DIGITAL TECHNICS
Dr. Bálint Pődör
Obuda University,
Microelectronics and Technology Institute

6. LECTURE (ANALYSIS AND SYNTHESIS OF SYNCHRONOUS SEQUENTIAL CIRCUITS)

SYNTHESIS: GENERAL CONCEPTS
Synchronous sequential circuits synthesis procedure
- Word description of problem (hardest; art, not science)
- Derive state diagram and state table
- Minimize (moderately hard)
- Assign states (very hard)
- Produce state and output transition tables
- Determine what FFs to use and find their excitation maps
- Derive output equations/K-maps
- Obtain the logic diagram

INTRODUCTORY EXAMPLES
- Control of an alarm system – role of memory
- Universal shift register – design considerations
- Pattern recognizer – shift register application
- Traffic light control - counter application
- Traffic sign control – designing with next-state

EXAMPLE: CONTROL OF AN ALARM SYSTEM
Control of an alarm system is one of the simplest case of sequential logic.
Alarm is ON when the sensor generates a positive voltage, SET, in response to an undesirable event. Once alarm is on, it can only be turned off manually through a RESET button.
Memory is needed to remember the alarm has to be active until the reset signal arrives.

EXAMPLE: APPLICATION OF THE SR LATCH
- An important application of SR latches is for recording short lived events
  - e.g. pressing an alarm bell in a hospital
REGISTERS: DESIGN ASPECTS

Collection of flip-flops with similar controls and logic
Stored values somehow related (e.g. form binary value)
Share clock, reset, and set lines
Similar logic at each stage

UNIVERSAL SHIFT REGISTER

- Holds N values
- Serial or parallel inputs
- Serial or parallel outputs
- Permits shift left or right
- Shift in new values from left or right

Pattern recognizer

- Combinational function of input samples
- In this case, recognizing the pattern 1001 on the single input signal

TRAFFIC LIGHT CONTROL

Design a traffic light control logic for the traffic signal sequence shown below:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>25 sec</td>
</tr>
<tr>
<td>Red &amp; Yellow</td>
<td>5 sec</td>
</tr>
<tr>
<td>Green</td>
<td>25 sec</td>
</tr>
<tr>
<td>Yellow</td>
<td>5 sec</td>
</tr>
</tbody>
</table>

Consider generating three signals R, Y, and G representing traffic light, the lights being switched on when a signal is 1. The timing sequence can be generated using a mod12 counter driven by a 5 sec period clock (0.2/sec repetition rate).
TRAFFIC LIGHT CONTROL

Signal sequence
- Red 25 sec
- Red & Yellow 5 sec
- Green 25 sec
- Yellow 5 sec

5 sec period CLK

TRAFFIC LIGHT: TRUTH TABLE AND CONTROL EQUATIONS

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Q_a</th>
<th>Q_c</th>
<th>Q_s</th>
<th>R</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Control equations
- \( R = \Sigma 4 (0-5)X(6,7,14,15) \)
- \( G = \Sigma 4 (5,13)X(6,7,14,15) \)
- \( G = \Sigma 4 (8-12)X(6,7,14,15) \)

TRAFFIC LIGHT CONTROL

With standard minimization the required logic to control the red, yellow, and green lights can easily be determined.

It was an arbitrary decision to begin the sequence at the 0 0 0 0 state of the counter. Beginning at any other state would result in a different but equally effective design.

TRAFFIC LIGHT CONTROL LOGIC: KARNAUGH MAP

Minimization and implementation is straightforward

TRAFFIC LIGHT CONTROL: MODEL
SYNTHESIS OF SYNCHRONOUS CIRCUITS: GENERAL PROCEDURE

1. Constructing the state transition diagram.
2. Selection or specifying the encoding of the states.
3. Constructing the state transition tables. It gives for each cycle the next-state of each flip-flop in the function of the previous states of all flip-flops and in the function of the control conditions (up/down).
4. Selection or specifying the type of flip-flop used in the implementation. Excitation table of the flip-flop type.
5. Determination of the logic functions of the control input(s) of each flip-flop. Performing the necessary or appropriate minimization.
6. Selection of the types of logic gates to be used and implementation of the feedback/control network.

