CSE140: ALU: Arithmetic Logic Unit

Prof. Tajana S Rosing
A One Bit ALU

- This 1-bit ALU performs AND, OR, and ADD

```
+  1 1
  1 1 1 0
  1 0 1 1
  1 0 1 0

0 0 0 0  - 4
0 0 0 0  - 2
0 0 0 0  - 6
```

Operation: \( 2 \) ADD

Result: \( = 10 \)

Sources: TSR, Katz, Boriello & Vahid
what signals accomplish ADD?

<table>
<thead>
<tr>
<th>Binvert</th>
<th>CiIn</th>
<th>Oper</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>NONE OF THE ABOVE</td>
<td></td>
</tr>
</tbody>
</table>

\[ A + B \quad C_{in} = \phi \]

2's compl. \[ \overline{B} + 1 \]

\[ C_{in} = 1 \]
Full 32-bit ALU

what signals accomplish OR?

<table>
<thead>
<tr>
<th>Binvert</th>
<th>CIn</th>
<th>Oper</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>NONE OF THE ABOVE</td>
<td></td>
</tr>
</tbody>
</table>

OR \( A \land \overline{B} \\ \overline{B} = 1 \)

\[ \text{Oper} = 1 \]

Sources: TSR, Katz, Boriello & Vahid
Full 32-bit ALU

Sources: TSR, Katz, Boriello & Vahid

what signals accomplish SUB?

<table>
<thead>
<tr>
<th>Binvert</th>
<th>ClIn</th>
<th>Oper</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>NONE OF THE ABOVE</td>
<td></td>
</tr>
</tbody>
</table>

Binv = 1

single bit slice

\[ \overline{B} + 1 \]

\[ 2\text{S comp.} \]
Arithmetic Logic Unit – Example 2

F₂:0 \rightarrow F₂ F₁ F₀

<table>
<thead>
<tr>
<th>F₂:0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>A &amp; B</td>
</tr>
<tr>
<td>001</td>
<td>A</td>
</tr>
<tr>
<td>010</td>
<td>A + B</td>
</tr>
<tr>
<td>011</td>
<td>Not used</td>
</tr>
<tr>
<td>100</td>
<td>A &amp; \sim B</td>
</tr>
<tr>
<td>101</td>
<td>A</td>
</tr>
<tr>
<td>110</td>
<td>A - B</td>
</tr>
<tr>
<td>111</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Arithmetic Logic Unit

Example 2

Sources: TSR, Katz, Boriello & Vahid
ALU Design Example 3

### Truth Table

<table>
<thead>
<tr>
<th>S2</th>
<th>S1</th>
<th>S0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>B-A</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>A-B</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>A+B</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>A xor B</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>A or B</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>A and B</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello & Vahid
What is a sequential circuit?

A circuit whose output depends on current inputs and past outputs

A circuit with memory

\[ y_i = f_i(S^t, X) \]
\[ s_{i+1} = g_i(S^t, X) \]
Why do we need circuits with ‘memory’?

- Circuits with memory can be used to store data
- Systems have circuits that run a sequence of tasks

**Memory Hierarchy**

- Registers
- Cache
- Main Memory
- Hard disk

Sources: TSR, Katz, Boriello & Vahid
Simplest memory element

SRAM
static RAM

"remember"

"load"

"data"

"stored value"

6T

2-transistor

sense amp

sense amp

Sources: TSR, Katz, Boriello, & Vahid
Flight attendant call button

- Flight attendant call button
  - Press call: light turns on
    - **Stays on** after button released
  - Press cancel: light turns off
  - Logic circuit to implement this?

- SR latch implementation
  - Call=1 : sets Q to 1 and keeps it at 1
  - Cancel=1 : resets Q to 0
- $S = 1, R = 0$: 
  then $Q = 1$ and $\overline{Q} = 0$

- $S = 0, R = 1$: 
  then $Q = 1$ and $\overline{Q} = 0$
SR Latch Analysis

– \( S = 0, R = 0 \):
  then \( Q = Q_{\text{prev}} \)
  – Memory!

– \( S = 1, R = 1 \):
  then \( Q = 0, \bar{Q} = 0 \)
  – Invalid State
  \( \bar{Q} \neq \text{NOT} \, Q \)

Sources: TSR, Katz, Boriello & Vahid
What if a kid presses both call and cancel & then releases them?

