Chapter 5

SEQUENTIAL CIRCUITS:

Small Designs

In this Chapter

- Sequential circuit design models
- Design examples:
  - registers, counters, sequence recognizer
- Sequential circuit timing
- Interfacing sequential circuits

Sequential Circuit as a Finite State Machine (FSM)

- **Requires flip-flop(s) to save circuit state**
- **There are three types of FSMs.**
  - Mealy State Machine
  - Moore State Machine
  - Hybrid
- **Requires combinational circuit(s) to generate next circuit state and output(s)**
- **Two types of circuits:**
  - **Functions of the circuits cannot be determined in advance**
    - **FSM design is modeled with a finite state diagram (FSD)**
  - **Functions of the circuits can be determined in advance**
    - **E.g., MUX, adder, ALU**
    - **No need for an FSD**

A simple design Example (Parallel-load register with enable)

- Assume unknown combinational circuit
  1. Design 1-bit register 1st
     - We have seen D flip-flop with enable in Ch4
     - Here, the flip-flop formally modeled as FSD (next slide)
  2. Combine register slices to create 4-bit register below
     - *enable*, *clk*, *_reset* connect to all

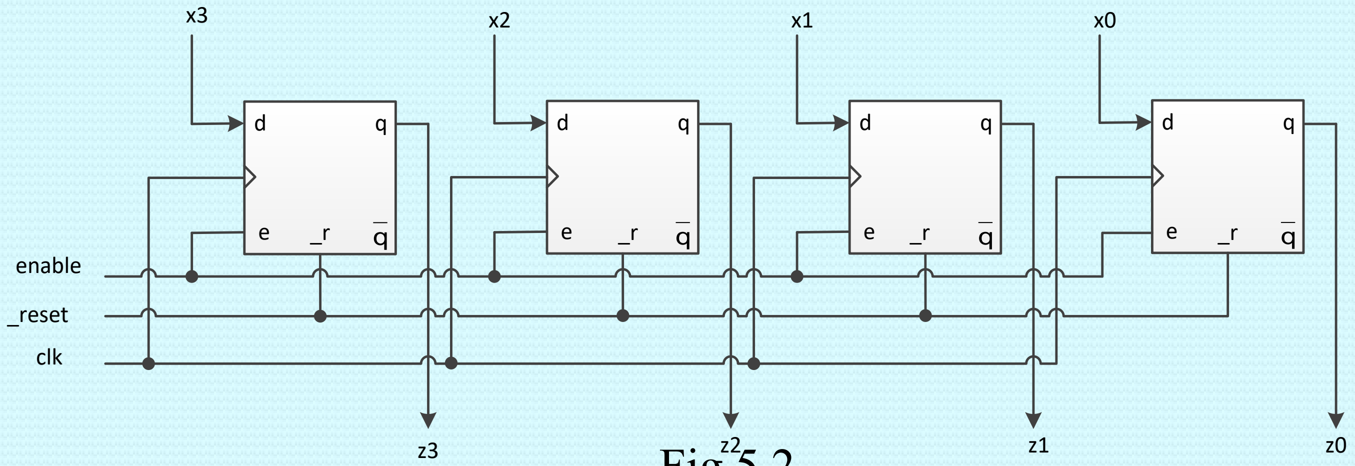


Fig 5.2

FSM design steps (1-bit parallel-load register example)

