Outline

- C Code ➔ HLSM
- Midterm 1
- Midterm 2
- Practice Problems
C Code ➔ HW

- General flow for HLSM Problems
C code to HLSM

Consider the following code. Create HLSM, datapath design and connection between datapath and controller.

```c
void main() {
    unsigned int i, start, data, mod4_count, A[128];
    while(1) {
        while(!start); i=0; mod4_count = 0;
        while(i<128) {
            if(A[i]%4==0) {
                mod4_count++;
            }
            i++;
        }
    }
}
```
HLSM : while(!start);
HLSM \( (i=0, \text{mod4}\_\text{cnt}=0) \)
HLSM (i<128)
HLSM \( (A[i] \mod 4 == 0) \)
HLSM (Increment i and mod4_cnt)

- **A** to **B**: start, with conditions (i<128)'
- **B** to **C**: i=0, mod4_cnt=0
- **C** to **B**: (i<128)
- **B** to **D**: i:=i+1
- **D** to **E**: (A[i] %4==0)
- **E** to **F**: mod4_cnt:=mod4_cnt+1
Datapath
Connection between datapath and controller
Midterm 1 & 2 Review
Boolean Algebra

Simplify the following equation using boolean algebra and implement the simplified equation using

i) NAND/ inverters

ii) NOR/ inverters.

\[(x+y')(xy + xz + yz)(x'y) + xyz + (xz)']\]
Solution

Expression : \( x' + y + z' \)

NAND/ inverter implementation : \( ((x.y')''.z)' \)

NOR/ inverter implementation : \( ((x' + y)'' + z')'' \)
Combined problem

Given \( f(a, b, c, d) = \Sigma m(1, 2, 5, 7, 8, 12, 14) + \Sigma d(3, 6) \)

a) Fill out the kmap
b) Minimum SOP expression
c) Minimum POS expression
d) Implementation using 8:1 MUX and inverters, 4:1 MUX and other gates.
e) Implementation using 3:8 & 2:4 Decoder + other gates.
Solution

Minimum SOP: \( a'd + a'c + bcd' + ac'd' \)

Minimum POS: \((a+c+d).(a'+d').(a'+b+c')\)
## Mux Implementation

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<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>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Mux Implementation (8:1 + inverters)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<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>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

MUX inputs
000 → d
001 → 1 / d’
010 → d
011 → 1 / d
100 → d’
101 → 0
110 → d’
111 → d’
Mux Implementation (4:1 + other gates)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<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>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>1</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>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

MUX inputs

00 → c XOR d / c OR d

01 → d / c OR d

10 → c NOR d

11 → d'
Demux implementation

Implementation similar to HW3 Q2. Just connect the required outputs to the OR gate.

Link: HW3_Solution
Multiplexers

Partially-specified schematic below implements the function \( F = A'C + B'D + CD \). Without adding any gates to the circuit, complete the implementation by identifying the unspecified inputs. (Hint: the inputs can only be \( A, B, C, D, 0 \) or \( 1 \)).
Solution
Circuit analysis

Give expression for \( \text{out}(r, x) \), \( r(z, y, b, c, d) \), \( r(a, b, c, d) \), \( \text{out}(a, b, c, d) \)

Solution : \( \text{out}(r, x) = (r'x')(r.x') = 0 \)

\[
r(z, y, b, c, d) = b'd'z + b'dc + bd'y
\]

\[
r(a, b, c, d) = b'd'(a.b) + b'dc + bd'(a'+b).d)
\]

\( \text{out}(a, b, c, d) = 0 \)
Latches and FFs

- **Sequential circuit**: depends on past outputs and current inputs
- **SR Latch**
  - Level sensitive
  - 11 at input is INVALID for NOR SR Latch
- **D Latch**
  - Inverter between S and R inputs
  - Avoids the INVALID state of SR Latch
- **D-Flip Flop**
  - Edge sensitive
  - Output changes only at transition from 0 → 1 (rising edge) or 1 → 0 (falling edge)
FSM

- Mathematical model
- Changes states based on inputs → transition
- Example: Vending machine, elevators, traffic lights and many more…….
- Flow for FSM Problems
Pattern Recognition FSM

3 bit pattern recognizer. Detect pattern “101” (repeating patterns allowed)

Example:

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

(a) Draw the state diagram for the FSM.

(b) Provide an excitation table. Use binary encoding for the states, so state S0=00, S1=01, S2=10 etc.

(c) Implement the circuit for the MSB of the next state logic.
Solution

FSM: MEALY
Solution

FSM: MOORE
Waveforms (Latch and FF)

Draw the waveforms for W, X, Y, Z and OUT, given waveforms for A and B.
Solution
Timing analysis

\[
\begin{array}{|c|c|c|}
\hline
\text{Gate} & \text{Tpd (ps)} & \text{Tcd (ps)} \\
\hline
\text{NAND} & 15 & 10 \\
\text{NOR} & 25 & 15 \\
\text{XOR} & 35 & 25 \\
\text{NOT} & 10 & 5 \\
\hline
\end{array}
\]

Consider the following circuit

(a) Maximum clock frequency.

(b) Is there a hold time violation?
Solution

a) Max_delay path = C ➔ X

Tc >= tsetup + tpcq + tpd

Tc >= 15ps + 15ps + (35+10+25)ps

Tc >= 100ps

fc <= 10 GHz
Solution

b) Min_delay path = A→ X

\[ \text{thold} < \text{tccq} + \text{tcd} \]

\[ \text{thold} < 10\text{ps} + (10+15)\text{ps} \]

10 ps < 35ps which is true

There is NO hold time violation
Practice Problem (HW6)

Use RTL design process to create the following system. The system receives a 4 bit number \( A \) as an input whenever signal \( u = '1' \). The 8-bit output \( S \) is the sum of all the input numbers up to now. Input \( r = '1' \) is used to reset \( S \) to 0. You may assume that overflow does not occur.

a. Draw HLSM for the system. Hint: you only need 3 states to capture the system behavior.

b. Design the system datapath. Clearly show the signals that need to be generated by controller.

c. Draw the Moore FSM corresponding to the controller.

d. Complete the excitation table for the system. Use minimum bit-change state assignment strategy to encode states. Use code 00 for the init state, and code 01 for wait (idle) state.

e. Implement the controller FSM using flipflops and minimum number of 2-input gates to generate the required control signals.

f. Assuming all gates (regardless of number of inputs) have 2ns delay and we have used ripple-carry adder in datapath, find the critical path for the system. (Wires have no delay)

g. Calculate the maximum clock frequency for the circuit.

Flipflop timing characteristics:
\( t_{setup} = t_{hold} = 6\text{ns} \)
\( t_{ccq} = 8\text{ns} \), \( t_{pcq} = 12\text{ns} \)
HLSM

Init:
S:=0

Wait

u = '0'

u = '1'

Input:
S:=S+A

r = '1'
Datapath
Controller FSM

- **Init:**
  - clr_S=1
  - load_S=0

- **Wait:**
  - clr_S=0
  - load_S=0
  - u = '0'

- **Input:**
  - clr_S=0
  - load_S=1
  - u = '1'

- r = '1'