E&CE 327 Midterm

2012t1 (Winter)

All requests for re-marks must be submitted in writing to Mark Aagaard before 10:30am on Friday March 16.

A random collection of midterms were photocopied. Exams that are submitted for re-marking will be verified against this set.

Q1 (20 Marks) Simulation Algorithm

(estimated time: 15 minutes)

You have been hired a small software startup company, Synflopsys, that is creating a new simulator for VHDL programs. The main feature of the simulator is that the value of each signal will be computed *at most once per simulation round.*

NOTES:

1. A *simulation round* is a sequence of simulation cycles that all represent the same moment in time.

Q1a (5 Marks) What would be the most important advantage of the Synflopsys simulator over a standard delta-cycle simulator?

Answer:

Faster simulation runs. By computing the value of each signal at most once per simulation round, the time to run a simulation will be decreased.

Marking:

5 marks *"faster simulation"*
4 marks *"simpler simulation algorithm"*
3 marks *demonstration of significant understanding*
1 mark *some correct information*

Q1b (5 Marks) For which VHDL programs could the proposed simulator give the same simulation results at each real moment in time (*e.g.*, each nanosecond) as a standard VHDL delta-cycle simulator? **For full marks, you must justify your answer.**

Answer:

VHDL programs without combinational loops.

Marking:

5 marks *correct answer*
3 marks *demonstration of significant understanding*
1 mark *some correct information*

Q1c **(10 Marks)** List the two fundamental rules that zero-delay simulators must obey, and then *briefly* hypothesize how the Synflopsys simulator attempts to achieve these rules.

Answer:

Rule 2 *The gates execute in parallel*

Hypothesis *Because there are no combinational loops, there is a topological order among the signals, and the values can be computed in topological order.*

Rule 1 *Events must propagate instantaneously through combinational circuitry.*

Hypothesis *Because each signal changes its value at most once per moment in time, increment time at the end of each iteration of computing the signals in topological order. There is no need for delta cycles or other infinitesimally small increments of time.*

Marking:

2 marks *for each correct rule*

3 marks *for each correct hypothesis*

Q2 (20 Marks) VHDL

(estimated time: 15 minutes)

For each of the processes below, draw the behaviour of the signals `i` and `z` on the waveform to the right of the process.

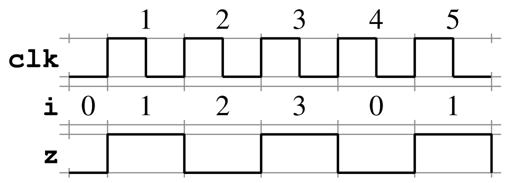
NOTES:

1. The signal declarations are:
   ```
   signal i : unsigned( 3 downto 0 );
   signal z : std_logic;
   ```
2. Draw the waveforms for clock cycles 0 through 5.
3. The waveform for `i` shall be drawn as integers (*e.g.*, 0,1,2...).
4. The waveform for `z` shall be drawn as a line (*i.e.*, in the same style as `clk`)

Q2a (6 Marks)

```
process begin
  i <= to_unsigned( 0, 4 );
  z <= '0';
  wait until rising_edge(clk);
  while i < 3 loop
    i <= i + 1;
    z <= not( z );
    wait until rising_edge(clk);
  end loop;
end process;
```

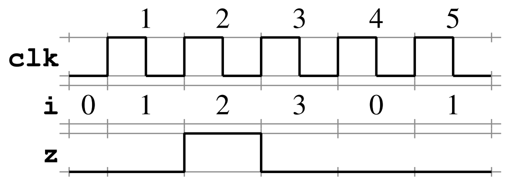
Answer:



Q2b (7 Marks)

```
process begin
  i <= to_unsigned( 0, 4 );
  z <= '0';
  wait until rising_edge(clk);
  while i < 3 loop
    i <= i + 1;
    z <= i(0);
    wait until rising_edge(clk);
  end loop;
end process;
```

Answer:

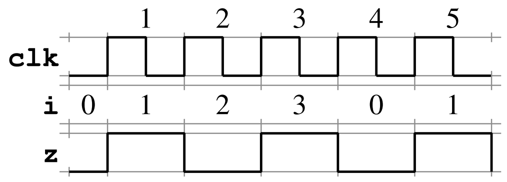


reg asn to z, z delayed by one clock cycle

Q2c (7 Marks)

```
process begin
  i <= to_unsigned( 0, 4 );
  z <= '0';
  wait until rising_edge(clk);
  i <= i + 1;
  while i < 3 loop
    z <= not( z );
    wait until rising_edge(clk);
    i <= i + 1;
  end loop;
end process;
```

Answer:



matches #1

Q3 (20 Marks) FPGA Cells

(estimated time: 15 minutes)

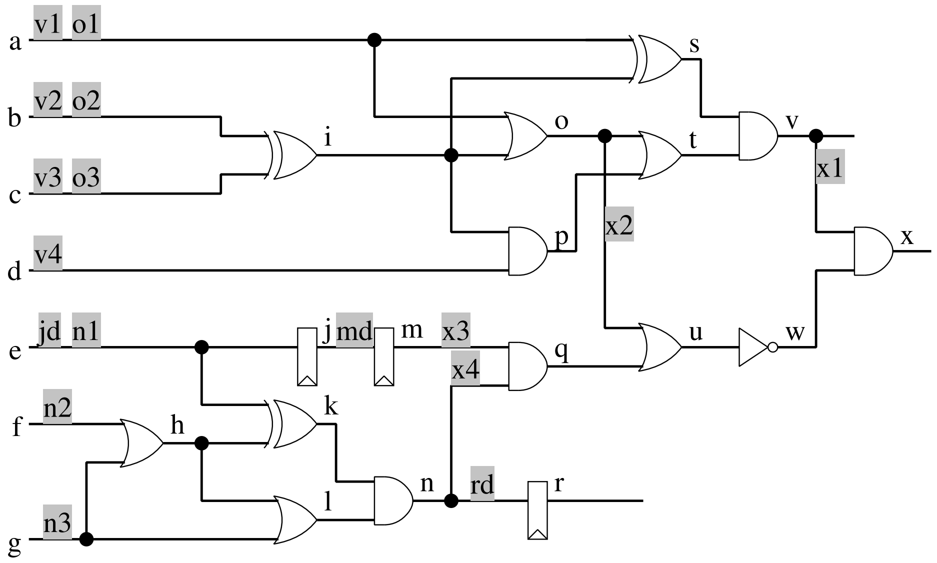
Design an FPGA implementation of the gate-level circuit shown below that uses the minimum number of FPGA cells. Use the FPGA cells on the following page to answer the question.

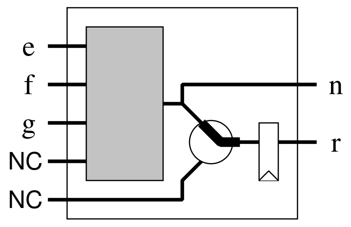
NOTES:

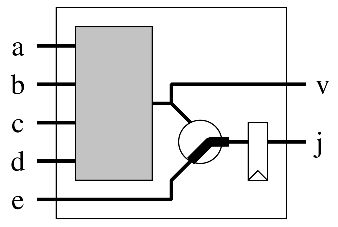
1. The primary inputs of the circuit are: a, b, c, d, e, f, and g.
2. The primary outputs of the circuit are: r, v, and x.
3. Do not perform any logic optimizations, e.g. Karnaugh map simplifications.
4. For each FPGA cell that you use:
   - Label the input and output ports of the cell using the signal names from the gate-level circuit for ports that you use and **NC** (for no-connect) for ports that you don't use.
   - Show the configuration for the internal multiplexer by connecting either the PLA or input port of the cell to the flop.