- If S=1 and R=1 at the same time and then released, Q=?
  - Can also occur also due to different delays of different paths
  - Q may oscillate and eventually settle to 1 or 0 due to diff. path delay

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>not allowed</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello & Vahid
CSE140: Components and Design Techniques for Digital Systems

Latches and Flip-Flops

Tajana Simunic Rosing
## CSE Research Open House

### Event Schedule

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:00 AM</td>
<td>Check-in and Continental Breakfast for Registered Attendees</td>
<td>CSE Lobby</td>
</tr>
<tr>
<td>9:00 AM</td>
<td>Welcoming Ceremony and Keynote by Edward Wu, CSE '04</td>
<td>CSE 1242</td>
</tr>
</tbody>
</table>
| 10:15 AM | **Session 1**                                                        | **Computer Vision and Robotics**  
CSE 1202 | **Sustainable Computing**  
CSE 1242 | **Security and Programming Languages**  
CSE 2154 |
| 11:25 AM | **Session 2**                                                        | **Architecture**  
CSE 1202 | **Machine Learning and AI**  
CSE 1242 | **Bioinformatics**  
CSE 2154 |
| 12:30 PM | Lunch at Calit2 Courtyard for all Registered Attendees               |                           |
| 1:30 PM  | **Session 3**                                                        | **Moderators:**  
Ryne Chaloux  
Senior Software Engineer and Tech Lead  
AppFolio  
Alessandro Muti  
VP and Fellow Software Engineer  
WeWork      | **Parand Darugar**  
VP Services Architecture  
Coupa      | **Tajana Rosing**  
CSE Professor  
Panel Moderator |
| 2:30 PM  | Open Labs and Grad Student Poster Session                           | 3rd Floor Lobby, 4th Floor Lobby. Labs on all floors |
| 4:00 PM  | Industry Info Sessions                                               | Front of CSE Building, CSE 1202 |
| 5:30 PM - 6:00 PM | Alumni Awards and Poster Awards                                   | CSE 1242 |

Sources: TSR, Katz, Boriello & Vahid
Industry Panel

Moderator: Tajana Rosing, Professor, CSE

Ryne Caloux ’15
Sr. Software Engineer and Tech Lead, AppFolio

Parand Darugar ’93
VP Services Architecture, Coupa

Alessandro Muti
VP and Fellow Software Engineer, WeWork

cseopenhouse.ucsd.edu

Sources: TSR, Katz, Boriello & Vahid
Panel Questions

• Please provide our audience with a brief overview of your company, as well as your role within the company.
• What do you like the most about your current role and company?
• What kind of research/development is your company currently working on?
• What kind of technology excites you the most?
• How can industry and academia collaborate better towards common goals?
• What motivated you to get into research?
• Have you have been involved in tech transfer and what were the results?
• Do you have any advice for our students as they prepare for their careers in computer science and engineering?
• How important is a background in research when seeking candidates at your company? Further, what kinds of career paths and/or areas for growth are open to applicants with strong research skills at your company?
• In an ideal situation, what research questions would you like to see be addressed in your field?
SR Latch

- SR stands for Set/Reset Latch
  - Stores one bit of state ($Q$)
- Control what value is being stored with $S$, $R$ inputs
  - **Set**: Make the output 1
    
    $\overline{Q} = 0$
    
    ($S = 1$, $R = 0$, $Q = 1$)
  - **Reset**: Make the output 0
    
    $\overline{Q} = 1$
    
    ($S = 0$, $R = 1$, $Q = 0$)
  - **Hold**: Keep data stored
    
    $\overline{Q} = \overline{Q}_{\text{previous}}$
    
    ($S = 0$, $R = 0$, $Q = Q_{\text{previous}}$)
SR Latch Characteristic Equation

To analyze, break the feedback path

![State Diagram]

SR Latch Symbol

SR Latch Symbol

Sources: TSR, Katz, Boriello & Vahid
Avoiding S=R=1 Part 1: Level-Sensitive SR Latch

- Add input “C”
  - Change C to 1 only after S and R are stable
  - C is usually a clock (CLK)
Clocks

- **Clock** -- Pulsing signal for enabling latches; ticks like a clock
- **Synchronous** circuit: sequential circuit with a clock

- **Clock period**: time between pulse starts
  - Above signal: period = 20 ns
- **Clock cycle**: one such time interval
  - Above signal shows 3.5 clock cycles
- **Clock duty cycle**: time clock is high
  - 50% in this case
- **Clock frequency**: 1/period
  - Above: \( \text{freq} = \frac{1}{20\text{ns}} = 50\text{MHz} \)

<table>
<thead>
<tr>
<th>Freq</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 GHz</td>
<td>0.01 ns</td>
</tr>
<tr>
<td>10 GHz</td>
<td>0.1 ns</td>
</tr>
<tr>
<td>1 GHz</td>
<td>1 ns</td>
</tr>
<tr>
<td>100 MHz</td>
<td>10 ns</td>
</tr>
<tr>
<td>10 MHz</td>
<td>100 ns</td>
</tr>
</tbody>
</table>
Clock question

