There are three 10-element 4-bit arrays, A[0:9], B[0:9], and C[0:9]. There are two more 10-element 1-bit arrays, AV[0:9] and BV[0:9] (V = Valid). If (AV[I] = 0), then A[I] is not valid. If (BV[J] = 0), then B[J] is not valid. Pick the next two valid A[I] and B[J], add them, and place the 4-bit sum in C[K]. Notice that the I, J, and K may all be the same or can be completely different depending on the distribution of invalid items in A and B arrays. If the first element valid in A is A[8] and the first element valid in B is B[2], then you end up doing C[0] <= A[8] + B[2]; One valid element of A (or B) can be used only once. Hence one of the two arrays A and B, can be exhausted more quickly and that should terminate the addition process. So, the number of sums gathered in C may be zero if all elements in A or B (or both) are all invalid. The DONE_NF (Done Not Found) state shall be reached if not a single sum is loaded in C.

Implement it using a Mealy machine and a Moore machine separately on the next two pages. Use shorthand labels for conditions (C) and statements (S) to reduce the clutter on your diagrams. I have made up a few shorthand labels and listed them on the next page. You can define some more such labels as needed. Both C’s and S’s can be used in the Mealy design. Only C’s can be used in the Moore design as all S’s are conditional statements (inappropriate for the Moore).

Notice that you would increment both I and J if both AV[I] and BV[J] indicate that both are valid or both are invalid. But if one is valid and the other is invalid, you want to increment the index of the invalid array in search of the next valid number. So, is it possible for I or J to reach 9 before the other? Yes, of course! But is it possible that one reaches 9 before the other waits for the other? Well consider the following example arrays. After doing C[0] <= A[8] + B[2]; I becomes 9 but has to wait for J to come to 5. Then you do C[1] <= A[9] + B[5]; and exit.

In preparation to your Mealy machine design, please answer the following:

Do you exit from the COMP (compute) state if

(A) (I == 9) & (J == 9) Y / N
(B) (I == 9) & (AV[I] == 0) Y / N
(C) (J == 9) & (BV[J] == 0) Y / N
(D) (I == 9) & (AV[I] == 1) & (J == 7) & (BV[J] == 0) Y / N
(E) (I == 9) & (AV[I] == 1) & (J == 7) & (BV[J] == 1) Y / N

Clocks spent in the COMP state of the Mealy machine
Number of Clocks spent for the above Data: _______
Smallest number of clocks spent for any Data _______
Largest number of clocks spent for any Data _______

Clocks spent in the middle 5 states of the Moore machine
Number of Clocks spent for the above Data: _______
Smallest number of clocks spent for any Data _______
Largest number of clocks spent for any Data _______

Note: All most all operations take two clocks in Moore compared to one clock in Mealy except the final operation if it goes from CHECK state to one of the DONE states directly.
Mealy machine implementation: Some shorthand labels

1.1

\[ C_1 = (I == 9) \land (J == 9); \]
\[ C_2 = (I == 9) \land (AV[I] == 0); \]
\[ C_3 = (J == 9) \land (BV[J] == 0); \]
\( (C_2 \lor C_3) \leq \text{what does this tell you?} \)
\[ C_4 = (I == 9) \land (AV[I] == 1) \land (J != 9) \land (BV[J] == 0); \]
\[ C_5 = (I != 9) \land (AV[I] == 0) \land (J == 9) \land (BV[J] == 1); \]
\[ C_6A = ( (I != 9) \land (J != 9) ) \land (AV[I] == 0) \land (BV[J] == 0); \]
\[ C_6B = ( (I != 9) ) \land (AV[I] == 0) \land (BV[J] == 1); \]
\[ C_6C = ( (J != 9) ) \land (AV[I] == 1) \land (BV[J] == 0); \]
\[ C_7 = (AV[I] == 1) \land (BV[J] == 1); \text{ Both are valid, Hurray} \]
\[ C_8A = ( (I == 9) \lor (J == 9) ) ; \text{ See if it is useful in the Moore design} \]
\[ C_8B = ( (I == 9) \lor (J == 9) ) \land (AV[I] == 1) \land (BV[J] == 1); \]
\[ = \text{ Both are valid and at least one of them is at the bottom of its array} \]

\[ S_1 \Rightarrow C[K] \leq A[I] + B[J]; \]
\[ S_2 \Rightarrow I \leq I + 1; \]
\[ S_3 \Rightarrow J \leq J + 1; \]
\[ S_4 \Rightarrow K \leq K + 1; \]

