ECE-327 Solution to Midterm

2020t1 (Winter)

All requests for re-marks must be submitted by email to Mark Aagaard before noon on Friday March 22.

		Total Marks	Approx. Time	Page
Q1	VHDL Semantics	21	15	2
Q2	Area Analysis	25	15	5
Q3	FSM	27	18	7
Q4	DFD	27	18	10
Totals		100	66	

Q1 (21 Marks) VHDL Semantics

(*estimated time: 15 minutes*)

For each pair of VHDL programs in Q1a and Q1b, answer whether the signal `z` has the same behaviour in both programs.

NOTES:

1. The code shown is the body of the architecture.
2. All signals are of type `std_logic`.
3. The inputs are `a`, `b`, and `clk`.
4. "*Same simulation cycle behaviour*" means that the two programs have the same number of simulation cycles and that at the end of each simulation cycle, `z` has the same value in both programs.
5. Similarly, "*same simulation round behaviour*" means that the two programs have the same number of simulation rounds and that at the end of each simulation round, `z` has the same value in both programs.
6. If the programs do *not* have the same behaviour:
   - Describe (with text or waveforms) the behaviour of the inputs that will cause the different behaviour.
   - Describe (with text or waveforms) how the behaviour of `z` is different in the two programs.
7. If the programs do have the same behaviour: **justify** your answer with text or waveforms.

Q1a (10 Marks)

Program 1

```
c <= not a;
z <= b and c;
```

Program 2

```
process (a,b) begin
  z <= b and not a;
end process;
```

Answer:

Same simulation cycle *behaviour:* ***No***

In program 1, `z` is updated 1 simulation cycle later than in program 2. In program 1, in the first simulation cycle after `a` changes, `c` is updated. Then, in the second simulation cycle, `z` is updated. In program 2, `z` is updated in the simulation cycle after `a` changes.

More precisely, for program 1:

1. *`a` projected value changes*
2. *`a` visible value changes, `b` projected value changes*
3. *`b` visible value changes, `z` projected value changes*
4. *`z` visible value changes*

Same simulation round *behaviour:* ***Yes***

All of the visible values stabilize within the simulation round, so at the end of the simulation round, `z` will have the same value in both programs.

Marking:

5 marks *simulation cycle*

5 marks *simulation round*

Q1b (10 Marks)

The lines of code that are different in the two programs are denoted with underline.

Program 1

```
process begin
  clk <= '0';
  wait for 5 ns;
  clk <= '1';
  wait for 5 ns;
end process;

process begin
  wait until rising_edge(clk);
  z <= a;  %!!MDA wait;
end process;
```

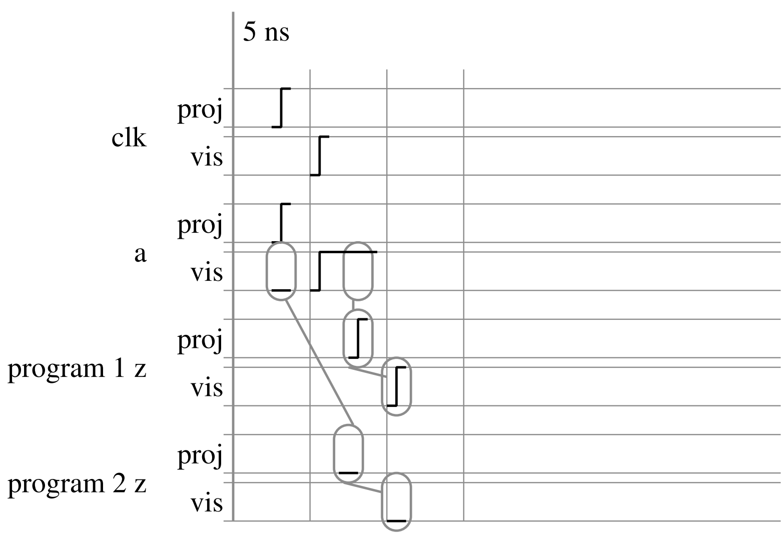
Program 2

```
process begin
  clk <= '0';
  wait for 5 ns;
  clk <= '1';
  wait for 5 ns;
end process;

process begin
  wait for 5 ns;
  z <= a;    %!!MDA wait;
end process;
```

