ECE 327/627
Digital Hardware Systems

Tutorial 2

Q1

Write the SystemVerilog code for a combinational circuit that sorts two inputs (a, b) into two outputs (o1, o2) such that o1 >= o2. Assume both inputs are 4-bit numbers.

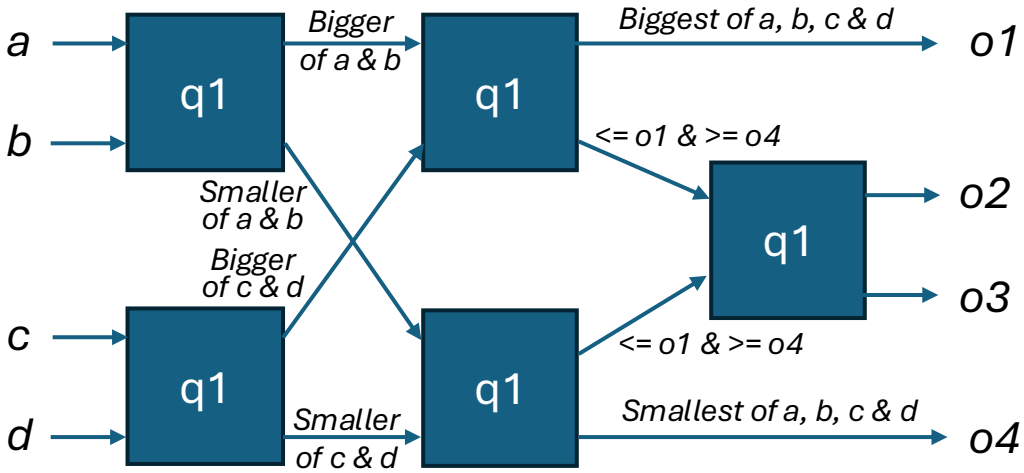
```
module q1 (
     input  [3:0] a,
     input  [3:0] b,
     output [3:0] o1,
     output [3:0] o2
);

assign o1 = (a>=b)? a : b;
assign o2 = (a>=b)? b : a;

endmodule
```

(Simple) purely combinational circuits can be described using continuous assignments or `always_comb` *blocks.*

```
module q1 (
     input  [3:0] a,
     input  [3:0] b,
     output [3:0] o1,
     output [3:0] o2
);

logic [3:0] out1, out2;

always_comb begin
     if (a>=b) begin
          out1 = a; out2 = b;
     end else begin
          out1 = b; out2 = a;
     end
end

assign o1 = out1;
assign o2 = out2;

endmodule
```

Q2

Using the circuit you implemented in Q1, design a circuit to sort four inputs (a,b,c,d) into four outputs (o1,o2,o3,o4) such that o1 >= o2 >= o3 >= o4.




```
module q2 (
      input  [3:0] a,
      input  [3:0] b,
      input  [3:0] c,
      input  [3:0] d,
      output [3:0] o1,
      output [3:0] o2,
      output [3:0] o3,
      output [3:0] o4
);

logic [3:0] t1,t2,t3,t4,t5,t6;

q1 q1_inst0(a, b, t1, t2);
q1 q1_inst1(c, d, t3, t4);

q1 q1_inst2(t1, t3, o1, t5);
q1 q1_inst3(t2, t4, t6, o4);

q1 q1_inst4(t5, t6, o2, o3);

endmodule
```

Q3

Given N numbers, each is 1-bit wide:

(a) How many bits will be required to represent the sum of these N numbers?

The highest value the sum of N 1-bit numbers can be is N. Therefore, we need $\lfloor log_2(N) \rfloor + 1$ *bits to represent this value.*

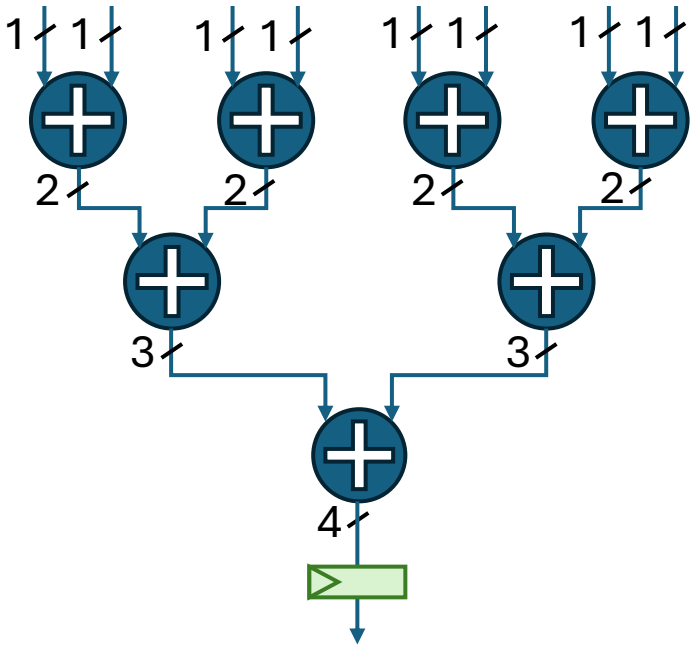
For example:

- *If N = 8, we need 4 bits*
- *If N = 15, we need 4 bits*
- *If N = 16, we need 5 bits*
- ...

Q3

Given N numbers, each is 1-bit wide:

(b) Design hardware to add N numbers in parallel. You can use as many 2-input adders as you need to compute the sum in 1 clock cycle. How many adders are needed? How many bits each adder will need?



Showing the circuit for N = 8

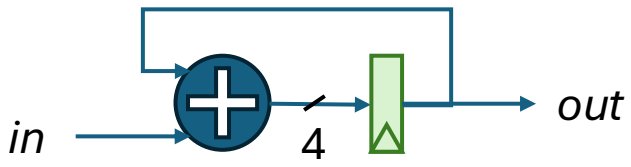
Latency = 1 cycle (to sum N numbers)
 # levels (L) = ceil($log_2 N$)
Delay = L x t_{adder}
 # adders = N – 1

Could be built as a chain instead of a tree
Latency = 1 cycle (to sum N numbers)
 # levels (L) = N – 1
Delay = L x t_{adder}
 # adders = N – 1

Q3

Given N numbers, each is 1-bit wide:

(c) Design hardware to add N numbers sequentially using a single adder. How many cycles are needed to sum these numbers? How many bits will the adder need?



Showing the circuit for N = 8 (needs 4 bits to represent the output)

Instead of pushing N inputs at the same time, push them sequentially one at a time over N cycles.

Latency = N cycle (to sum N numbers)
 # levels (L) = 1
Delay = L x t_{adder}
 # adders = 1 (regardless of N)