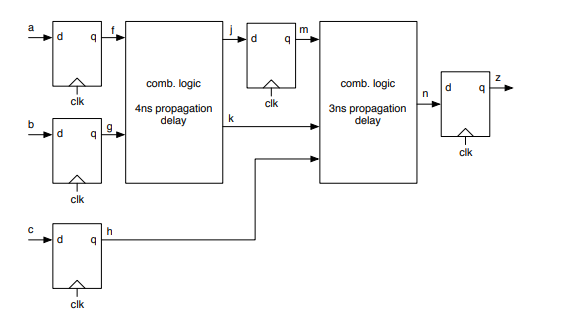
**PROJECT 1**

**Part1:**



Shortest possible clock period,

Tclk = propagation delay of the input register + critical path delay for the combinational logic + set-up time of the output register.

Tclk = 4ns + (4ns + 3ns) + 3ns

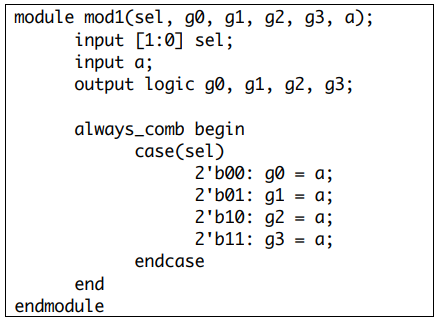
**Tclk = 14ns**.

Fastest possible clock frequency**, Fmax= (1/14) GHz = 71.4MHz**

Fot 1 ns hold time, there is no hold time violation because all the signals will be stable for 1ns after the clock edge.

However, if the flip-flops' hold time is 10ns, and input 'c' toggles, causing a change in signal 'n' after 7ns (4ns + 3ns), the circuit has a hold time violation because 'n' is not stable for 10ns but changes in 7ns.

To fix this problem, there should be more delay added between c to n(3ns-4ns) and a to j(2ns-3ns), for all the signals to be stable for 10ns after clock is triggered.



The above code cannot be synthesized because of a latch caused by the case statement. In each case, only one of the outputs has a value assigned, while the others are all unassigned which will be an inferred latch scenario.

With the following change added to the case statement, the code can be synthesized:

module mod1(sel, g0, g1, g2, g3, a);

input [1:0] sel;

input a;

output logic g0, g1, g2, g3;

always\_comb begin

case(sel)

2'b00: g0 = a; g1 = 0; g2 = 0; g3 = 0;

2'b01: g1 = a; g0 = 0; g2 = 0; g3 = 0;

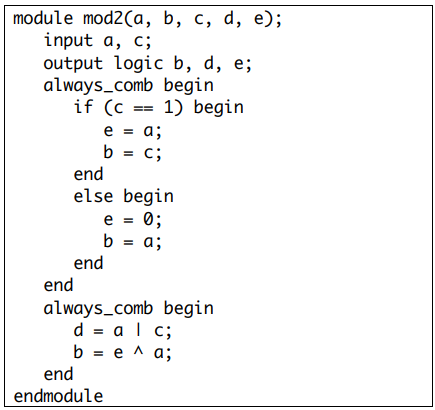
2'b10: g2 = a; g0 = 0; g1 = 0; g3 = 0;

2'b11: g3 = a; g0 = 0; g1 = 0; g2 = 0;

endcase

end

endmodule



The above code cannot be synthesized because the output b is driven by two different inputs at the same time. The code can be synthesized with the below change:

module mod2(a, b, c, d, e);

input a, c;

output logic b, d, e;

always\_comb begin

if (c == 1) begin

e = a;

end

else begin

e = 0;

end

end

always\_comb begin

d = a | c;

b = e ^ a;

end

endmodule

**PART2:**

Path for part2\_mac.sv:

Path for the testbench:

Path for C code used to generate inputData for testbench:

1. Reset is set to 1 at the beginning. Then, the inputs a, b, valid\_in and reset are generated randomly using a C program. 100000 inputs and its expected outputs are generated. The generated random inputs are fed into the design. Then we check if the generated output matches with the expected output. a & b are randomized between -8162 to 8161, and the probability of reset getting triggered is reduced in the code to make the inputs close to real-time inputs and to generate instances for overflow/underflow to occur.
2. The testbench mentioned above has multiple instances where accumulator overflows. When two inputs are positive and the output is a negative number, this implies that there is an overflow. When two inputs are negative and the output is a positive number, this implies that there is an overflow. In order to detect when overflow happened, we check the MSB of a\*b, previous value of f(f\_prev), and the current value of f(f). If MSB of a\*b and f\_prev are 0 and f is 1, it means that there is an overflow occurred. Similarly, If MSB of a\*b and f\_prev are 1 and f is 0, it means that there is an overflow occurred.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| frequency | Area(um2) | Power(uW) | critical path | Critical path |
| 500MHz | 1352.61 | 414.7782 | Startpoint: a\_r\_reg [3] (rising edge-triggered flip-flop clocked by clk)  Endpoint: f\_reg [27] (rising edge-triggered flip-flop clocked by clk) | 3rd bit output of input register a to 27th bit input of the o/p register f |
| 625MHz | 1405.80991 | 507.25 | Startpoint: b\_r\_reg [9] (rising edge-triggered flip-flop clocked by clk)  Endpoint: f\_reg [27] (rising edge-triggered flip-flop clocked by clk) | 9th bit output of input register b to 27th bit input of the o/p register f |
| 666.67MHz | 1432.941991 | 546.7825 | Startpoint: b\_r\_reg[13](rising edge-triggered flip-flop clocked by clk)  Endpoint: f\_reg[25] (rising edge-triggered flip-flop clocked by clk) | 13th bit output of input register b to 25th bit input of the o/p register f |
| 769.23MHz | 1463.265991 | 631.8738 | Startpoint: b\_r\_reg [1] (rising edge-triggered flip-flop clocked by clk)  Endpoint: f\_reg [27] (rising edge-triggered flip-flop clocked by clk) | 1st bit output of input register b to 27th bit input of the o/p register f |
| **800MHz** | **1478.95999** | **671.5012** | Startpoint: b\_r\_reg [1] (rising edge-triggered flip-flop clocked by clk)  Endpoint: f\_reg [22] (rising edge-triggered flip-flop clocked by clk) | 1st bit output of input register b to 22nd bit input of the o/p register f |
| 833.33MHz | slack not met (-0.04) | | |  |

Started the frequency at 500MHz and since the slack is positive, increased the frequency till 800MHz (clock period is 1.25ns). If frequency is increased from 800MHz to 833.33MHz (clock period is 1.2), slack is not met. So, the **maximum frequency for the system is 800MHz** (considering 0.05ns as precision for clock).

Critical path is the path with maximum delay in the circuit. For 800MHz, the critical path is between 1st bit output of b register to the 22nd bit of output register f.

As the frequency increases, the synthesis tool will add more combinational logic to the circuit, which in turn increases the total area.

As, the frequency increases, the dynamic power increases (because circuit has to switch faster), which in turn increases the total power consumption.

Energy = Power \* Time

For system to process 50 input cycles, since the output is generated 2 clock cycles later, it takes 52 cycles for the final output to come out of system.

Energy for 52 cycles = 671.5012uW \* 52 \* 1.25ns(shortest clock period)

= 43.647e-12 J

Dynamic power is dependent on frequency of circuit activity as it is largely the result of switching capacitive loads.

Dynamic power dissipation, ∝ .

Energy (dynamic) ∝ . The highest frequency at which a CMOS circuit can operate also has a proportionality with voltage.

Whereas, leakage power is caused by transistor not completely turning off and it is independent of frequency.

Energy(leakage) ∝ 1/f;

Leakage energy decreases with frequency and dynamic energy increases with frequency.

There is a slight increase in the energy with frequency according the outputs from synthesis tool.

1. There is no need to add reset signals for all the registers. If a control signal is generated for the system to clear the output f for one more clock cycle after reset is 0, then even if a & b has a junk value passed from input registers, it will not affect the output.

**PART3:**

1. To detect when overflow happened, we check the MSB of a\*b, MSB of previous value of f(f\_prev), and MSB of current value of f(f). If MSB of a\*b and f\_prev are 0 and MSB of f is 1, it means that there is an overflow. Here the output is set to 7ffffff so that it saturates.

Similarly, If MSB of a\*b and f\_prev are 1 and f is 0 then it means that is an overflow. Here the output is set to 8000000 so that it saturates.

The below logic is added to part2\_mac.sv:

if (~(f[27] ^ mult\_ab[27]) & (f[27]^f\_wire[27])) begin

if(f[27] == 0)

f\_wire = 134217727;

else

f\_wire = -134217728;

end

else begin

f\_wire = f + mult\_ab;

end

1. Used the same verilog testbench as part2 but modified the random generator to generate expected output that saturates (either 7ffffff or 8000000).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Max frequency | power | area | critical path |
| with saturation | 689.65MHz | 618.129 | 1673.40599 | Startpoint: b\_r\_reg[5] (rising edge-triggered flip-flop clocked  by clk) Endpoint: f\_reg[5] (rising edge-triggered flip-flop clocked by clk) |
| without saturation | 800MHz | 671.5012 | 1478.95999 | Startpoint: b\_r\_reg[1] (rising edge-triggered flip-flop clocked  by clk) Endpoint: f\_reg[22] (rising edge-triggered flip-flop clocked by clk) |

As there is an additional delay (due to added combinational logic) added to the circuit to implement saturation functionality , the maximum frequency got decreased to 689.65 MHz.

As the rate of switching(frequency) is reduced, the total power consumption is also reduced.

With the addition of saturation logic, there are more number of combinational cells that are included in the circuit and even if frequency is reduced, the area is still higher than part2.

In the case of system with saturation, critical path is between a 5th output bit of input register b to 5th input bit of output register f.

In the case of system without saturation, critical path is between a 1st output bit of input register b to 22nd input bit of output register f.

d)

Energy = Power \* Time

For system to process 50 input cycles, since the output is generated 2 clock cycles later, it takes 52 cycles for the final output to come out of system.