Answer:

The signal `z` *will have different behaviour in the two programs if* `a` *changes exactly at 5 ns.*



VHDL code for `a`*:*

```
process begin
  a <= '0';
  wait for 5 ns;
  a <= '1';
  wait;
end process;
```

In program 2, `z` *samples* `a` *in the first simulation cycle of 5 ns. In program 1,* `z` *samples* `a` *in the simulation cycle that the change on* `clk` *becomes visible.*
The change on `a` *and the change on* `clk` *both become visible in the second simulation cycle of 5 ns. Thus, program 2 samples* `a` *before* `a` *changes and program 1 samples* `a` *after* `a` *changes.*

An alternative solution is based on the observation that Program 1 runs once every 10 ns and Program 2 runs once every 5 ns. This obviously results in different simulation cycle and simulation round behaviour.

Marking:

6 marks *simulation cycle*

5 marks *simulation round*

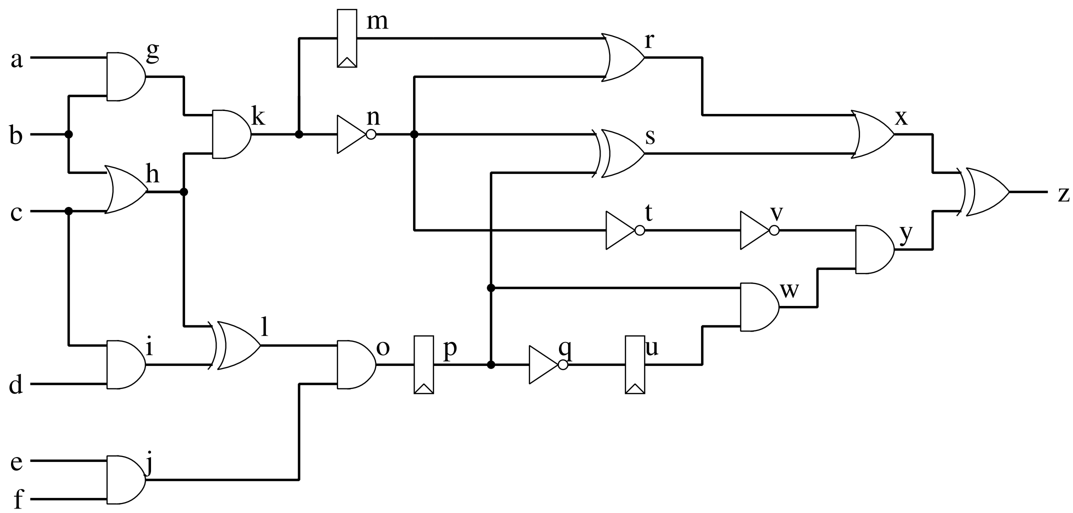
Q2 (25 Marks) Area Analysis

(*estimated time: 15 minutes*)

