Q1. (15 pts) We have a 10-bit number. We need to add all the bits. Draw the circuit using a Carry Save Tree logic. Then write the Verilog code by instantiating full-adders (module $fa(a,b,cin, cout, sum)$) and a final 3-bit carry-looakahead adder (module $cla(a, b, sum)$).

```
module csa_sum10(B9,B8,B7, B6,B5,B4,B3,B2,B1,B0, S3, S2, S1, S0);
    input  B9,B8,B7, B6,B5,B4,B3,B2,B1;B0;
    output S3, S2, S1, S0;
    fa FA0 (B8,B7,B6,cout0, sum0);
    fa FA1 (B5,B4,B3,cout1, sum1);
    fa FA2 (B2,B1,B0,cout2, sum2);
    fa FA3 (sum0,sum1,sum2,cout3, sum3);
    fa FA4 (cout0,cout1,cout2,cout4, sum4);
    cla CLA0 ({1'b0,cout3,sum3}, {cout4,sum4, B9}, {S3,S2,S1,S0});
endmodule
```

Q2. (10 pts) Draw the CMOS transistor circuit for $\overline{(A\overline{B} + (A + C)B)}$.

Q3. (10 pts) Write an always block which is equivalent to
assign a = b ? c : {c[0], c[1], c[2], c[3]};

always @ * begin
    if (b)
        a = c;
    else
        a = c[0:3];
end

Q4. (15 pts) Write a module which describes a flop which latches in a new value once every two cycles.

always @(posedge clk) begin
    toggle<=~toggle;
    q<=(toggle) ? d: q;
end
or
always @(posedge clk) begin
    toggle<=~toggle;
    if (toggle)
        q<=d;
end

Q5. (15 pts) Design a circuit that generates a 20MHz clock with 40% duty cycle from 100MHz clock input. Write the Verilog code for it.

module q5 (clk100M, clk20M);
    input clk100M;
    output clk20M;
    reg [2:0] state;
    always @(posedge clk100M) begin
        state <= 3'bxxx;
        case (state)
            3'b000: state<=3'b001;
            3'b001: state<=3'b010;
            3'b010: state<=3'b011;
            3'b011: state<=3'b100;
            3'b100: state<=3'b000;
        endcase
    end
    assign clk20M = state[1];
endmodule

Q6. (10 pts) Consider the circuit below. The gate delays, clock_to_Q delay and setup time of a flop are given below. All delays are assumed to be constant. Ignore clock issues such as skew and jitter. If the below circuit barely meets timing for a 2GHz clock, what is the delay of the inverter?

\[ T_{CL}=60\text{ps}, \ T_{AND}=120\text{ps}, \ T_{NAND}=75\text{ps}, \ T_{OR}=125\text{ps}, \ T_{\text{setup}}=60\text{ps} \]
1/2GHz = 0.5 ns = 500ps.

500ps = 60 + Tinv + 120 + 75 + 125 + 60 \rightarrow Tinv = 60ps.

Q7. (25 pts) We intend to compute \( x = (((a+b) + (c+d)) + (e+f)) + (g+h) + (i+j) \). You have adders at your disposal. An adder takes 150ps. Assume Tsetup = 15ps, Thold = 10ps, Tcq = 15ps, Tskew = Tjitter = 5ps.

a. Draw DFG.

b. What is the minimum time you can compute \( x \)?

\[ Tc = 5 \times 150 + 15 + 10 + 15 + 5 + 5 = 750 + 50 = 800 \text{ps} \]

c. How many ps will it take to compute \( x \) 100 times?

\[ 800 \times 100 = 80 \text{ns} \]

d. If I pipeline the computation so that every cycle a new computation can be started, how many ps will it take to perform the same computation 100 times?

\[ Tc = 150 + 15 + 10 + 15 + 5 + 5 = 200 \text{ps} \]

\[ 5 \times 200 + 99 \times 200 = 1000 + 19800 = 20800 \text{ps} = 20.8 \text{ns} \]
e. Now suppose you only have 3 adders. Come up with an ASAP scheduling in minimum cycles.

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>Cycle#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1+</td>
<td>2+</td>
<td>3+</td>
<td>1</td>
</tr>
<tr>
<td>6+</td>
<td>4+</td>
<td>5+</td>
<td>2</td>
</tr>
<tr>
<td>7+</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>8+</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>9+</td>
<td></td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>

GOOD LUCK😊