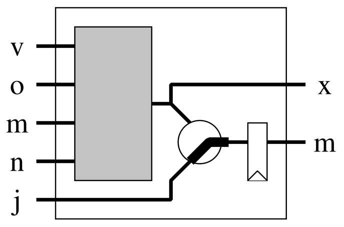
Answer:

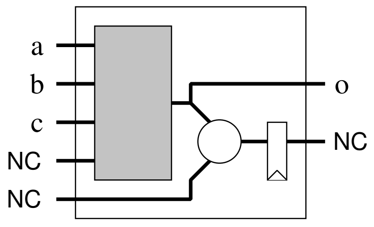
The gray boxes describe how the signals are used. For example, "n1" is the first input to the PLA that drives n *and "jd" is the D-input to the flop that drives* j*.*



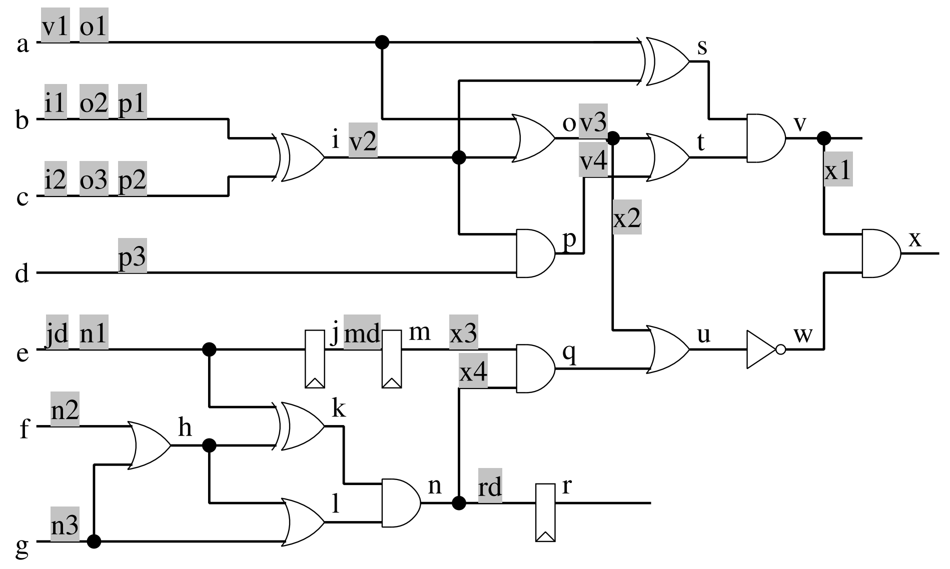

e
f
g
NC
NC
n
r

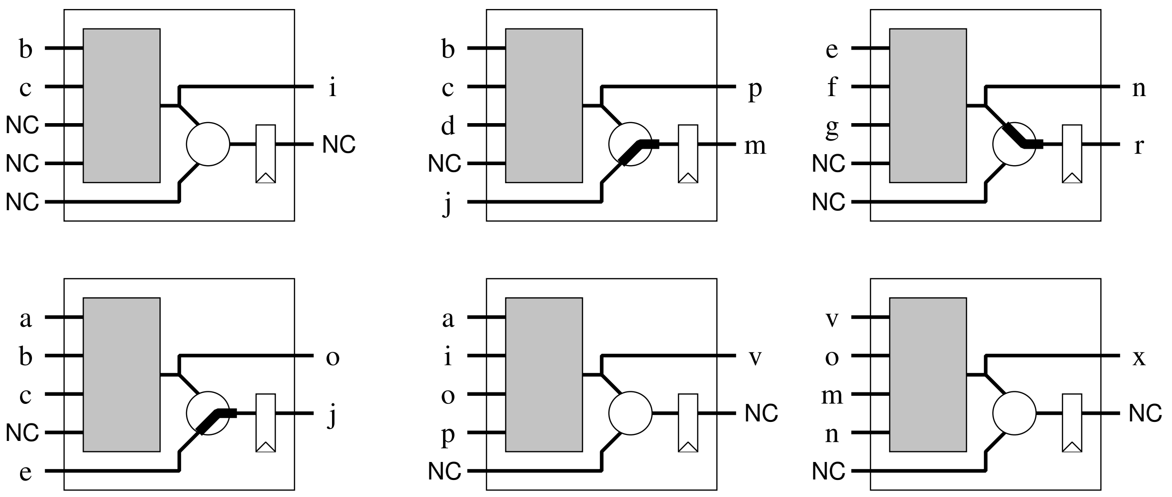

a
b
c
d
e
v
j


v
o
m
n
j
x
m


a
b
c
NC
NC
o
NC

Answer if stop when first encounter 4 inputs per lut:





Marking:

20 marks *correct answer*

18 marks *5 FPGA cells*

16 marks *functionally correct, 6 FPGA cells*

14 marks *functionally correct, 7 FPGA cells*

7 marks *1 gate per lut (14 cells)*

5 marks *at least one correct lut mapping and one correct flop*

-1 mark *each small mistake*

-1 mark *missing NC*

-1 mark *missing mux configuration*

-1 mark *incorrect connection between inputs, flop, and output*

-3 marks *not sharing cell for unrelated lut and flop*

Q4 (25 Marks) Dataflow Diagram and Control Table

(estimated time: 20 minutes)

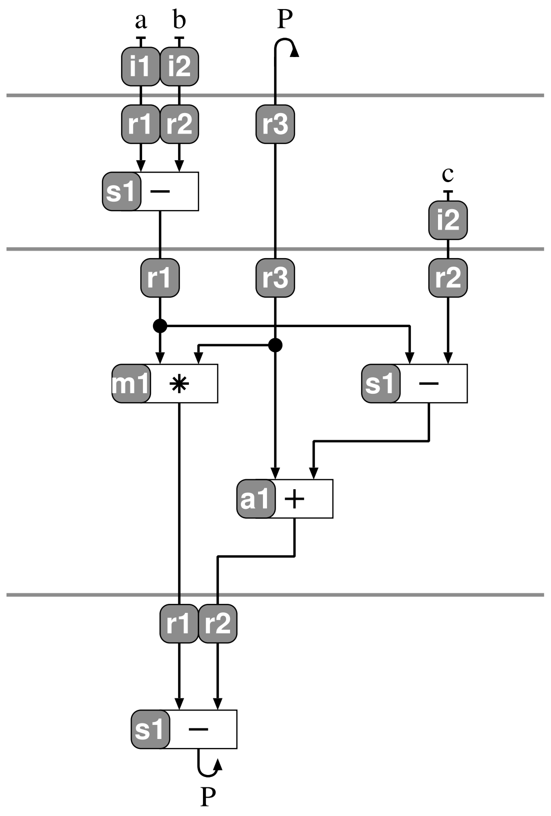
This question explores various aspects of the system described by the specification and dataflow diagram below.

Specification: `P' = P * (a - b) - (P + a - b - c)`

Q4a (4 Marks) Register Allocation

Allocate registers for the last clock-cycle boundary in the dataflow diagram.

Answer:



Marking:

2 marks *Each register*

Q4b (6 Marks) Dataflow diagram analysis

Fill in the boxes in the table below.

Answer:

Latency	*3 clock cycles*
Throughput	*1/3*
Clock period	*max(mul, sub+add)*

Marking:

2 marks *each answer*

Q4c (9 Marks) Control Table

Draw the control table for the registers `r1`, `r2`, and `r3` from the dataflow diagram.

NOTES:

1. Label each row and column of the table clearly.
2. The system shall support an *indeterminate number of bubbles* between valid parcels. If you do not know how to support an indeterminate number of bubbles with this system, you may earn *part marks* by writing a $\surd$ in the box at at the end of this note and answering the question for parcels that *arrive as soon as possible (ASAP).*

   I am answering for *ASAP parcels*, not an indeterminate number of bubbles. ☐
3. Mark cells as "don't-care" or "unused" using the symbols from class. If you do not know the symbols, you may earn *part marks* by writing a $\surd$ in the box at at the end of this question and using "DC" for "don't-care" and "UN" for unused.