IMPLEMENTATION

Example solution:

- Logic diagram

\[ D_1 = y_1 \bar{y}_2 + xy_2 \]
\[ D_2 = \bar{x}y_1 + x \bar{y}_1 = x \oplus y_1 \]
\[ z = x \bar{y}_1 y_2 + \bar{x}y_1 \bar{y}_2 \]

SYNTHESIS: A SIMPLE EXAMPLE

Example:

- Find D FF realization of circuit defined in table (a)
- (b): state assignment
- (c): transition table
- (d): output K-map
- (e): excitation K-map

SYNTHESIS

Example is same as before, but use JK FFs
(a): transition table; (b): Excitation tables; (c): Excitation maps

IMPLEMENTATION

Example JK FF solution:

- Logic diagram

\[ J_1 = X \bar{Y}_2 \]
\[ K_1 = \bar{X} Y_2 \]
\[ J_2 = X \bar{Y}_1 + X Y_1 \]
\[ K_2 = X Y_1 + \bar{X} Y_1 \]

COMPARISON OF TWO DESIGNS
**COMPARISON OF DIFFERENT DESIGNS**

| Flip-flop: | D | D | JK | JK |
| Logic: | AND-OR | XOR | AND-OR | XOR |
| Pin count: | 20 | 16 | 28 | 15 |
| Gate count: | 9 | 7 | 11 | 7 |

---

**SYNTHESIS OF SEQUENTIAL CIRCUIT: A CASE STUDY**

- Synthesize a network which determines the parity of a four bit serial code word.
- Should indicate the parity of the incoming code word after receiving the 4-th bit as
  - 1 if the parity is odd,
  - 0 if the parity is even.
- The output is irrelevant (don’t care) during the first three cycle of the period.

---

**CHARACTERISTICS**

- Because there are two input conditions, two connecting lines emanate from each node.
- The network returns to its initial state after the fourth cycle.
- The operation of the network is cyclic, the length of the period is four cycles.

---

**THE NUMBER OF INTERNAL STATES AND THEIR ENCODING**

- Total number of internal states: seven
- Three flip-flops (Q1, Q2, Q3) are necessary and enough for the encoding.
- The actual state encoding greatly influences the complexity and structure of the network.
- Here we use the final (optimal) state encoding.
**STATE ENCODING**

- In the first row, we make use of the redundancy.
- To the states in the same level of the state transition diagram, the same $Q_1$ and $Q_2$ codes are ascribed.
- $Q_1$, $Q_2$: cycle counters.
- $Q_3$: indicates whether the system is in the even or on the odd branch of the state transition diagram.

**STATE FUNCTIONS AND THE OUTPUT FUNCTION**

<table>
<thead>
<tr>
<th>$i$</th>
<th>$n$-edik item</th>
<th>$(n+1)$-edik item</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 a b</td>
<td>1 0 1 x</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 0 1 a b</td>
<td>0 1 0 x</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 1 0 b d</td>
<td>1 1 0 x</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0 1 1 c e</td>
<td>1 1 1 x</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1 0 0 f a</td>
<td>0 0 0 x</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1 0 1 g a</td>
<td>0 0 0 x</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1 1 0 d f</td>
<td>1 0 0 x</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1 1 1 e g</td>
<td>1 0 1 x</td>
<td></td>
</tr>
</tbody>
</table>

**STATE FUNCTIONS AND THE OUTPUT FUNCTION**

$Q_1^{n+1} = \Sigma(2,3,6,7,10,11,14,15)$;

$Q_2^{n+1} = \Sigma(0,3,8,12)$;

$Q_3^{n+1} = \Sigma(3,7,8,9,10); x: (4,5,12,13)$;

$Z^n = \Sigma(5,12); x: (0,3,6,11,14,15)$

**EXCITATION TABLE OF THE JK FLIP-FLOP**

The logic synthesis is based on the excitation table of the flip-flop chosen for the implementation.

<table>
<thead>
<tr>
<th>$Q^n$</th>
<th>$Q^{n+1}$</th>
<th>$J$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

**CONTROL OF FLIP-FLOP $Q_1$**

$K_1 = Q_2$

$J_1 = Q_2$

Note the role of the don’t care terms in the minimization.
CONTROL OF FLIP-FLOP Q_2

\[ K_2 = Q_1 \quad J_2 = \tilde{Q}_1 \]

Due to the proper state-encoding, the X input variable is not present in the control equations of Q_1 and Q_2. These two flip-flops act as cycle counter.

THE OUTPUT FUNCTION Z

Note the chessboard pattern! This implies XOR function:

\[ Z = \tilde{X} Q_3 + X \tilde{Q}_3 = X \oplus Q_3 \]

IMPLEMENTATION ALTERNATIVE USING D FLIP-FLOPS

\[ D_1 = Q_2 \quad D_2 = \tilde{Q}_3 \]

\[ D_1 = x \tilde{Q}_1 + x \tilde{Q}_1 + \tilde{X} Q_3 \]

Due to the "clever" state encoding, the control of the two flip-flops acting as the cycle counter corresponds to the usual one. However, the control network of the third flip-flop is somewhat more complex than in the former implementation.

CONTROL OF FLIP-FLOP Q_3

\[ K_3 = x Q_2 + x \tilde{Q}_2 \quad J_3 = X \]

The X input is among the variables controlling the flip-flop. The state of Q_3 will represent the actual parity. Q_2 will "remember" then parity of the input sequence.