The clock shown in the waveform below has:

A. Clock period of 4ns with 250MHz frequency
B. Clock duty cycle 75%
C. Clock period of 1ns with 1GHz frequency
D. A. & B.
E. None of the above

\[ \frac{3}{4} = 75\% \]

A. Clock period of 4ns with 250MHz frequency
B. Clock duty cycle 75%
C. Clock period of 1ns with 1GHz frequency
D. A. & B.
E. None of the above

Sources: TSR, Katz, Boriello & Vahid
Avoiding $S=R=1$ Part 2: Level-Sensitive D Latch

- SR latch requires careful design so SR=11 never occurs
- D latch helps by inserting the inverter between S & R inputs
  - Inserted inverter ensures R is always the opposite of S when C=1

Sources: TSR, Katz, Boriello & Vahid
# D Latch Truth Table

```
<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>D</th>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Sources: TSR, Katz, Boriello & Vahid
D Latch Summary

- Two inputs: \( CLK, D \)
  - \( CLK \): controls when the output changes
  - \( D \) (the data input): controls what the output changes to

- Function
  - When \( CLK = 1 \),
    \( D \) passes through to \( Q \) (transparent)
  - When \( CLK = 0 \),
    \( Q \) holds its previous value (opaque)

- (Mostly) avoids invalid case \( Q = Q' \)
Assume that data in all latches is initially 0. Input Y=1 and Clk transitions from 0->1. When Clk=0 again, the stored values in latches are:

A. Q1=1, Q2=0, Q3=0, Q4=0 for both clock A & B
B. Q1=1, Q2=1, Q3=1, Q4=1 for clock A
   Q1=1, Q2=0, Q3=0, Q4=0 for clock B
C. Q1=1, Q2=1, Q3=1, Q4=1 for both clocks
D. More information is needed to determine the answer
E. None of the above
- **Flip-flop**: Bit storage that stores on the clock edge, not level
- Master-slave design: master loads when Clk=0, then slave when Clk=1
D Flip-Flop: Characteristic Equation

Characteristic Equation

\[ Q(t+1) = D(t) \]
Bit Storage Overview

SR latch

S=1 sets Q to 1, R=1 resets Q to 0. Problem: SR=11 yield undefined Q.

S and R only have effect when C=1. We can design outside circuit so SR=11 never happens when C=1. Problem: avoiding SR=11 can be a burden.

SR latch

SR can’t be 11 if D is stable before and while C=1, and will be 11 for only a brief glitch even if D changes while C=1. Problem: C=1 too long propagates new values through too many latches: too short may not enable a store.

D flip-flop

Only loads D value present at rising clock edge, so values can’t propagate to other flip-flops during same clock cycle. Tradeoff: uses more gates internally than D latch, and requires more external gates than SR – but gate count is less of an issue today.

SRAM - cache

Sources: TSR, Katz, Boriello & Vahid
Rising vs. Falling Edge D Flip-Flop

Symbol for rising-edge triggered D flip-flop

Symbol for falling-edge triggered D flip-flop

Internal design: Just invert servant clock rather than master

The triangle means clock input, edge triggered

Rising edges

Falling edges

Sources: TSR, Katz, Boriello & Vahid
Enabled D-FFs

• **Inputs:** $CLK$, $D$, $EN$
  – The enable input ($EN$) controls when new data ($D$) is stored

• **Function**
  – $EN = 1$: $D$ passes through to $Q$ on the clock edge
  – $EN = 0$: the flip-flop retains its previous state

Sources: TSR, Katz, Boriello & Vahid
Additional D-FF Features

• **Reset** (set state to 0) – R
  – synchronous: $D_{\text{new}} = R' \cdot D_{\text{old}}$ (when next clock edge arrives)
  – asynchronous: doesn't wait for clock

• **Preset or set** (set state to 1) – S (or sometimes P)
  – synchronous: $D_{\text{new}} = D_{\text{old}} + S$ (when next clock edge arrives)
  – asynchronous: doesn't wait for clock

• **Both reset and preset**
  – $D_{\text{new}} = R' \cdot D_{\text{old}} + S$ (set-dominant)
  – $D_{\text{new}} = R' \cdot D_{\text{old}} + R'S$ (reset-dominant)

• **Selective input capability** (input enable or load) – LD or EN
  – multiplexor at input: $D_{\text{new}} = LD' \cdot Q + LD \cdot D_{\text{old}}$
  – load may or may not override reset/set (usually R/S have priority)

• **Complementary outputs** – Q and Q'
Comparison of latches and flip-flops

- **Positive edge-triggered flip-flop**
  - Inputs: D, Q
  - Clock: CLK

- **Level-sensitive latch**
  - Inputs: D, G
  - Clock: CLK
  - Outputs: Q_FF, Q_latch