1. Draw FSD and detailed block diagram of 1-bit register slice

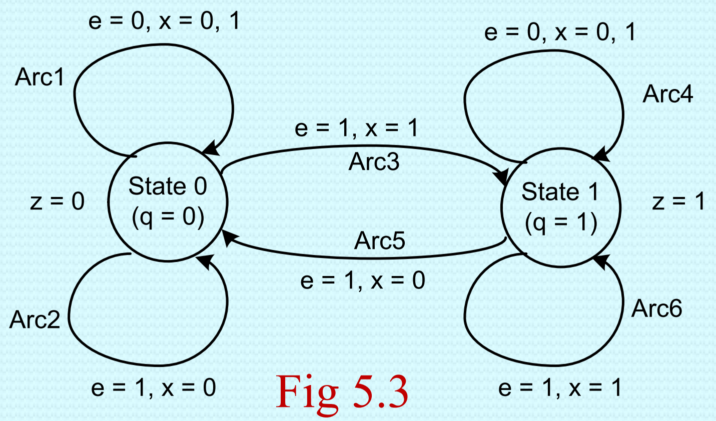


Fig 5.3

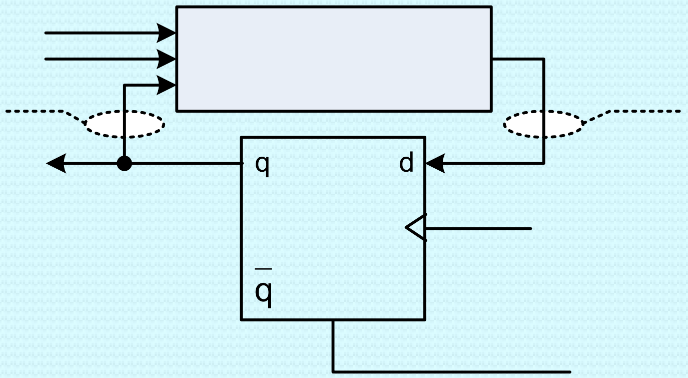


Fig 5.4

2. Convert FSD to truth table (also called transition table)

Current State	External Inputs		Next State
q	*E*	*x*	*d*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Table 5.1

4. Draw final circuit

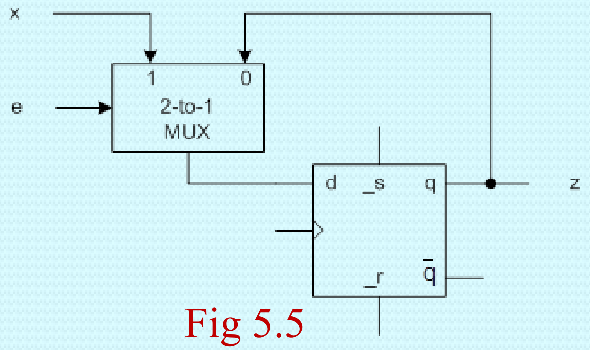


Fig 5.5

3. Find minimal SOP or POS expression(s) for the output(s)

$d = \bar{e}q + ex$ Ah, a MUX!

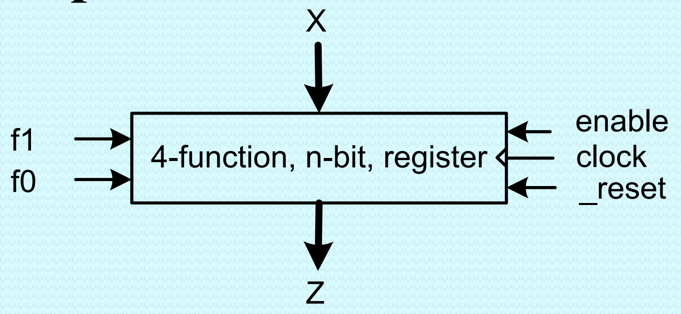
Circuit shown 1st in Chap. 4.

Some design rules

- If cannot determine function(s) of combinational circuit(s) in advance:
  1. Model FSM as FSD
     - May need to design bit-slice 1st
  2. Determine number of flip flops
  3. Convert the FSD to truth table
  4. Find minimal expressions for next state variable(s) and output(s)
  5. Draw the complete circuit with flip-flops
- Otherwise
  - Use bit-serial design with known modules
  - Or, bit-parallel design with known modules