An acceptable RTL statement: if (C6) S1;

You can write this condition in terms of the other two conditions

\[ \text{ACK} \]
\[ \text{ACK} \]

\[ \text{ACK} \]
\[ \text{ACK} \]
1.2 Moore machine implementation, Design plan discussion: Of course we need to split the COMP state of the Mealy machine into several Moore states. In the COMP state of the mealy, we may be incrementing either I or J or both. When we increment both, it may be because both A and B are invalid or both are valid. And if both are valid, we want to deposit into C[K] and increment K also. So, in the following Moore machine, we have a CHECK state to check AV[I] and BV[J] and decide which of the 6 actions to take as shown by the six exit arrows exiting from the CHECK state. In the Mealy machine, we may be incrementing I and/or J, and simultaneously preparing to exit to DONE_F or DONE_NF. However, in Moore since it costs extra clocks and/or states, our actions are rather measured. We will go to another state to increment I and/or J only if we were to return to CHECK state to check some more elements of A and/or B. So the return to the CHECK state from these three states incrementing I and/or J is unconditional. If we go to the DEP_C (Deposit into C) state, if the A[I] and/or B[J], involved in producing the sum, happens to be the last in their respective arrays, then you can exit to DONE_F state, otherwise return to the CHECK state. Based on this discussion, please complete the state transition conditions below. All the states and state transitions are in place. Also RTL statements in the states are already complete! Try to use some of the shorthand labels defined on the previous page.
2 (3+3+4+16+5+8+10+6+6 = 61 points) 40 min. Timing

2.1 An inverter gate has two delays: tpHL and tpLH.

(i) tpHL and tpLH
(ii) tpLH and tpHL

2.2 In the context of Timing Design, STA stands for STATIC_TIMING_ANALYZER (or ANALYSER).

2.3 Shannon’s theorem helps to reduce __________ delay by ________ (combinational logic sequential logic).
   A = arranging circuits in a tree fashion instead of linear cascade
   B = by producing multiple results of a combinational circuit for different possible values of the late-arriving signals and finally selecting one of those results when the late-arriving signals do arrive.

2.3.1 Complete the following two choices for implementing the datapath to perform (X+Y) or (X-Y) to produce nZ (next Z) in the state COMPUTE below.

Z is assigned only in the COMPUTE state in the state machine containing 10 states. Complete the data-enable control on data-register Z. One-hot state assignment is used. You can use Q_{COMPUTE} besides any other control signal such as (P>Q) in controlling the DE (Data-Enable).

Choice _____ (#1/#2) is faster.
Choice _____ (#1/#2) is more expensive.
2.4 A Flip-Flop has three main timing specs: setup time tsu, hold time th and propagation delay tfppd. Which of these are likely to have minimum, typical, maximum values in the spec sheet? What values others have? __________ tfppd has three values minimum, typical, maximum __________

2.5 The following timing equations are incomplete. Also there was a typo. By mistake someone typed an extra tclk. The parentheses shall be filled with (min) or (max). Correct them and complete them.

\[
\text{Setup Margin} = \text{tclk} - \text{tfppd}(\max) - \text{tcomb}(\max) - \text{tsetup}(\min)
\]

\[
\text{Hold Margin} = \text{tclk} + \text{tfppd}(\min) + \text{tcomb}(\min) - \text{thold}(\min)
\]

2.5.1 A 16-bit counter was built using a 16-bit register and a 16-bit ripple carry adder (built by cascading 16 half-adders) as shown on the side. Three students are arguing.

\[\times\] S1: The counter may at most have setup time problems.

\[\times\] S2: The counter may at most have hold time problems. __________

\[\checkmark\] S3: The counter may have setup time and hold time problems simultaneously. Right.