Sources: TSR, Katz, Boriello & Vahid
D Flip-Flops

Assume that the data in all D-FFs is initially 0. Input Y=1. When Clk goes from 0->1, the stored values in D-FFs are:

A. $Q_1=1$, $Q_2=0$, $Q_3=0$, $Q_4=0$ for both clock A & B
B. $Q_1=1$, $Q_2=1$, $Q_3=1$, $Q_4=1$ for clock A
   $Q_1=1$, $Q_2=0$, $Q_3=0$, $Q_4=0$ for clock B
C. $Q_1=1$, $Q_2=1$, $Q_3=1$, $Q_4=1$ for both clocks
D. More information is needed to determine the answer
E. None of the above
CSE140: Registers and Counters

Prof. Tajana Simunic Rosing
Bit Storage Overview

**SR latch**
- S (set)
- R (reset)
- S=1 sets Q to 1, R=1 resets Q to 0.
- Problem: SR=11 yield undefined Q.

**Level-sensitive SR latch**
- S and R only have effect when C=1.
- We can design outside circuit so SR=11 never happens when C=1.
- Problem: avoiding SR=11 can be a burden.

**D latch**
- SR can’t be 11 if D is stable before and while C=1, and will be 11 for only a brief glitch even if D changes while C=1.
- Problem: C=1 too long propagates new values through too many latches: too short may not enable a store.

**D flip-flop**
- Only loads D value present at rising clock edge, so values can't propagate to other flip-flops during same clock cycle.
- Tradeoff: uses more gates internally than D latch, and requires more external gates than SR – but gate count is less of an issue today.

Sources: TSR, Katz, Boriello & Vahid
Timing diagrams of D-FFs

\[ \text{D}_1 = A \text{ NOR } Q_0 \]

Sources: TSR, Katz, Boriello & Vahid
Comparison of latches and flip-flops

- Positive edge-triggered flip-flop
- Level-sensitive latch

Sources: TSR, Katz, Boriello & Vahid
Building blocks with FFs: Basic Register

Sources: TSR, Katz, Boriello & Vahid
Shift register

• Holds & shifts samples of input

Sources: TSR, Katz, Boriello & Vahid
Pattern Recognizer

- Combinational function of input samples

Sources: TSR, Katz, Boriello & Vahid
Design of a Universal Shift Register

### Nth Cell

- **input:** 4 lines
- **output:** 4 lines
- **clock:** 1 line
- **CLEAR:** 1 line
- **s0** and **s1** control mux

#### Table

<table>
<thead>
<tr>
<th>clear</th>
<th>s0</th>
<th>s1</th>
<th>new value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>output</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>output value of FF to left (shift right)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>output value of FF to right (shift left)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>input</td>
</tr>
</tbody>
</table>

#### CLEAR

- **CLEAR is:**
  - A) SYNCH
  - B) ASYNCH
  - C) Not the Above
Counters

• Sequences through a fixed set of patterns
Finite State Machines
Circuit Specifications

- Combinational Logic
  - Truth tables, Boolean equations, logic diagrams (no feedback)
- Sequential Networks: State Diagram (Memory)
  - State and Excitation Tables
  - Characteristic Expression
  - Logic Diagram (FFs and feedback loops)

RTL: Register-Transfer Level Description
Finite State Machines: Two Bit Counter Example

Symbol/ Circuit

2 bit Counter

State Diagram

State Table

<table>
<thead>
<tr>
<th>Current state</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$</td>
<td>$S_1$</td>
</tr>
<tr>
<td>$S_1$</td>
<td>$S_2$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>$S_3$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$S_0$</td>
</tr>
</tbody>
</table>

$Q_1(t)$ | $Q_0(t)$ | $Q_1(t+1)$ | $Q_0(t+1)$ |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
</tr>
<tr>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
<td>$\phi$</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello & Vahid
Which is the most likely circuit realization of the two bit counter?

**State Table**

<table>
<thead>
<tr>
<th>$Q_1(t)$</th>
<th>$Q_0(t)$</th>
<th>$Q_1(t+1)$</th>
<th>$Q_0(t+1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

> $2 \text{ bit} \approx 2 \text{ FF}$

A. **Combinational circuit**

B. **Circuit with no flip flops**

C. **Circuit with 2 flip flops**

Sources: TSR, Katz, Boriello & Vahid
Two Bit Counter Circuit

State Table

<table>
<thead>
<tr>
<th>$Q_1(t)$</th>
<th>$Q_0(t)$</th>
<th>$Q_1(t+1)$</th>
<th>$Q_0(t+1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

We store the current state using D-flip flops so that:

• Inputs to the combinational circuit don’t change while the next output is computed

• The transition to the next state only occurs at the rising edge of the clock

$Q_0(t+1) = Q_0(t) = D_0$

$Q_1(t+1) = Q_1(t) + Q_0(t)' Q_1(t)$

Implementation of 2-bit counter

Sources: TSR, Katz, Boriello & Vahid
Finite State Machine (FSM) Definition

• FSM consists of
  – Set of states
  – Set of inputs, set of outputs
  – Initial state
  – Set of transitions
    • Only one can be true at a time

• FSM representations:
  – State diagram
  – State table
FSM Controller Design Process with a Three Bit Counter Example

1. State Diagram
2. State Table
3. State Assignments
4. Excitation Table
   (present state, inputs; next state, outputs)
5. Circuit

Excitation Table with Assigned State Patterns

<table>
<thead>
<tr>
<th>C3</th>
<th>C2</th>
<th>C1</th>
<th>N3</th>
<th>N2</th>
<th>N1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Mealy and Moore Machines

Mealy Machine: $y_i(t) = f_i(X(t), S(t))$
Moore Machine: $y_i(t) = f_i(S(t))$

$s_i(t+1) = g_i(X(t), S(t))$
Life on Mars?

Mars rover has a binary input $x$. When it receives the input sequence $x(t-2, t) = 001$ from its life detection sensors, it means that the rover has detected life on Mars 😊 and the output $y(t) = 1$, otherwise $y(t) = 0$ (no life on Mars 😞).

This pattern recognizer should have
A. One state because it has one output
B. One state because it has one input
C. Two states because the input can be 0 or 1
D. More than two states because ….
E. None of the above
Which of the following diagrams is a correct Mealy solution for the 001 pattern recognizer on the Mars rover?

A. 1/0
B. 1/0
C. Both A and B are correct
D. None of the above

Sources: TSR, Katz, Boriello & Vahid
What does state table need to show to design controls of C1?
A. (current input $x(t)$, current state $S(t)$ vs. next state, $S(t+1)$)
B. (current input, current state vs. current output $y(t)$)
C. (current input, current state vs. current output, next state)
D. None of the above
State Diagram => State Table with State Assignment

State Assignment
S0: 00
S1: 01
S2: 10

S(t)\x | 0 | 1
-----|---|---
S0   | S1,0 | S0,0
S1   | S2,0 | S0,0
S2   | S2,0 | S0,1

Mealy Machine

Sources: TSR, Katz, Boriello & Vahid
State Diagram => State Table => Excitation Table => Circuit

<table>
<thead>
<tr>
<th>Q_1(t) Q_0(t) x</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01,0</td>
<td>00,0</td>
</tr>
<tr>
<td>01</td>
<td>10,0</td>
<td>00,0</td>
</tr>
<tr>
<td>10</td>
<td>10,0</td>
<td>00,1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>Q_1 Q_0 x</th>
<th>D_1</th>
<th>D_0</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Borella, Vahid
State Diagram => State Table => Excitation Table => Circuit

<table>
<thead>
<tr>
<th>Q₁(t) Q₀(t)x</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01,0</td>
<td>00,0</td>
</tr>
<tr>
<td>01</td>
<td>10,0</td>
<td>00,0</td>
</tr>
<tr>
<td>10</td>
<td>10,0</td>
<td>00,1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>Q₁Q₀x</th>
<th>D₁</th>
<th>D₀</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
State Diagram => State Table => Excitation Table => Circuit

\[ D_1(t) = x'Q_0 + x'Q_1 \]
\[ D_0(t) = Q'_1Q'_0 x' \]
\[ y = Q_1x \]
State Diagram => State Table => Excitation Table => Circuit

D_1(t) = x'Q_0 + x'Q_1
D_0(t) = Q'_1Q'_0 x'

y = Q_1x

Mealy Machine
CSE140: Finite State Machines

Prof. Tajana Simunic Rosing
Class Status, Updates & Plans

• What we covered:
  – Combinational logic
  – Latches, FFs
  – Registers, counters, started FSMs

• Plan for today:
  – FSMs
  – Timing: How fast can my CPU clock run?

• Upcoming deadlines:
  – HW #4 due, HW#5 assigned
Exam #1 Results

- Excellent work!

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>95.06</td>
</tr>
<tr>
<td>Median</td>
<td>98</td>
</tr>
<tr>
<td>Std dev</td>
<td>13.45</td>
</tr>
<tr>
<td>Min</td>
<td>38</td>
</tr>
<tr>
<td>Max</td>
<td>110</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello & Vahid
Life on Mars?

Mars rover has a binary input $x$. When it receives the input sequence $x(t-2, t) = 001$ from its life detection sensors, it means that the rover has detected life on Mars 🙂 and the output $y(t) = 1$, otherwise $y(t) = 0$ (no life on Mars 😞).