Multi-Function Register

- Performs one of many functions
- Can be designed bit-serial or bit-parallel
- If cannot determine function(s) of combinational circuit(s) in advance
    - Model a bit-slice (e.g., 1-bit) as FSD
- Otherwise: Oh, I can use a MUX
    - Bit-serial: Use 1-bit 4-to-1 MUXs, for example
    - Bit-parallel: Use n-bit 4-to-1 MUX



f1	f0	Action
0	0	Clear (synchronous reset)
0	1	Load X
1	0	Arithmetic shift (shifts right repeating the sign bit)
1	1	Right shift (shifts right entering 0 from left)

Fig 5.6

FSM (A formal view)

Three types of FSMs:

- **Moore**
  - External inputs do not synchronously affect outputs
  - Outputs called Moore
- **Mealy**
  - External inputs synchronously affect outputs
  - Outputs called Mealy
- **Hybrid**
  - Generates both Moore and Mealy outputs

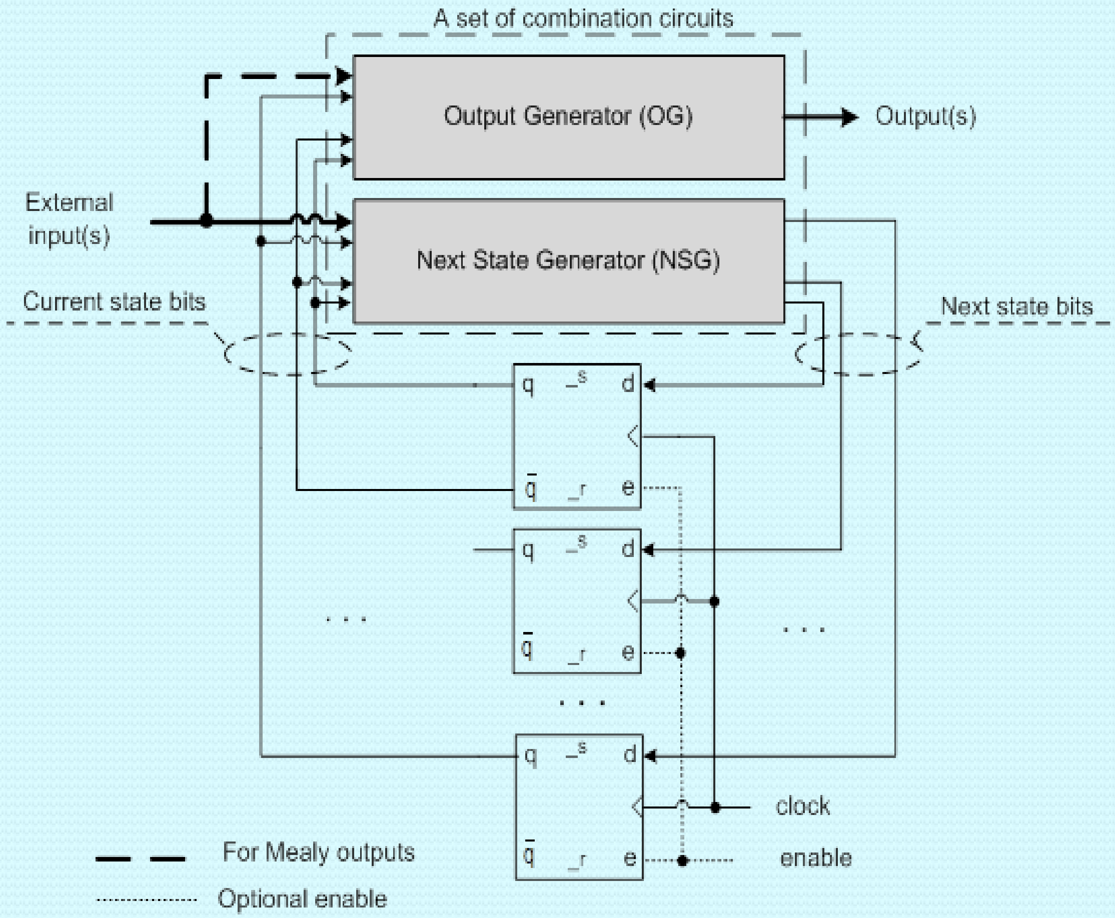
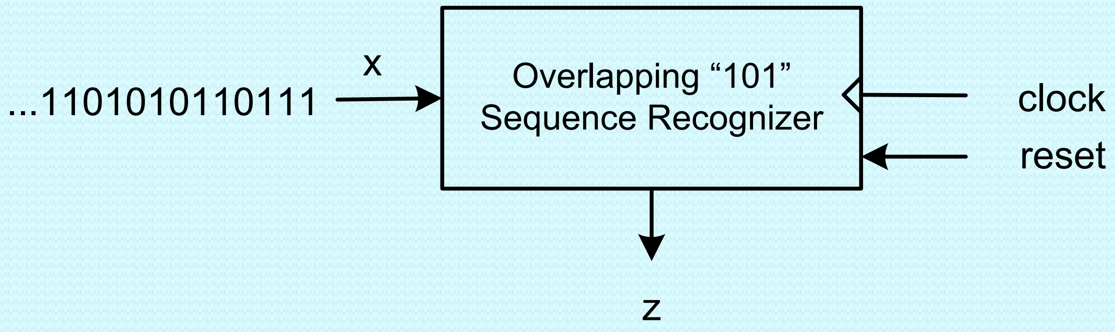


