t), ..., x(0)]\rangle$	Interpretation
26	26	10	2	2	0	$Y_t \triangleq \langle y(t), ..., y(0)\rangle$	
0	0	0	0	1	1	EQ(t)	Outputs
1	1	0	0	0	0	LT(t)	
0	0	1	1	0	0	GT(t)	

- Observation 1: The inputs are streamed from the LSB $\Rightarrow$MSB
- Observation 2: Outputs at time t depend on inputs at time t.
- What will our circuit need to “memorize” to be able to determine the output at time t?
  - All the input bits from time 0? Namely $\{x(\tau), y(\tau)\}_{\tau=0}^{t-1}$

FSM - Understand the required behavior

5	4	3	2	1	0	t	
0	0	1	1	1	0	x(t)	Inputs
0	1	1	0	1	0	y(t)	
14	14	14	6	2	0	$X_t \triangleq \langle x(t), \ldots, x(0)] \rangle$	Interpretation
26	26	10	2	2	0	$Y_t \triangleq \langle y(t), \ldots, y(0) \rangle$	
0	0	0	0	1	1	EQ(t)	Outputs
1	1	0	0	0	0	LT(t)	
0	0	1	1	0	0	GT(t)	

- Observation 1: The inputs are streamed from the LSB ⇒MSB
- Observation 2: Outputs at time t depend on inputs at time t.
- What will our circuit need to "memorize" to be able to determine the output at time t?
  - All the input bits from time 0? Namely $\{x(\tau), y(\tau)\}_{\tau=0}^{t-1}$
    Bad, the required information grows with time

FSM - Understand the required behavior

5	4	3	2	1	0	t	
0	0	1	1	1	0	x(t)	Inputs
0	1	1	0	1	0	y(t)	
14	14	14	6	2	0	$X_t \triangleq \langle x(t), \ldots, x(0)] \rangle$	Interpretation
26	26	10	2	2	0	$Y_t \triangleq \langle y(t), \ldots, y(0) \rangle$	
0	0	0	0	1	1	EQ(t)	Outputs
1	1	0	0	0	0	LT(t)	
0	0	1	1	0	0	GT(t)	

- Observation 1: The inputs are streamed from the LSB ⇒MSB
- Observation 2: Outputs at time t depend on inputs at time t.
- What will our circuit need to "memorize" to be able to determine the output at time t?
  - All the input bits from time 0? Namely $\{x(\tau), y(\tau)\}_{\tau=0}^{t-1}$
    Bad, the required information grows with time
  - Just the relation between X_{t-1} and Y_{t-1}.

FSM - Understand the required behavior

5	4	3	2	1	0	t	
0	0	1	1	1	0	x(t)	Inputs
0	1	1	0	1	0	y(t)	
14	14	14	6	2	0	$X_t \triangleq \langle x(t), \ldots, x(0)]\rangle$	Interpretation
26	26	10	2	2	0	$Y_t \triangleq \langle y(t), \ldots, y(0)\rangle$	
0	0	0	0	1	1	EQ(t)	Outputs
1	1	0	0	0	0	LT(t)	
0	0	1	1	0	0	GT(t)	

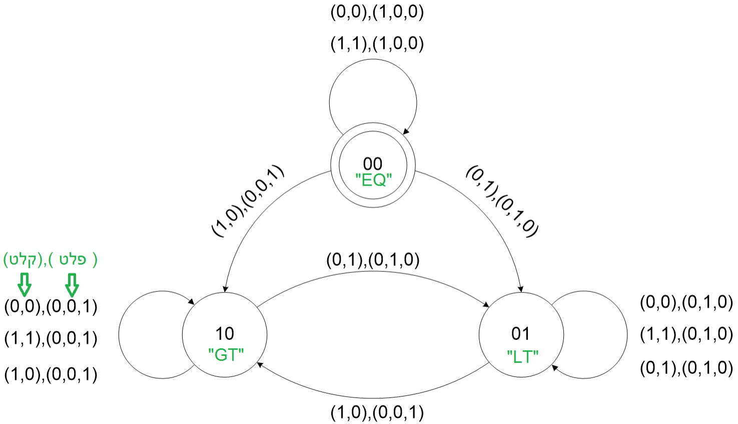
- Observation 1: The inputs are streamed from the LSB $\Rightarrow$MSB
- Observation 2: Outputs at time t depend on inputs at time t.
- What will our circuit need to "memorize" to be able to determine the output at time t?
  - All the input bits from time 0? Namely $\{x(\tau), y(\tau)\}_{\tau=0}^{t-1}$
    Bad, the required information grows with time
  - Just the relation between X_{t-1} and Y_{t-1}.
    Good, only 3 options "equal" "less than" "greater than"