This pattern recognizer should have
A. One state because it has one output
B. One state because it has one input
C. Two states because the input can be 0 or 1
D. More than two states because ….
E. None of the above

Sources: TSR, Katz, Boriello & Vahid
Moore FSM for the Mars Life Recognizer

Which of the following diagrams is a correct Moore solution to the ‘001’ pattern recognizer?

A. 1/0

B. 1

C. Both A and B are correct

D. None of the above

Sources: TSR, Katz, Boriello & Vahid
Moore Mars Life Recognizer: FF Input Specs

Pattern Recognizer ‘001’

What does state table need to show to design controls of C2?
A. (current input x(t), current state S(t) vs. next state, S(t+1))
B. (current input, current state vs. current output y(t))
C. (current state vs. current output y(t) and next state)
D. (current state vs. current output y(t))
E. None of the above

Moore Machine

Sources: TSR, Katz, Boriello & Vahid
Moore Mars Life Recognizer: State Table

<table>
<thead>
<tr>
<th>S(t)\x</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>S1,0</td>
<td>S0,0</td>
</tr>
<tr>
<td>S1</td>
<td>S2,0</td>
<td>S0,0</td>
</tr>
<tr>
<td>S2</td>
<td>S2,0</td>
<td>S3,0</td>
</tr>
<tr>
<td>S3</td>
<td>S1,1</td>
<td>S0,1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q1Q0\x</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01,0</td>
<td>00,0</td>
</tr>
<tr>
<td>01</td>
<td>10,0</td>
<td>00,0</td>
</tr>
<tr>
<td>10</td>
<td>10,0</td>
<td>11,0</td>
</tr>
<tr>
<td>11</td>
<td>01,1</td>
<td>00,1</td>
</tr>
</tbody>
</table>

Q1(t+1)Q0(t+1), y

<table>
<thead>
<tr>
<th>ID</th>
<th>Q1Q0\x</th>
<th>D1</th>
<th>D0</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
### Mars Life Recognizer: Combinational Circuit Design

**Inputs:** 3 inputs  
**Knots:** 3 knots

<table>
<thead>
<tr>
<th>id</th>
<th>$Q_1Q_0x$</th>
<th>$D_1$</th>
<th>$D_0$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**$D_1(t)$:**

$$D_1(t) = \delta_0Q_1x^3 + \delta_0'Q_1$$

**$D_0(t)$:**

$$D_0(t) = \delta_0Q_1x^3 + \delta_0'Q_1$$

**$y(t)$:**

$$y(t) = \delta_0Q_1x^3 + \delta_0'Q_1$$

Sources: TSR, Katz, Boriello & Vahid
Mars Life Recognizer Circuit Implementation

\[ D_1 = Q_0 Q_1' x_1 + Q_0' Q_1 \]

State Diagram \(\Rightarrow\) State Table \(\Rightarrow\) Excitation Table \(\Rightarrow\) Circuit

Moore Machine
15 cents for candy! Watch out – no change!

- **Moore machine**
  - outputs associated with state

- **Mealy machine**
  - outputs associated with transitions

Sources: TSR, Katz, Boriello & Vahid
Example: Moore implementation

- Encode states and map to logic

<table>
<thead>
<tr>
<th>Present state</th>
<th>Inputs</th>
<th>Next state</th>
<th>Output open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 Q0</td>
<td>D N</td>
<td>D1 D0</td>
<td>open</td>
</tr>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>X X 1 X</td>
<td>D N</td>
<td>D1 D0</td>
<td></td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 1</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 0</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>X X 1 X</td>
<td>X X 1 X</td>
<td>X X 1 X</td>
<td></td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 1</td>
<td>0 1</td>
<td></td>
</tr>
</tbody>
</table>

open = Q0 Q1
Example: Mealy implementation

<table>
<thead>
<tr>
<th>present state</th>
<th>inputs</th>
<th>next state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 Q0</td>
<td>D N</td>
<td>D1 D0 open</td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>0 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>0 1 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>1 0</td>
<td>1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1 1</td>
<td>– – –</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 0</td>
<td>0 1 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0 1</td>
<td>1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>1 0</td>
<td>1 1 1</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1 1</td>
<td>– – –</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>0 0</td>
<td>1 0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>1 1</td>
<td>1 1 1</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1 1</td>