Fig 5.6 Fig 5.12

Sequence Recognizer

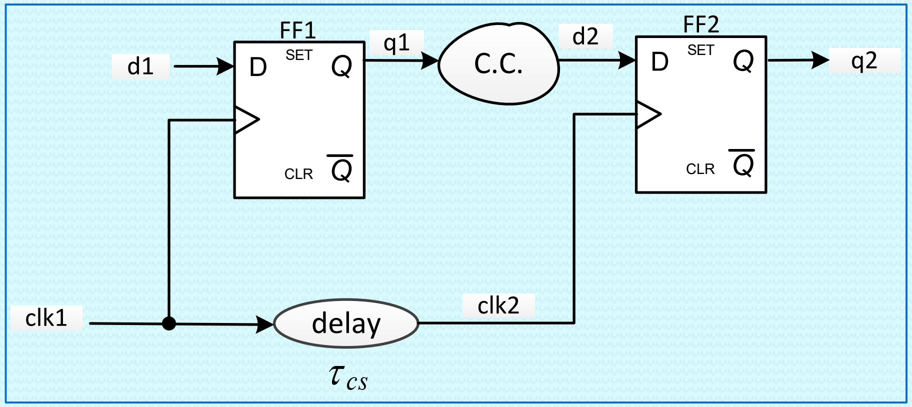
Suppose FSM recognizes overlapping input sequence "101":

- **Inputs processed one bit at a time (one per clock cycle)**
- **Not possible to determine the functions of the combinational circuits in advance**
  - **Model FSM as FSD**
- **Reset initializes the machine to known state**
- **Can be designed either as Moore or Mealy machine**



Sequential Circuit Timing (1)

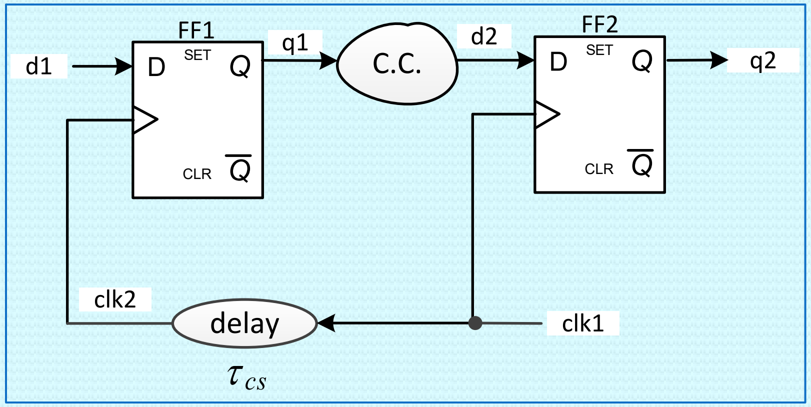
- Clock skew due to delay (τ_{cs}) in clock signal
- Clock skew can create circuit timing problems
  1. If Δcc < τ_{cs}, FF2 will load $d2^{\text{New}}$ and not $d2^{\text{Current}}$
     - FSM results in functional error
  2. If Δcc ~= τ_{cs}, *d2* may be still changing when *clk2* arrives
     - FSM malfunction due to setup or hold time violation at FF2
- Δcc > τ_{cs}, FF2 will load $d2^{\text{Current}}$ as it should
  - Circuit will function correctly