FSM - Declare the states and the encodings

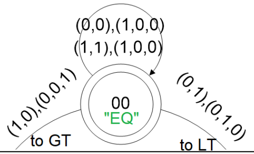
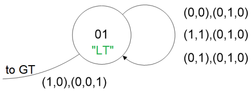
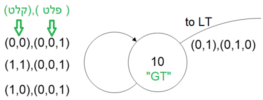
- Let's Describe the $FSM(S) = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$ and also provide the encodings:
  - $Q \triangleq \{0,1\}^2$ - the states
    00 for $X_t = Y_t$ (The "EQ" state)
    01 for $X_t < Y_t$ (The "LT" state)
    10 for $X_t > Y_t$ (The "GT" state)
  - $\Sigma \triangleq \{0,1\}^2$ - the input alphabet.
    The two inputs $x(t), y(t)$ are simply concatenated together.
  - $\Delta \triangleq \{0,1\}^3$ - the output alphabet
    100 for EQ(t)
    010 for LT(t)
    001 for GT(t)
  - q_0 is 00, (The equality state)

FSM - State Diagram

- Let's Describe the $FSM(S) = \langle Q, \Sigma, \Delta, \delta, \lambda, q_0 \rangle$
- δ - state-transfer function, receives the current input and the current state and returns the next state.
- λ - output-function, receives the current input and the current state and returns the current output.
- We describe the two functions using a state diagram:

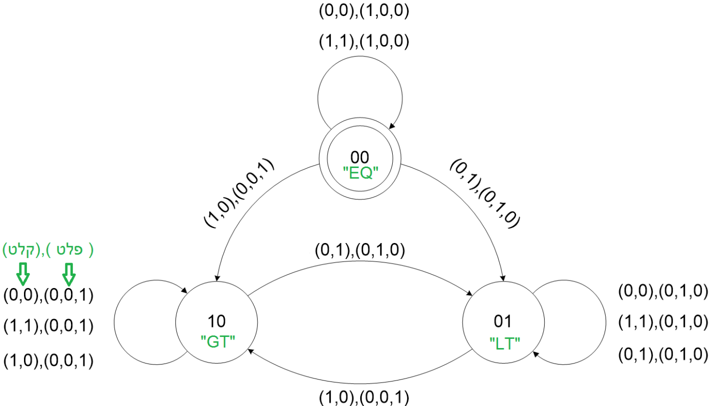


Translate each state into 4 rows of the table.
Note that $S[1:0] = 11$ is an inaccessible state.

Current State	S[1]	S[0]	x	y	NS[1]	NS[0]	EQ	LT	GT
	0	0	0	0	0	0	1	0	0
	0	0	0	1	0	1	0	1	0
	0	0	1	0	1	0	0	0	1
	0	0	1	1	0	0	1	0	0
	0	1	0	0	0	1	0	1	0
	0	1	0	1	0	1	0	1	0
	0	1	1	0	1	0	0	0	1
	0	1	1	1	0	1	0	1	0
	1	0	0	0	1	0	0	0	1
	1	0	0	1	0	1	0	1	0
	1	0	1	0	1	0	0	0	1
	1	0	1	1	1	0	0	0	1

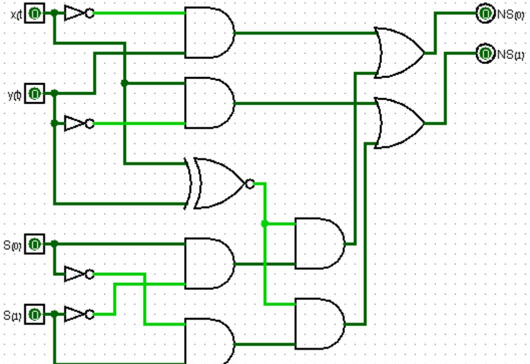
Synthesis - Implementing C_δ (skipping truth table)

- The C_δ - state-transfer circuit receives the $IN = x(t) \circ y(t)$ and the current state S and outputs the next state NS.
- We can derive the C_δ formulas directly from the FSM(S):
  $NS = 00 \Leftrightarrow S = 00$ and $x(t) = y(t)$
  $NS = 01 \Leftrightarrow x(t) < y(t)$ or $(S = 01$ and $x(t) = y(t))$
  $NS = 10 \Leftrightarrow x(t) > y(t)$ or $(S = 10$ and $x(t) = y(t))$