Energy for 52 cycles = 618.129uW \* 52 \* 1.45ns

= 46.607e-12 J

System consumes more energy than that of part2 if it were to process a sequence of 50 cycles of input values.

**Part4:**

1. **Adding extra register between multiplier and adder:**

Adding an extra register will delay the value of f from part3 by one more clock cycle. The control logic is changed, so that the enable signal given as input to output register will also be delayed by one more clock cycle. When reset, all 4 registers are reset. There is no change in the verilog testbench but we modified random generator to generate correct expected outputs taking the new delay into consideration.

|  |  |  |  |
| --- | --- | --- | --- |
| Max frequency | power | area | critical path |
| 869.56 MHz | 913.0095 | 1718.093983 | Startpoint: a\_r\_reg[9] (rising edge-triggered flip-flop clocked by clk) Endpoint: mult\_ab\_delay\_reg[19] (rising edge-triggered flip-flop clocked by clk) |

The critical path is changed from “input register - output register” to “input register - register between multiplier and accumulator”. Since the critical path is reduced, the maximum frequency increased from 689.65MHz to 869.56MHz.

1. **Pipelining the multiplier:**

Generated a control signal such that when the system is reset, the output ‘f’ is cleared for (S+1) clk cycles, where ‘S’ is the number of pipeline stages used. This makes sure that ‘f’ stays at zero until there are valid inputs reaching the input of output register. Valid\_out is also set to zero for (S+1) clk cycles and then follows valid\_in that is delayed for (S+1) clk cycles.

Summary of the pipeline stages synthesized:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mult Pipeline Stage | Clk  (ns) | Max  Frequency | Area(um2) | Power(uW) | Energy (J)  (Power \* Time) | Critical path |
| 2 | 1 | 1GHz | 1986.487983 | 1.3218e+03 | 71.3772e-12  (54 clk cycles) | Startpoint: b\_r\_reg[0] (rising edge-triggered flip-flop clocked by clk)  Endpoint: multinstance/mult\_x\_1/clk\_r\_REG28\_S1  (rising edge-triggered flip-flop clocked by clk) |
| 3 | 0.9 | 1.1GHz | 2175.081972 | 1.9279e+03 | 95.43e-12  (55 clk cycles) | Startpoint: multinstance/mult\_x\_1/clk\_r\_REG12\_S2  (rising edge-triggered flip-flop clocked by clk)  Endpoint: mult\_ab\_delay\_reg[27]  (rising edge-triggered flip-flop clocked by clk) |
| 4 | 0.8 | 1.25GHz | 2320.317966 | 2.3654e+03 | 105.97e-12  (56 clk cycles) | Startpoint: multinstance/mult\_x\_1/clk\_r\_REG56\_S1  (rising edge-triggered flip-flop clocked by clk)  Endpoint: multinstance/mult\_x\_1/clk\_r\_REG11\_S2  (rising edge-triggered flip-flop clocked by clk) |
| 5 | 0.8 | 1.25Ghz | 2408.895966 | 2.5727e+03 | 117.31512e-12  (57 clk cycles) | Startpoint: multinstance/mult\_x\_1/clk\_r\_REG102\_S1  (rising edge-triggered flip-flop clocked by clk)  Endpoint: multinstance/mult\_x\_1/clk\_r\_REG72\_S2  (rising edge-triggered flip-flop clocked by clk) |
| 6 | 0.8 | 1.25GHz | 2587.647957 | 2.9663e+03 | 137.63632  (58 clk cycles) | Startpoint: f\_reg[0] (rising edge-triggered flip-flop clocked by clk)  Endpoint: f\_reg[0] (rising edge-triggered flip-flop clocked by clk) |

From part3 to part4b with stage2\_multiplier, energy increased by 1.5 times.

From part3 to part4b with stage3\_multiplier, energy increased by 2 times.

From part3 to part4b with stage4\_multiplier, energy increased by 2.2 times.

From part3 to part4b with stage5\_multiplier, energy increased by 2.5 times.

From part3 to part4b with stage6\_multiplier, energy increased by 2.9 times.

Which is the best design?

Part4b with stage4\_multiplier is the best design considering frequency as the main performance criteria.

Part4b with stage5\_multiplier and stage6\_multiplier has the same maximum frequency as stage4\_multiplier with an increase in area and power. So, stage5 and stage6 cant be the best designs.

For part4b with stage4\_multiplier, the frequency is increased by 1.8 times when compared to part3, the energy is increased by 2.2 times and the area is increased by 1.3 times.

For part4b with stage3\_multiplier, the frequency is increased by 1.6 times when compared to part3, the energy is increased by 2 times and area is increased by 1.3 times.

For part4b with stage2\_multiplier, the frequency is increased by 1.4 times when compared to part3, the energy is increased by 1.5 times and area is increased by 1.18 times.

It will not be feasible to add pipelining to the adder. It is observed from the above table that the critical path is in the multiplier stages. So, the maximum delay is still caused by the multiplier and not the adder. Adding pipelining to adder will only cause the area to increase, but will not provide any improvement to the clock frequency.