THE LOGIC DIAGRAM OF THE PARITY CHECK CIRCUIT

IMPLEMENTATION USING T FLIP-FLOPS

The feedback network is somewhat more complicated than in the case of D flip-flops. Main reason: Counting in Gray code with T flip-flops needs more gates for the feedback.

Perhaps somebody might check a design with T flip-flops, the cycle counter operating in the simple binary code.
8-BIT PARITY INDICATOR
Generalization to 8 bit s is straightforward.
Design procedure and the state transition diagram is similar.
There will be 15 states, therefore four flip-flops are necessary. If the encoding is the same as previously, then three FFs form the cycle counter, and the fourth will store the information concerning the parity.

SYNCHRONOUS COUNTER
DESIGN EXAMPLE AND CASE STUDY
Consider the synthesis of a 3-bit up-counter in Gray code using JK flip-flops.
Discuss the possibility of using other types of flip-flops as well.

3-BIT GRAY CODE UP-COUNTER
A
Gray code generation using the Karnaugh map
C
B
State transition diagram

STATE TRANSITION
AND FLIP-FLOP EXCITATION TABLES
\[
\begin{array}{c|ccc|ccc|cccc}
 n & A & B & C & n+1 & A & B & C & J\bar{A}/K\bar{A} & J\bar{B}/K\bar{B} & J\bar{C}/K\bar{C} \\
 a & 0 & 0 & 0 & b & 0 & 0 & 1 & 0/X & 0/X & 0/X \\
 b & 0 & 0 & 1 & c & 1 & 0 & 0 & 0/X & 0/X & 0/X \\
 c & 1 & 0 & 0 & a & 0 & 0 & 0 & 1/X & 1/X & 1/X \\
\end{array}
\]

FLIP-FLOP CONTROL EQUATIONS
\[
\begin{align*}
& J_A = B \bar{C} \quad K_A = B \bar{C} \\
& J_B = \bar{A} \quad K_B = A \quad C \\
& J_C = A \bar{B} + \bar{A} \bar{B} \\
& K_C = A \bar{B} + \bar{A} \bar{B}
\end{align*}
\]

OPTIMIZING THE FEEDBACK CIRCUIT
The JA, KA, JB, and KB control functions are implemented by four AND gates in one package.
The JC and KC control function can be implemented in various ways. The simplest one would be using two anti-valency (XOR) gates and one inverter. However in a two-level system the simplest solution is the use of six NAND gates.

\[
\text{Circuit complexity:} \quad \begin{align*}
\text{Gate count:} & \quad 10 \\
\text{Pin count:} & \quad 20
\end{align*}
\]
CIRCUIT LAYOUT: COMPONENT POSITIONING

Note the bus structure!

DESIGN ALTERNATIVES: USING D OR T FLIP-FLOPS

Using D flip-flops, the feedback circuit could be implemented either with 6 AND and 3 OR gates (PLD implementation) or 9 NAND gates using modular SSI gates (pin count 18), or with 3 XOR gates and 1 inverter.

Using T flip-flops is not practical because of the Gray code.

UP/DOWN 3-BIT GRAY CODE COUNTER

State transition diagram

Next-state table

<table>
<thead>
<tr>
<th>Present State</th>
<th>S = 0 (DOWN)</th>
<th>S = 1 (UP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0 D1 D2</td>
<td>D0 D1 D2</td>
<td>D0 D1 D2</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 1 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0 1 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

UP/DOWN 3-BIT GRAY CODE COUNTER

Logic expressions for flip-flop control:

\[ J_0 = Q_0 \overline{Q}_1 Y + Q_0 \overline{Q}_1 \overline{Y} + Q_0 Q_1 \overline{Y} + Q_1 Q_0 Y \]

\[ J_1 = Q_0 Q_1 Y + Q_0 Q_1 \overline{Y} \]

\[ J_2 = Q_0 \overline{Q}_1 Y + \overline{Q}_0 Q_1 \overline{Y} \]

\[ K_0 = Q_0 \overline{Q}_1 \overline{Y} + Q_0 Q_1 Y + Q_0 Q_1 \overline{Y} + Q_0 Q_1 Y \]

\[ K_1 = Q_0 Q_1 \overline{Y} + Q_0 \overline{Q}_1 Y \]

\[ K_2 = Q_0 \overline{Q}_1 \overline{Y} + Q_0 \overline{Q}_1 Y \]
4-BIT BI-DIRECTIONAL
GRAY CODE COUNTER
Features of design provided by one of the students of my previous course.

Compared designs using D or T flip-flops.

Using T flip-flops, some several common terms could be realized by XOR gate or XOR gate and inverter, leading to further simplification of the feedback circuit.

Complexity: 16 NAND gates (2,3 or 4 inputs), 2 XOR gates and 2 inverters.

Estimated the maximum clock frequency of the counter when using high speed CMOS logic components.