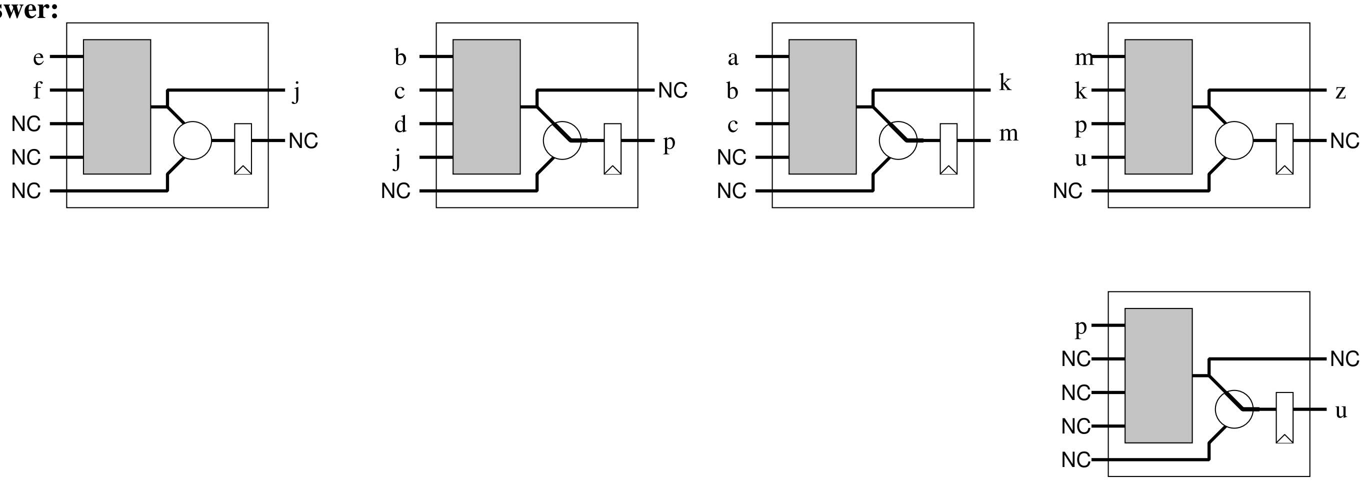
Design an FPGA implementation of the gate-level circuit shown below that uses the minimum number of FPGA cells. Use the FPGA cells on the following page to answer the question.

NOTES:

1. The primary inputs of the circuit are: `a`, `b`, `c`, `d`, `e`, and `f`.
2. The primary output of the circuit is: `z`.
3. Do not perform any logic optimizations.
4. For each FPGA cell that you use:
   - Label the input and output ports of the cell using the signal names from the gate-level circuit for ports that you use and **NC** (for no-connect) for ports that you do not use.
   - Show the configuration for the internal multiplexer.



Answer:



Marking:

Conceptual mistakes

- *missing signal j, k, m, p, u, z (1 mistake per signal)*
- *incorrect fanin for a LUT or flop*
- *missing LUT or flop (*i.e.*, a signal without a LUT or flop)*
- *not stopping at flop*
- *not sharing cell for unrelated lut and flop*
- *using a LUT as a wire*
- *missing NCs*
- *missing mux configurations*