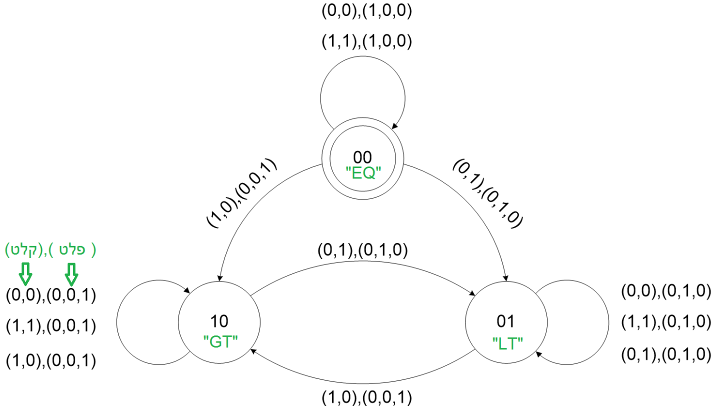
Synthesis - Implementing C_δ

- We can derive the C_δ formulas from the FSM(S):
  $NS = 00 \Leftrightarrow S = 00$ and $x(t) = y(t)$
  $NS = 01 \Leftrightarrow x(t) < y(t)$ or $(S = 01$ and $x(t) = y(t))$
  $NS = 10 \Leftrightarrow x(t) > y(t)$ or $(S = 10$ and $x(t) = y(t))$
- We further provide the boolean function per each flip-flop:
  $NS[0] = \bar{x} \cdot y + (\overline{S[1]} \cdot S[0] \cdot NXOR(x, y))$
  $NS[1] = x \cdot \bar{y} + (S[1] \cdot \overline{S[0]} \cdot NXOR(x, y))$



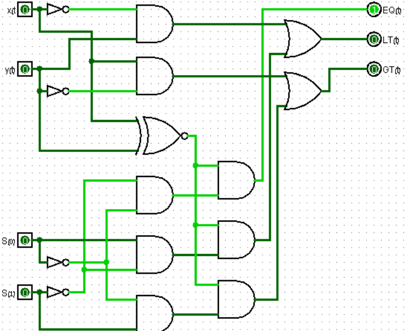
Synthesis - Implementing C_λ (skipping truth table)

- The C_λ - output circuit receives the $IN = x(t) \circ y(t)$ and the current state S and generates outputs EQ(t),LT(t),GT(t).
- We can derive the C_λ formulas directly from the FSM(S):
  $EQ = 1 \Leftrightarrow S = 00 \text{ and } x(t) = y(t)$
  $LT = 1 \Leftrightarrow x(t) < y(t) \text{ or } (S = 01 \text{ and } x(t) = y(t))$
  $GT = 1 \Leftrightarrow x(t) > y(t) \text{ or } (S = 10 \text{ and } x(t) = y(t))$



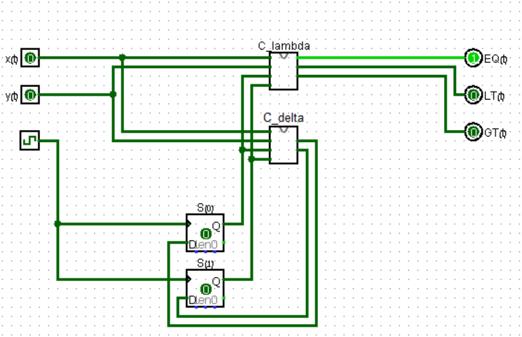
Synthesis - Implementing C_λ

- We can derive the C_λ formulas from the FSM(S):
  $EQ = 1 \Leftrightarrow S = 00 \text{ and } x(t) = y(t)$
  $LT = 1 \Leftrightarrow x(t) < y(t) \text{ or } (S = 01 \text{ and } x(t) = y(t))$
  $GT = 1 \Leftrightarrow x(t) > y(t) \text{ or } (S = 10 \text{ and } x(t) = y(t))$
- We further provide the boolean function per each flip-flop:
  $EQ = \overline{S[0]} \cdot \overline{S[1]} \cdot NXOR(x, y)$
  $LT = \bar{x} \cdot y + (\overline{S[1]} \cdot S[0] \cdot NXOR(x, y))$
  $GT = x \cdot \bar{y} + (S[1] \cdot \overline{S[0]} \cdot NXOR(x, y))$



Synthesis - Putting it all together into S

- Instantiate the C_δ and the C_λ circuits.
- Add k flip-flops for the state storage.



RESAb3 simulator

RESAb3 software contains the **text editor** for writing the assembly code, and the **DLX simulator** for running it.

1. Open the Resa program and choose a text editor.
2. Write a program in a text editor, and compile it to ensure that it has no compilation errors. This should generate the ".cod" file with the machine code.
3. File→New to open simulator window.
4. In the simulator window File→Open COD to load the machine code.
5. Simulator→Add Watch to observe a particular memory cell.
6. Run→Run Until Halt to execute the whole program