Cite an example of shortest path and an example of longest path for the counter and help those three student resolve their argument. \(Q_0\) to \(Q_3\) (example \(Q_0\) to \(Q_3\)) is an example of a shortest path. There are 16 such equally short paths! There is only one longest path \(Q_0\) to \(Q_{15}\). The shortest path has one XOR gate. By substituting the min delay of the XOR gate for \(tcomb(\min)\) in the above hold time margin equation, we can calculate the margin and if it turns out to be negative, we have hold time violation. The longest path has 15 AND gates and an XOR gate. The maximum cumulative delay of these 16 gates can be substituted for \(tcomb(\max)\). If the setup margin comes out to be negative, then you have hit setup time violation.

2.6 While human interaction with the machine is a clearly an appealing example of an asynchronous input, asynchronous inputs occur very frequently (several millions of them per second) on a motherboard of a computer. How? And what is done to avoid meta stability problem.

2.7 What is this art work below called "H Tree"? What happens if we do not use something like that?

This is called a CLOCK TREE or a CLOCK DISTRIBUTION NETWORK. Clock arrives at the center of the chip and follows the paths which look like a series of "H" to arrive at each register on the chip. We note that distance travelled from the center of the chip to the clock input of any register is the same irrespective of the location of the register on the chip. Hence the relative skew are minimal. Clock tree reduces clock skew (difference in the arrival times of clock) which very important for successful simultaneous (coincident) RTL operations on the chip. A simple example is a swap operation. A \(\leftrightarrow\) B which not work if there is a lot of clock skew.
3 (6 + 10 + 10 + 16 + 4 = 46 points) 30 min.

FIFOs

3.1 A change in depth of the FIFO _______ (will/will not) cause a change in the pin-out of the FIFO. A change in width of the FIFO _______ (will/will not) cause a change in the pin-out of the FIFO. Explain: The WP and RP pointer counters will change in size when the depth of the FIFO is changed. However, they are internal to the FIFO and will not cause a change in the pinout. WP in and RD out are for write data in and read data out. The FIFO PINOUT never changes when FIFO width is changed, these will change accordingly.

3.2 WP is always _______ ahead of/behind) RP except in one occasion, namely when the FIFO is empty.

WP always points to ____________ (an empty/a filled) location in the FIFO except in one occasion, namely when the FIFO is full.

RP always points to ____________ (an empty/a filled) location in the FIFO except in one occasion, namely when the FIFO is empty.

3.3 Never ever synchronize by sampling and holding (circle all applicable):
(a) a multi-bit data item where one bit at most changes at any time
(b) a multi-bit data item where multiple bits could be changing simultaneously

What happens if we do so? If you happen to sample when multiple bits are changing, all those bits can potentially go into meta stable state and some can fall to their old value, and some to their new value. This leads to absurd values of the sample which is neither new nor old.

Do you take WP_binary or WP_Gray from the WCLK domain to RCLK domain? WP-Grey

Why not the other? Well, multiple bits change simultaneously in WP_binary, in a 4-bit WP 0111 => 1000 change involves all your bits.

3.4 Complete the missing portion a 4-bit Gray code sequence on the side.
You know the following conversions between 4-bit binary and 4-bit gray codes. Add one more line (and relabel the bits if needed) to extend these to 5-bit conversions.

<table>
<thead>
<tr>
<th>G3</th>
<th>G2</th>
<th>G1</th>
<th>G0</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>0</td>
<td>0</td>
</tr>
<tr>
<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>
</tr>
</tbody>
</table>

The 8 bit Gray code number 1111_1111 is an ________ (even/odd) number. The largest valued 8-bit Gray code number is 1000_0000.