-4 marks *1 mistake*

-8 marks *2 mistakes*

-11 marks *3 mistakes*

-14 marks *4 mistakes*

Optimality mistakes *mistakes in mapping gates to LUTs*

-2 marks *1 extra cell*

-4 marks *2 extra cells*

-5 marks *3 extra cells*

-2 marks *incorrect mapping of signal to flop vs comb*

-1 mark *labeling m (should be NC)*

-1 mark *each small mistake*

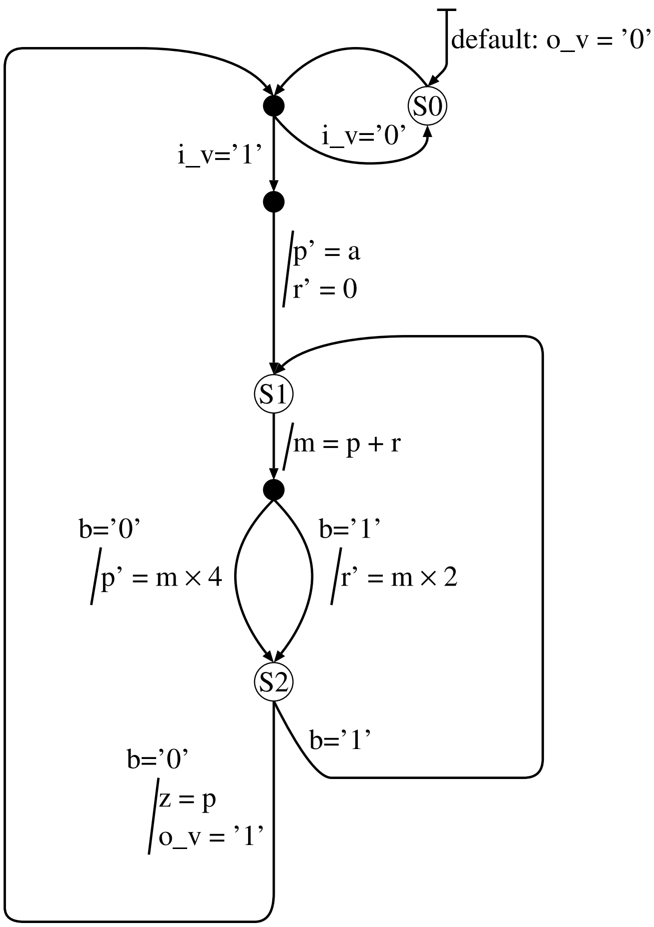
Q3 (27 Marks) FSM

(*estimated time: 18 minutes*)

Design the hardware for the state-machine shown below.

NOTES:

1. The design shall use the normal building blocks: arithmetic datapath components, multiplexers, Boolean gates, and registers.
2. The output and registers shall have the same behaviour, clock-cycle by clock-cycle, as the state machine.
3. Clearly label all inputs, registers, and outputs.
4. For each control signal (mux-select or chip-enable):
   - Label the signal in the schematic
   - In the space below the schematic, write an expression for the circuitry to drive the signal
   - Do *not* draw the circuitry to drive the signal
5. Marks will be earned for functional correctness, elegance of the design, and neatness of the drawing.



Q3a (4 Marks) State Encoding

Based on the ECE-327 guidelines, show your encoding for each state,

Answer:

S0	*00*
S1	*01*
S2	*10*

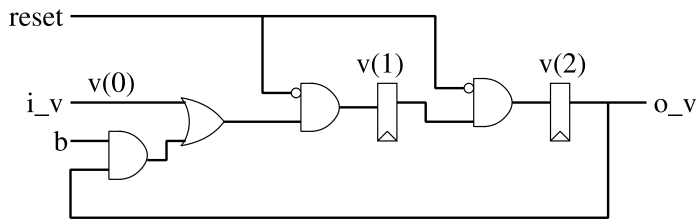
Marking:

- **4 marks** *valid bit encoding*
- **2 marks** *one-hot encoding*
- **1 mark** *binary encoding*
- **-1 mark** *each mistake in the encoding*

Q3b (9 Marks) Next-State Circuitry

Draw a schematic for the next-state circuitry.

Answer:



Marking:

- **+2 marks** *correct number of flops*
- **+2 marks** *reset is done correctly*
- **+2 marks** OR *gate with* `i_valid` *on input to* `v(1)`
- **+2 marks** AND *gate with* `b` *on input to*
- **+1 mark** *neatness and clarity* `v(1)`
- **-1 marks** *each mistake*