<td>– – –</td>
<td>0</td>
</tr>
</tbody>
</table>

open = \( Q_0 Q_1 + N Q_1 + D Q_1 + Q_0 \cdot D \)

Sources: TSR, Katz, Boriello & Vahid
Timing Constraints in Sequential Designs
• Our seemingly logically correct design can go wrong – signals don’t travel in zero time 😞
• We next look at timing constraints for combinational and sequential logic.
I. Min delay of a gate, also called contamination delay: $t_{cd}$

Minimum time from when an input changes until the output starts to change

II. Max delay of a gate, also called propagation delay: $t_{pd}$

Maximum time from when an input changes until the output is guaranteed to reach its final value (i.e., stop changing)
Which path in the above circuit determines the contamination delay of the circuit (assuming the delay of all the gates is the same)?

A. Blue path
B. Red path
C. Both
D. Neither
Which path in the above circuit determines the propagation delay of the circuit (assuming the delay of all the gates is the same)?

A. Blue path
B. Red path
C. Both
D. Neither
D-FF Input Constraints: Setup and Hold Times

I. Setup time: $t_{\text{setup}}$
   Time before the clock edge that data must be stable (i.e. not change)

II. Hold time: $t_{\text{hold}}$
   Time after the clock edge that data must be stable

Aperture time: $t_a$
Time around clock edge that data must be stable ($t_a = t_{\text{setup}} + t_{\text{hold}}$)

Sources: TSR, Katz, Boriello & Vahid
I. Min delay of FF, also called contamination delay or min CLK to Q delay: $t_{ccq}$
Time after clock edge that $Q$ might be unstable (i.e., starts changing)

II. Max delay of FF, also called propagation delay or maximum CLK to Q delay: $t_{pcq}$
Time after clock edge that the output $Q$ is guaranteed to be stable (i.e. stops changing)
The timing of which of the following signals can cause a setup-time violation?

A. The input signal $D(t)$
B. The output signal $Q(t)$
C. Both of the above
D. None of the above
Causes of Timing Issues in Sequential Circuits

- Input to a FF comes from the output of another FF through a combinational circuit
- The FF and combinational circuit have a min & max delay

Which of the following violations occurs if max delay of R1 is zero & max delay of the combinational circuit is equal to the clock period?

A. Hold time violation for R2
B. Setup violation for R2
C. Hold time violation for R1
D. Setup violation for R1
E. None of the above

Sources: TSR, Katz, Boriello & Vahid
Setup Time Constraint

- Input to a FF comes from the output of another FF through a combinational circuit.
- The FF and combinational circuit have a min & max delay.

**Setup time constraint:**

\[ T_c \geq t_{setup} + \text{max delay(FF)} + \text{max delay(combinational)} \]

\[ T_c \geq t_{pcq} + t_{pd} + t_{setup} \]
Causes of Timing Issues in Sequential Circuits

- Input to a FF comes from the output of another FF through a combinational circuit.
- The FF and combinational circuit have a min & max delay.

Which of the violations would occur if the min delay of R1 was zero and the combinational circuit was just a wire?

A. Hold time violation for R2
B. Setup violation for R2
C. Hold time violation for R1
D. Setup violation for R1
E. None of the above

Sources: TSR, Katz, Boriello & Vahid
Hold Time Constraint

- Input to a FF comes from the output of another FF through a combinational circuit
- The FF and combinational circuit have a min & max delay

Hold time constraint:
\[ t_{\text{hold}} < \min(\text{delay(FF)}) + \min(\text{delay(combinational)}) \]
\[ t_{\text{hold}} < t_{\text{ccq}} + t_{\text{cd}} \]
Once a flip flop has been built, its timing characteristics stay fixed: $t_{\text{setup}}$, $t_{\text{hold}}$, $t_{\text{ccq}}$, $t_{\text{pcq}}$.

What about the clock? Does the clock edge arrive at the same time to all the D-FFs on the chip?
The clock doesn’t arrive at all registers at the same time

**Skew**: difference between the two clock edges

Perform the **worst case analysis**
Setup Time Constraint with Skew

- In the worst case, CLK2 is earlier than CLK1
- $t_{pcq}$ is max delay through FF, $t_{pd}$ is max delay through logic

\[ T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew} \]

\[ t_{pd} \leq T_c - (t_{pcq} + t_{setup} + t_{skew}) \]
