1) For the function $F(A,B,C,D) = \Sigma m(3,4,9,11,12)$ draw the K-map, and write a minimal sum of products expression for $F$.

\[
F = \overline{B} \overline{C} \overline{D} + A \overline{B} D + \overline{B} \overline{C} D
\]

-1 not simplified
-2 simple mistake in K-map with consistent formula
-3 2+ errors
-4 very wrong

2) Implement $G=AB+BC+CD$ using 2-input and/or 3-input NAND gates only.

\[
G = X \overline{Y} \overline{Z}
\]

-1 point it very complex
-3 it wrong but close

3) Implement the function $G$ above using the PLA below

(almost every error -2 for wrong crosses)
4) Due to a design error in your Checkpoint1 UART transmitter, the serial output line glitches low for 9 microseconds. What effect does this have on the receiver?
   a. Nothing  
   b. Generates an error if implemented properly  
   c. Receives the following character:  
   d. Receives a sequence of characters. Describe:  
   e. unknown

5) Same as the previous question, but the UART glitches low for 1 second.

6) Your partner gives you a Verilog block with a ready/valid interface on its output. The output contains the value 0x43 on the rising edge of clock cycles one through five, and 0x88 on the rising edge of clock cycles six through ten. The valid output is high on the rising edge of cycles three through seven inclusive.
   a. If the other side of the interface is operating correctly, and asserts ready continuously for all ten cycles, what values are transferred across the interface?
   b. Same as in part a) above, but ready is asserted only on falling edges, never on rising edges.
   c. Same as in part a) above, but ready is only asserted on the rising edges of cycles one through four.

7) In the context of the project checkpoint 1 memory map,
   a. What is the 4 bit write enable signal to data memory for storing a value to the address 0x10000005?
   b. What is the write mask into data memory for 0x20000008?
8) Write Verilog to implement the FSM below.

```verilog
module FSM(input clk, input rst, input in, output out);

localparam 0 = Fred;
1 = Sue,
2 = Bob;

reg [1:0] cs;
reg [1:0] ns;

always @(posedge clk) begin
if (rst) cs <= Bob;
else cs <= ns;
end

always @(*) begin
  case(cs)
    Fred: ns = in ? Bob : Sue;
    Sue: ns = in ? Fred : Bob;
    Bob: ns = Sue;
    default: ns = Bob;
  endcase
end

assign out = (cs == Sue);
endmodule
```

KEY

- Correct port declaration: +1
- Used some sort of encoding to map states to values: +1
- State registers w/ correct widths: +1
- Synchronous logic for current state: +6
- Correct reset condition: +2
- Asynchronous logic for next state: +4
- No default case: -1
- Output assigned based on state: +3
9) In a single-cycle MIPS processor you need to implement a new i-type instruction, **shia** – store half-word immediate and add. The sign-extended immediate is stored to the memory address contained in rt, and rt is incremented by r5.

\[ M([rt]) = \text{Sign} \text{mm} \ ; \ [rt] = [rs] + [rt] \]

a) Describe in words how you need to modify if the datapath, and what control signals you need to add, and the values of all control signals when executing this new instruction.

You want the ALU to sum [r7] and [r5], but you no longer want the output of the ALU to always be the address to DRAM; add a mux so you can select just the value [r+7] to use as the address. You also want to write the sign extended immediate to memory, so add another mux.

New control signals and values:

- \( shia = 1 \)
- \( t2 = -1 \) per extra control
- \( t1 = +1 \)

Control signals values:
- \( \text{MemtoReg} = 0 +1 \)
- \( \text{MemWrite} = 1 \)
- \( \text{Branch} = 0 +1 \)
- \( \text{ALUControl} = \text{Add} +1 \)
- \( \text{ALUSrc} = 0 +1 \)
- \( \text{RegDst} = 0 +1 \)
- \( \text{RegWrite} = 1 +1 \)

b) Draw any modifications that you need to make to the datapath below the figure, labeling any new control signals. You don’t need to redraw the parts that you don’t touch, just the parts that you change.

+2 per extra hardware
-2 for not forwarding correct ALU result
-2 for storing wrong value
-2 for using register value as rt

+1 for the value of each control signal
+5 for each mux added to the datapath correctly
+2 for the correct value of the added control signal
+6 for a good description for how to modify the datapath

[7 total]
[10 total]
[2 total]
[6 total]
KEY

MUX input to select between SREB, Addr, Sign, Imm, etc.

SrcB goes into MUX for 2 + 2

All result by passes address MUX + 2

-2 for not telling how you need to modify

+1 if recognizing a MUX to input

-1 if signals for MUX not specified

Each extra unnecessary hardware
10) In a new technology, you have implemented a single-cycle MIPS processor. You find that the delay for the instruction and data memories is 1 ns, the ALU is 200 ps, register file access is 100 ps, and all other logic such as PC increment logic and muxes are so fast that they can be ignored.

2. a. What is the minimum cycle time for the single-cycle processor?

\[ t_{im} + t_{RF} + t_{ALU} + t_{dm} = 1 \text{ ns} + 0.1 \text{ ns} + 0.2 \text{ ns} + 1 \text{ ns} = 2.3 \text{ ns} \]

2. b. If you convert this to a 3 stage pipeline, which stage of the pipeline should the register file go in to minimize the cycle time?

\[ \text{not with I-mem or D-mem, so stage 2, Execute, ALU} \]

2. c. What is the minimum cycle time for the 3 stage pipeline version, assuming no pipeline register overhead?

\[ \max \left\{ t_{im}, t_{RF} + t_{ALU}, t_{dm} \right\} = 1 \text{ ns} \]

2. d. If the SRAMs could each be broken into two stages with an internal pipeline register, “address decode” and “array access”, each 500 ps long, what is the right number of stages to minimize delay, and the minimum cycle time?

5 stages, 500 ps

2. e. Assume a 5-stage pipeline using the SRAMs from part d), where the instruction memory is accessed in stages Fetch1 and Fetch2, X is the execution stage with both register file and ALU, the data memory is accessed in D1 and D2, and register file write-back happens on the rising edge at the end of the D2 stage. The following assembly sequence is executed:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
---|---|---|---|---|---|---|---|
LW $s0, 0($s4) | F1 | F2 | X | D1 | D2 |
ORI $t0,$s0, $s1 | F1 | F2 | X | D1 | D2 |
ORI $t1,$s0, $s2 | F1 | F2 | X | D1 | D2 |
ORI $t2,$t1, $s3 | F1 | F2 | X | D1 | D2 |
ORI $t3,$s0, $t1 | F1 | F2 | X | D1 | D1 |

2. When is the first time that the result of the LW is available in the datapath?

Rising edge, early, middle, or end of cycle 4

2. When is the value written into $s0$?

Rising edge, early, middle, or end of cycle 5

When is the value of $s0$ needed assuming no load delay slot is used?

Rising edge, early, middle, or end of cycle 3

Can this hazard be solved with forwarding? If so, how?

No.

2. When is the first time that the result of ORI $t1$, $s0$, $s2$ is available in the datapath?

Rising edge, early, middle, or end of cycle 4

2. When is the value written into $t1$?

Rising edge, early, middle, or end of cycle 7

When is the value of $t1$ needed?

Rising edge, early, middle, or end of cycle 5

Can this hazard be solved with forwarding? If so, how?

Yes. Forward ALUout D1