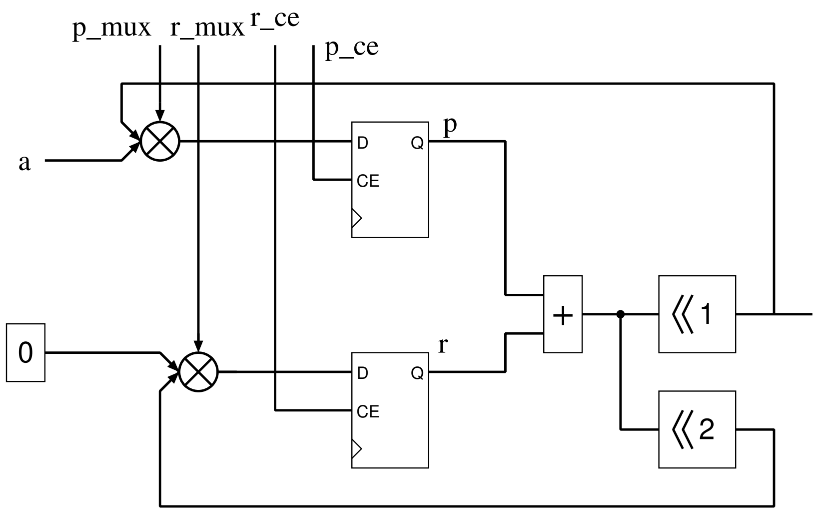
Write an expression for each control signal (mux-select or chip-enable) in the next-state circuitry.

Answer:
None

Q3c (14 Marks) Datapath Circuitry

Draw a schematic for the datapath circuitry.

Answer:



Marking:

- **+1 mark** *register for* p
- **+1 mark** *mux for* p
- **+1 mark** *chip-enable for* p
- **+1 mark** *register for* r
- **+1 mark** *mux for* r
- **+1 mark** *chip-enable for* r
- **+1 mark** *1 adder*
- **+1 mark** *mul by 2*
- **+1 mark** *mul by 4*
- **+1 mark** *neatness and clarity*
- **-0.5 marks** *use mult not shift*

Write an expression for each control signal (mux-select or chip-enable) in the datapath circuitry.

Answer:

p_mux	*= i_valid*
p_ce	*= i_valid or (state(0) and not b)*
r_mux	*= i_valid*
r_ce	*= i_valid or (state(0) and b)*

Marking:

- **+1 mark** *expression for mux select for* p
- **+1 mark** *expression for mux select for* q
- **+1 mark** *expression for chip enable for* p
- **+1 mark** *expression for chip enable for* q

Q4 (27 Marks) DFD

(*estimated time: 18 minutes*)

In this question, you will design and analyze a dataflow diagram for the equation:

```
z = a*e + b + b*e + 7*d + c*d
```

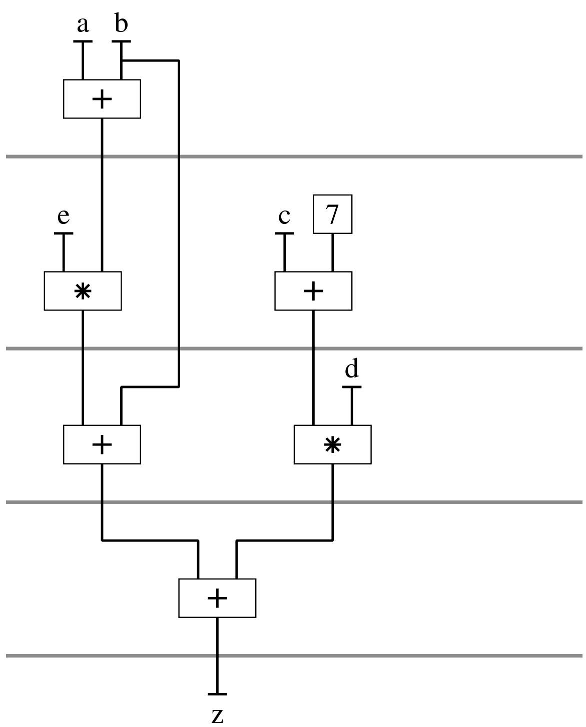
NOTES:

1. Inputs shall be combinational and outputs shall be registered.
2. The delay of a multiplier is approximately twice that of an adder.
3. Optimization goals in order of decreasing importance:
   (a) minimize number of *multipliers*
   (b) minimize *clock period*
   (c) minimize *latency*
   (d) minimize number of *adders* and *subtracters*
   (e) minimize number of *registers*
   (f) minimize number of *input ports*
   (g) minimize number of *output ports*
4. Input values may be read in any clock cycle, but each input value shall be read exactly once.
5. Algebraic optimizations are allowed, as long as the final value of `z` is correct.
6. You do *not* need to perform allocation.