Hold Time Constraint with Skew

- In the worst case, CLK2 is later than CLK1
- $t_{ccq}$ is min delay through FF, $t_{cd}$ is min delay through logic

\[ t_{ccq} + t_{cd} > t_{hold} + t_{skew} \]
\[ t_{cd} > t_{hold} + t_{skew} - t_{ccq} \]

Sources: TSR, Katz, Boriello & Vahid
Timing Analysis Example

Timing Characteristics

- $t_{ccq} = 30 \text{ ps}$
- $t_{pcq} = 50 \text{ ps}$
- $t_{setup} = 60 \text{ ps}$
- $t_{hold} = 70 \text{ ps}$
- $t_{pd} = 35 \text{ ps}$
- $t_{cd} = 25 \text{ ps}$

$t_{pd} = 3 \times 35 \text{ ps} = 105 \text{ ps}$

$t_{cd} = 25 \text{ ps}$

Setup time constraint:

$T_c \geq (50 + 105 + 60) \text{ ps} = 215 \text{ ps}$

$f_c = 1/T_c = 4.65 \text{ GHz}$

Hold time constraint:

$t_{ccq} + t_{cd} > t_{hold}$?

$(30 + 25) \text{ ps} > 70 \text{ ps}$? **No!**

Sources: TSR, Katz, Boriello & Vahid
Timing Analysis Example

Add buffers to the short paths:

\[ t_{pd} = 3 \times 35 \text{ ps} = 105 \text{ ps} \]

\[ t_{cd} = 2 \times 25 \text{ ps} = 50 \text{ ps} \]

**Setup time constraint:**

\[ T_c \geq (50 + 105 + 60) \text{ ps} = 215 \text{ ps} \]

\[ f_c = 1/T_c = 4.65 \text{ GHz} \]

**Hold time constraint:**

\[ t_{ccq} + t_{cd} > t_{hold} \]

\[ (30 + 50) \text{ ps} > 70 \text{ ps} \quad \text{Yes!} \]

Sources: TSR, Katz, Boriello & Vahid
Sequential Circuit Design Summary

- SRAM memory, SR Latch, D Latch, D-FF
- Design procedure for FSMs
  1. Capture FSM
  2. Create state table
  3. Assign the states
  4. Excitation table
  5. Implement the combinational logic
- Mealy vs. Moore FSM
- Non-ideal properties of FFs
  - Setup/hold time constraints
  - Maximum operating frequency
  - Clock skew
MORE EXAMPLES
New FF Design

Sources: TSR, Katz, Boriello & Vahid
Counter Design
Design FSM with Minimum Number of States

- If two inputs are equal during any four consecutive clock cycles, $w_1 = w_2$, the circuit produces output $z=1$; else, $z=0$. 
### FSM Design

#### State Transition Table

<table>
<thead>
<tr>
<th>$Q_1$</th>
<th>$Q_0$</th>
<th>$a$</th>
<th>$D_1$</th>
<th>$D_0$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### State Diagram

- **Input:** $a$
- **Output:** $y$
- States: 00, 01, 10, 11
- Transitions:
  - From 00 to 00 on input $a$
  - From 00 to 01 on input $a'$
  - From 01 to 11 on input $a$
  - From 01 to 10 on input $a'$
  - From 10 to 10 on input $a$
  - From 10 to 11 on input $a'$
  - From 11 to 00 on input $a'$
  - From 11 to 11 on input $a$

- Output $y$:
  - $y = 0$ for states 00, 01, and 10
  - $y = 1$ for states 11
# FSM Analysis

<table>
<thead>
<tr>
<th>Q1Q0A</th>
<th>Q1(t+1)</th>
<th>Q0(t+1)</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The FSM diagram on the right illustrates the circuit's logic. The states Q1 and Q0 are updated based on the input A and the current state. The output Y is determined by the updated states Q1(t+1) and Q0(t+1).

Sources: TSR, Katz, Boriello & Vahid
FSM design: Multiple input counter

- Given FSM of a multiple input counter, design the circuit implementing its functionality

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>S0 S1 S2 S3</td>
<td>1</td>
</tr>
<tr>
<td>S1</td>
<td>S0 S3 S1 S3</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>S1 S0 S0 S3</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>S1 S3 S2 S0</td>
<td>1</td>
</tr>
</tbody>
</table>

Input Truth Table:

<table>
<thead>
<tr>
<th>Input</th>
<th>00</th>
<th>00</th>
<th>01</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>11</td>
<td>00</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>01</td>
<td>00</td>
<td>10</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello & Vahid
Multiple input counter: Logic for D-FF

- Derive logic equations for inputs of D-FF

<table>
<thead>
<tr>
<th>Input</th>
<th>State</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>01</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>11</td>
<td>00</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>01</td>
<td>00</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello & Vahid
Timing diagrams w setup and hold times

\[ t_{ccq} + t_{cd} > t_{hold} + t_{skew} \]

\[ T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew} \]
ALU: Minimum POS of Decoder-Mux Circuit