3.5 If you double the depth of a FIFO between a producer and a consumer, you need to (circle all applicable) (a) redesign the producer, (b) redesign the consumer (c) no need to redesign any

A ____________ 2-clock (single-clock) FIFO works as a clock-domain crossing mechanism.
4 (12 + 10 + 20 = 42 points) 25 min. Memory

4.1 Memory map reading and interpreting: State the size and range of the shaded area in the map on the side. Assume that there is a RAM memory chip occupying that area and generate a low active chip-select signal CS when an address appears on A19-A0 which falls in the shaded area. Label the address pins and complete the address connections to the RAM chip below.

Size: \( 64K \) (\( 64 \) kB)

Range: \( 0000 \) - \( FFFF \)

\( q_1^\text{data} = 1001_2 \)

\( A19 \) - A0

\( A15 \) - A0

\( MEMR \)

\( MEMW \)

\( WE \)

\( CS \)

\( D7 - D0 \)

4.2 Memory width and depth expansion: We needed a 64Kx8 ROM memory using the 4 smaller ROM chips. Miss Bruin was not taught about natural boundaries of address ranges and has arranged the four chips as shown on the side. Since ROMs were already programmed with the needed content, we can not change the map. So manage to complete the design below to suit.

The 64Kx8 ROM has 16 address pins and 8 data pins.

Complete the design below. Add missing labels, wires, and gates.

An 64K range in hex is from 0000 Hex to \( FFFF \) hex.

Tell Miss Bruin the range of address implied by 32KB placement in the map: 4000 H to \( BFFF \) H (sum of the two natural 16K ranges 4000-7FFF and 8000-BFFF).

32KB-CE

Another design using simple gates and no decoder.

16KB-Top

16KB-BOT

32KB

16KB-Top

32KB

16KB-BOT

32KB
5  \[ 8 + 8 = 16 \]  points) 10 min. Schematic => Verilog Coding

Given below is a small schematic and the associated Verilog code containing two assign statements and three always clocked blocks. Rewrite the code without any assign statements in two ways: Method #1: one clocked always block Method #2: Q1 and Q2 generated in one always clocked block and Q3 generated in a separate clocked block. Do not worry about the data types of signals (wire or reg). Let us assume that they are adjusted accordingly.

module Inputs/Outputs

<table>
<thead>
<tr>
<th>D1</th>
<th>Q1</th>
</tr>
</thead>
<tbody>
<tr>
<td>D2</td>
<td>Q2</td>
</tr>
<tr>
<td>D3</td>
<td>Q3</td>
</tr>
</tbody>
</table>

EN and D3_Int are internal signals (not module port pins)

```verilog
always @(posedge CLK)
begin
    Q1 <= D1;
end

always @(posedge CLK)
begin
    Q2 <= D2;
end

assign EN = Q1 & Q2 & X;

assign D3_Int = EN ? D3 : Q3;

always @(posedge CLK)
begin
    Q3 <= D3_Int;
end
```

Method #1: one clocked always block

```verilog
always @(posedge CLK)
begin
    Q1 <= D1;
    Q2 <= D2;
    Q3 <= D3;
end
```

It is not desirable to do low-level coding of the AND gate and the mux by producing the unneeded intermediate signals, EN and D3_Int. But if they are produced, they should be produced using Blocking assignment operator. The three Q's shall always be assigned using non-blocking assignment operator.

Method #2: Q1 and Q2 in one always block Q3 in another

```verilog
always @(posedge CLK)
begin
    Q1 <= D1;
    Q2 <= D2;
end

always @(posedge CLK)
begin
    Q3 <= D3;
end
```

It is important that \( X \& Q1 \& Q2 \), which is the upstream logic of Q3, stays in the always block producing Q3. It is wrong if students put it in the other always block.

We enjoyed teaching this course. Several of you made very interesting projects. Hope to see you in EE457. - Gandhi