clk1 reaches FF1 1st and after some delay as *clk2* to FF2

Sequential Circuit Timing (2)

- **Possible problem: Next *clk1* reaches FF2 before previous *clk2* reaches FF1**
  - **If Δcc ~= $\tau - \tau_{cs}$, *d2* may be still changing when *clk1* arrives**
    - **FSM malfunction due to setup or hold time violation at FF2**
- **Normal operation**
  - **When Δcc < $\tau - \tau_{cs}$**



***clk1* reaches FF2 1st and after some delay as *clk2* to FF1**

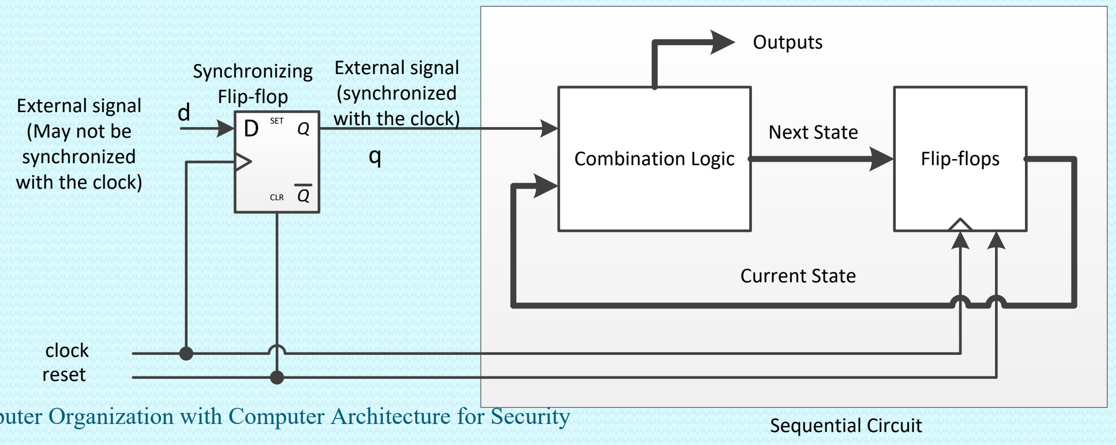
Clock Frequency Estimation – With Clock Skew

Add τ_{cs} to the clock period determined in Ch. 4

$$\boldsymbol{\tau \geq \tau_{cq-max} + \tau_{pd-max} + \tau_{st} + \tau_{cs}}$$

Interfacing Sequential Circuits

- How to handle asynchronous inputs
  - Asynchronous inputs may change at any time with respect to *clock* signal
    - They can cause setup or hold time violation if directly enter FSM
- Solution: Use synchronization flip-flop(s)
  - Asynchronous input *d* may violate setup or hold time of synchronizing FF, not FFs of FSM
    - *q* of synchronizing FF will eventually stabilize if setup or hold time violated
  - May use two synchronizing FFs if *q* slow to stabilize



External signal (May not be synchronized with the clock)
d
Synchronizing Flip-flop
D
SET
Q
CLR
Q̄
External signal (synchronized with the clock)
q
Outputs
Combination Logic
Next State
Flip-flops
Current State
clock
reset
Sequential Circuit