Sequential Circuit Analysis

☐ Objectives

- This section introduces synchronous sequential circuits with the following goals:
  - Give a precise definition of synchronous sequential circuits.
  - Introduce several structural and behavioral models for synchronous sequential circuits.
  - Demonstrate by example how to analyze synchronous sequential circuits by deriving their behavior from a structural description.
  - Demonstrate how to represent the behavior of a synchronous sequential circuit with Verilog.

☐ Reading assignment

- Section 3.4
- Sections 4.4, 4.5, and 4.6

☐ Topics

- Sequential Circuit Models
- Mealy and Moore Models
- Blocking and Non-blocking Assignment statements
- Verilog representation of sequential circuits
10.1. Synchronous Sequential Circuits

Definition. A sequential circuit is said to be a *synchronous sequential circuit* if it satisfies the following conditions:

- There is at least one flip-flop in every loop
- All flip-flops have the same type of dynamic clock
- All clock inputs of all the flip-flops are driven by the same clock signal.

Sequential Circuit Canonical Form

Any synchronous sequential circuit can be drawn in this form by pulling the flip-flops to the bottom of the figure (think of the lines as elastic). Since all loops have a flip-flop in them, this will leave the remaining circuit without loops, and hence combinational.
1. Synchronous Sequential Circuits

Definition. A sequential circuit is said to be a synchronous sequential circuit if it satisfies the following conditions:
- There is at least one flip-flop in every loop
- All flip-flops have the same type of dynamic clock
- All clock inputs of all the flip-flops are driven by the same clock signal.

Sequential Circuit Canonical Form

- Any synchronous sequential circuit can be drawn in this form by pulling the flip-flops to the bottom of the figure (think of the lines as elastic). Since all loops have a flip-flop in them, this will leave the remaining circuit without loops, and hence combinational.
There are two versions of this model called the Mealy Model and the Moore model. The only difference is in the way the output signals are generated.

- Mealy Model: Outputs depend on current state and inputs
- Moore Model: Outputs only depend on current state.

2. Synchronous Sequential Circuit Models

- Structural
  - Logic diagram
  - Excitation Equations
  - Output equations

- Behavioral
  - Transition and output equations
  - Transition table
  - State table
  - State diagram (graph)

- SSC Analysis: Derive one of the behavioral models from an instance of a structural model

- SSC Synthesis: Derive a structural model from one of the behavioral models
### 10.3. Analysis Examples

**Example # 1**

Derive the excitation and output equations:

- \( J_0 = X' + Q_1' \)
- \( J_1 = Q_0 \)
- \( K_0 = 1 \)
- \( K_1 = X + Q_0 \)
- \( Y = X \cdot Q_0' \cdot Q_1 + X' \cdot Q_0 \cdot Q_1 \)

Derive the transition equations from the excitation equations:

- **Use the characteristic equation for JK flip-flops:**
  \[ Q^* = JQ' + K'Q \]
- **The resulting transition equations are:**
  \[ Q_0^* = J_0 \cdot Q_0' + K_0' \cdot Q_0 = (X' + Q_1') \cdot Q_0' + 0 \cdot Q_0 = X' \cdot Q_0' + Q_0' \cdot Q_1' \]
  \[ Q_1^* = J_1 \cdot Q_1' + K_1' \cdot Q_1 = Q_0 \cdot Q_1' + (X + Q_0)' \cdot Q_1 = Q_0 \cdot Q_1' + X' \cdot Q_0' \cdot Q_1 \]

**Construct the transition table from the transition equations:**

<table>
<thead>
<tr>
<th>( Q_1, Q_0 )</th>
<th>( X=0 )</th>
<th>( X=1 )</th>
<th>( X=0 )</th>
<th>( X=1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
<td>01</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>00</td>
<td>00</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( Q_1^* )</td>
<td>( Q_0^* )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Construct the state table and state diagram from the transition table:

- Assign 0 to 00, 1 to 01, 2 to 10, and 3 to 11 giving the following state table and diagram

<table>
<thead>
<tr>
<th>Q1, Q0</th>
<th>X=0</th>
<th>X=1</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>01</td>
<td>01</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>11</td>
<td>00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>00</td>
<td>00</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Q1* Q0* Y

<table>
<thead>
<tr>
<th>Q</th>
<th>X=0</th>
<th>X=1</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Transition Table | State Table

State Diagram

Example #2

- Derive the excitation and output equations:

\[
D2 = X \oplus Q2 \oplus Q0 \\
D1 = Q2 \\
D0 = Q1 \\
Y = X \oplus Q2 \oplus Q0
\]
Construct the excitation/transition table from the excitation/transition equations:

- Note that since D flip-flops are used, the sets of excitation and transition equations are the same. Therefore the transition table is obtained by plotting the excitation equations.

\[
Q_2^* = Y = X \oplus Q_2 \oplus Q_0; \quad Q_1^* = Q_2; \quad Q_0^* = Q_1;
\]

<table>
<thead>
<tr>
<th>Q2</th>
<th>Q1</th>
<th>Q0</th>
<th>X=0</th>
<th>X=1</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>000</td>
<td>100</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td>100</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>001</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>101</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>110</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>010</td>
<td>110</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td>111</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>011</td>
<td>111</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
Q_2^* Q_1^* Q_0^* \quad Y
\]

Construct the state table from the transition table

- Let 000 = A, 001 = B, 010 = C, 011 = D, 100 = E, 101 = F, 110 = G, 111 = H

<table>
<thead>
<tr>
<th>Q2</th>
<th>Q1</th>
<th>Q0</th>
<th>X=0</th>
<th>X=1</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>000</td>
<td>100</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 1</td>
<td>100</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>001</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>101</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>110</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>010</td>
<td>110</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 0</td>
<td>111</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>011</td>
<td>111</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
Q_2^* Q_1^* Q_0^* \quad Y
\]
Construct the state diagram from the state table

State Table:

<table>
<thead>
<tr>
<th>Q</th>
<th>X=0</th>
<th>X=1</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>E</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>F</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>F</td>
<td>B</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>G</td>
<td>C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>C</td>
<td>G</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
<td>D</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>D</td>
<td>H</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Q*</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

State Diagram:

Example #3 (Problem 7.18)

Derive the excitation and output equations:

\[ J_0 = K_0 = EN \]
\[ J_1 = K_1 = EN\cdot Q_0 \]
\[ MAX = EN\cdot Q_0\cdot Q_1 \]

Derive the transition equations from the excitation equations:

\[ Q_0^* = J_0\cdot Q_0' + K_0'\cdot Q_0 = EN\cdot Q_0' + EN'\cdot Q_0 \]
\[ Q_1^* = J_1\cdot Q_1' + K_1'\cdot Q_1 = EN\cdot Q_0\cdot Q_1' + (EN\cdot Q_0)\cdot Q_1 \]
Construct the transition table from the transition equations:

Transition Equations:

\[
Q_0^* = E\overline{N}Q_0' + EN'Q_0 \\
Q_1^* = E\overline{N}Q_0Q_1' + EN'Q_1 + E\overline{N}Q_0Q_1
\]

Construct the state table from the transition table:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q0</th>
<th>EN=0</th>
<th>EN=1</th>
<th>EN=0</th>
<th>EN=1</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>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</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>1</td>
</tr>
</tbody>
</table>

\[
Q1^* Q0^* \text{ MAX}
\]

Construct the state diagram from the state table:

State Table:

<table>
<thead>
<tr>
<th>Q</th>
<th>EN=0</th>
<th>EN=1</th>
<th>EN=0</th>
<th>EN=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
Q^* \text{ MAX}
\]
Example #4

- Logic Diagram

![Logic Diagram]

- Derive the excitation equations

\[
\begin{align*}
J_0 &= X \cdot Q_1' \cdot Q_0' \\
J_1 &= X \cdot Q_0 \\
K_0 &= X \cdot Q_1 \\
K_1 &= X \cdot Q_1 \cdot Q_0'
\end{align*}
\]

- Derive the transition equations from the excitation equations:

\[
\begin{align*}
Q_0^* &= J_0 \cdot Q_0' + K_0' \cdot Q_0 \\
&= (X \cdot Q_1' \cdot Q_0' \cdot Q_0' + (X \cdot Q_1') \cdot Q_0 \\
&= X \cdot Q_1' \cdot Q_0' + X' \cdot Q_0 + Q_1' \cdot Q_0 \\
Q_1^* &= J_1 \cdot Q_1' + K_1' \cdot Q_1 \\
&= (X \cdot Q_0) \cdot Q_1' + (X \cdot Q_1 \cdot Q_0') \cdot Q_1 \\
&= (X \cdot Q_0 \cdot Q_1' + (X' + Q_1' + Q_0) \cdot Q_1
&= X \cdot Q_0 \cdot Q_1' + X' \cdot Q_1 + Q_0 \cdot Q_1
\end{align*}
\]

- Derive the transition table from the transition equations:

<table>
<thead>
<tr>
<th>Q1 Q0</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| Q1* Q0* | Z
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Derive the state table from the transition table:

- Where 00 = A, 01 = B, 10 = C, 11 = D

<table>
<thead>
<tr>
<th>Q</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>Q*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Derive the state diagram from the state table:

4. Synchronous Sequential Circuits & Verilog

- Blocking vs. non-blocking assignment statements
  - Write code for the following circuit (a 2-bit shift register)

```
module bexample (D, Ck, Q1, Q2);
input D, Ck;
output Q1, Q2;
reg Q1, Q2;
always @(posedge Ck)
begin
  Q1 = D;
  Q2 = Q1;
end
endmodule
```
The assignment statements in the previous example are called *blocking*. They have the following properties:

- The assignment symbol is the equal sign (=).
- Multiple assignment statements in the same always block are executed in the order written.
  - In other words a blocking assignment statement will “block” until the assignment is complete before going to the next statement.
  - Hence Q1 first gets D and then Q2 gets the new value of Q1 in the previous example.

What we need is an assignment that does not change the state variables until the inputs of all of them have been computed. This is called a non-blocking assignment and has the following properties:

- The assignment symbol is <=
- All statements are evaluated using the values the variables have when the always block is entered.
- Only when the always block terminates are the variables updated with their new values.

The following module using blocking assignments will realize the circuit in the previous example:

```verilog
module nbexample (D, Ck, Q1, Q2);
  input D, Ck;
  output Q1, Q2;
  reg Q1, Q2;
  always @(posedge Ck)
  begin
    Q1 <= D;
    Q2 <= Q1;
  end
endmodule
```

As a general rule, it is best to use non-blocking assignments for the state variables of a sequential circuit and use blocking assignments in combinational logic modules.
Verilog description of example #1

```verilog
module example1 (Ck, X, Q, Y);
  input Ck, X;
  output Y;
  output [1:0] Q;
  reg [1:0] Q;
  always @(posedge Ck)
    case (Q)
      0: Q <= 1;
      1: Q <= 2;
      2: if (X) Q <= 0;
          else Q <= 3;
      3: Q <= 0;
      default: Q <= 2'bxx;
    endcase
  assign Y = ((Q == 2) & X) | ((Q == 3) & ~X);
endmodule
```

Alternate description of example 1

```verilog
module example1a (Ck, X, Q, Y);
  input Ck, X;
  output Y;
  output [1:0] Q;
  reg [1:0] Q, NXST;
  reg Y;
  always @(posedge Ck)
    Q <= NXST;
  always @(Q or X)
    case (Q)
      0: begin NXST = 2'b01; Y = 0; end
      1: begin NXST = 2; Y = 0; end
      2: if (~X) begin NXST = 3; Y = 0; end
          else begin NXST = 0; Y = 1; end
      3: begin NXST = 0; if (~X) Y = 1; else Y = 0; end
    endcase
endmodule
```
Another Example

module simple (Clock, Resetn, w, z);
  input Clock, Resetn, w;
  output z;
  reg [2:1] y, Y;
  parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10;

always @(w or y) // Define the next state combinational logic
  case (y)
    A: if (w) Y = B;
    else Y = A;
    B: if (w) Y = C;
    else Y = A;
    C: if (w) Y = C;
    else Y = A;
    default: Y = 2'bxx;
  endcase

always @(negedge Resetn or posedge Clock) // Define the flip-flops
  if (Resetn == 0) y <= A;
  else case (y)
    A: if (w) y <= B;
    else y <= A;
    B: if (w) y <= C;
    else y <= A;
    C: if (w) y <= C;
    else y <= A;
    default: y <= 2'bxx;
  endcase

assign z = (y == C); // Define output logic
endmodule

Alternate description of previous example:

module simple (Clock, Resetn, w, z);
  input Clock, Resetn, w;
  output z;
  reg [2:1] y;
  parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10;

always @(negedge Resetn or posedge Clock) // Define the sequential block
  if (Resetn == 0) y <= A;
  else case (y)
    A: if (w) y <= B;
    else y <= A;
    B: if (w) y <= C;
    else y <= A;
    C: if (w) y <= C;
    else y <= A;
    default: y <= 2'bxx;
  endcase

assign z = (y == C); // Define output
endmodule
5. Review

- Sequential circuit models:
  - Canonical form
  - Mealy vs. Moore models
  - Excitation & output equations
  - Transition equations & transition table
  - State table and state diagram

- Verilog models
  - Blocking vs. non-blocking assignment statements

- The treatment of clocks in all the models.