Answer:

Optimal design

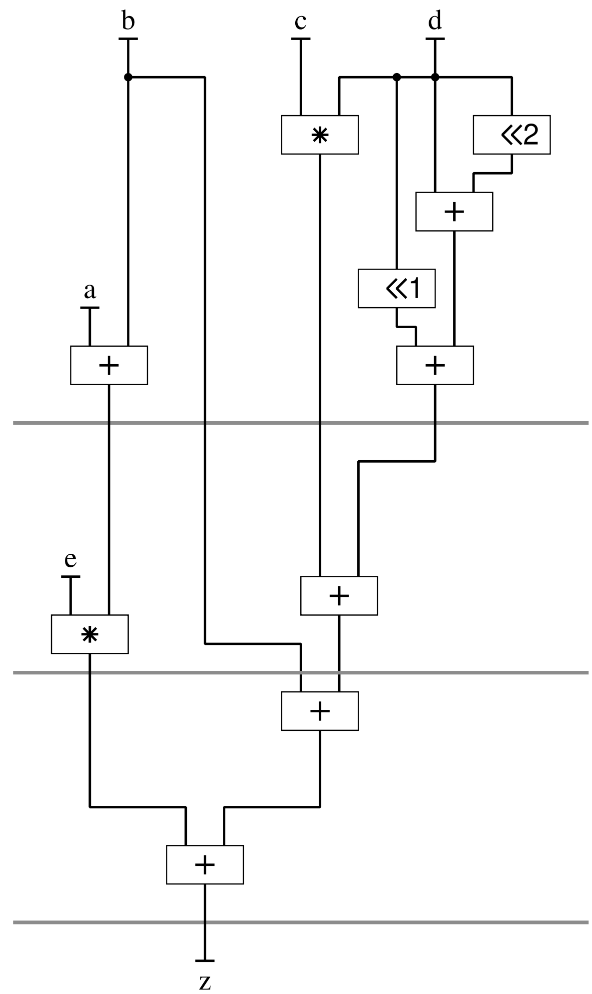
```
z = e*(a + b) + b + c*d +
    d + (d sll 1) + (d sll 2)
```



Latency	*3*
Clock period	*max(mul,2add) + flop*
Number of multipliers	*1*
Number of adders	*2*
Number of registers	*5*
Number of input ports	*4*
Number of output ports	*1*

Example suboptimal design

```
z = e*(a + b) + d*(c+7) + b
```



Latency	*4*
Clock period	*max(add, mul) + flop*
Number of muls	*1*
Number of adds	*1*
Number of regs	*3*
Number of inputs	*2*
Number of outputs	*1*

DFD: 23 marks max

Marking:

- **-6 marks** *2 multipliers*
- **-5 marks** *clock period = mul + add + flop*
- **-3 marks** *latency = 4*
- **-3 marks** *each extra adder*
- **-2 marks** *each extra register*
- **-1 marks** *each extra input*
- **-1 mark** *inputs not comb*
- **-1 mark** *outputs not reg*
- **-1 mark** *datapath components not comb*
- **-1 mark** *constant uses an input or reg*
- **-1 mark** *read an input multiple times*
- **-1 – -3 marks** *DFD is difficult to understand*

Analysis: 4 marks max

Marking:

- **1 mistake** *answer is off by 1*
- **2 mistakes** *answer is off by 2 or more*
- **3 mistakes** *missing answer*

- **4 marks** *0 mistakes*
- **3.5 marks** *1 mistake*
- **3 marks** *2 mistakes*
- **2.5 marks** *3–4 mistakes*
- **2 marks** *5–6 mistakes*
- **1 mark** *7 or more mistakes*