   I am answering with *DC and UN*, not the symbols from class. ☐

Answer:

Note: complete control table shown, but only the registers are needed.

		r1		**r2**		**r3**		**a1**		**s1**		**m1**	
		ce	**d**	**ce**	**d**	**ce**	**d**	**src1**	**src2**	**src1**	**src2**	**src1**	**src2**
	idle	—	—	—	—	0	—	—	—	—	—	—	—
	0	1	i1	1	i2	×	×	×	×	×	×	×	×
clock	1	1	s1	1	i2	0	—	—	—	r1	r2	—	—
cycle	2	1	m1	1	a1	—	—	r3	s1	r1	r2	r1	r3
	3	×	×	×	×	1	s1	—	—	r1	r2	—	—

Marking:

1 mark	*registers as labels for x-axis*
1 mark	*states as labels for y-axis*
1 mark	*chip-enable, d-input labels for each register*
1 mark	*included idle state*
1 mark	*d-input is don't-care when ce='0'*
1 mark	*r3 is don't-care in cycle-2*
1 mark	*correct values for "unused"*
1 mark	*correct values for chip-enable*
1 mark	*correct values for d-inputs*
-1 mark	*used ASAP parcels*
-1 mark	*used "DC/UN" rather than symbols from class*

Q4d (6 Marks) State Encoding

What type of state encoding would be best for this system?

1. If used indeterminate number of bubbles

   Answer:
   Valid bit, because of the indeterminate number of bubbles.

   Marking:

   3 marks *valid bits*
   2 marks *one-hot*
   1 mark *binary*

2. If used ASAP parcels

   Answer:
   One-hot

   Marking:

   3 marks *one-hot*
   2 marks *valid bits*
   1 mark *binary*

How many states does the system have?

1. If used indeterminate number of bubbles

   Answer:
   6: one for each of the 4 clock cycle that a parcel spends within the system, one idle state, and one state for when a parcel enters in the same clock cycle as the previous parcel leaves.

   Marking:

   3 marks *6 states*
   2 marks *5 states*
   1 mark *other reasonable answer*

2. If used ASAP parcels

Answer:
3 states

Marking:

3 marks *3 states*
2 marks *2 or 4 states*
1 mark *other reasonable answer*

Q5 (15 Marks) Memory

(estimated time: 10 minutes)

Draw a dataflow diagram that implements the specification:

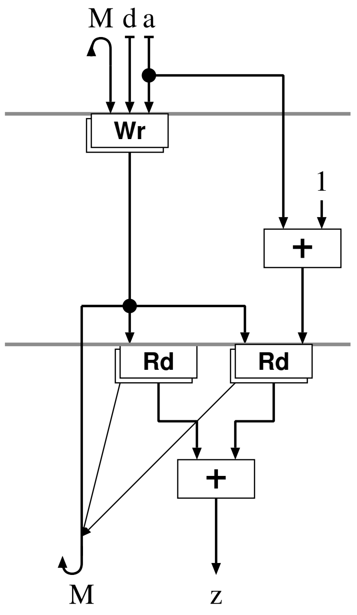
```
M'[a] = d;
z     = M[a] + M[a+1];
```

NOTES:

1. Inputs shall be *registered*
2. Outputs shall be *combinational*
3. Memory has registered inputs and combinational outputs (same as in class)
4. The memory is *dual-ported* with one read/write port and one read port.
5. There are exactly *five bubbles* between each pair of consecutive valid parcels.
6. The first optimization goal is to minimize *latency*
7. The second optimization goal is to minimize *area*: first minimize the number of registers, then the number of adders.

Answer:

```
M'[a] = d;
z     = M[a] + M[a+1
```



Marking:

2 marks	*functional correctness*
2 marks	*DFD syntax is correct*
2 marks	*DFD uses M*
1 marks	*Wr produces M*
1 marks	*mem operations on clock cycle boundaries*
1 marks	*no Wr(M, a) and Rd(M, a) in same clock cycle*
1 marks	*registered inputs*
1 marks	*combinational outputs*
1 marks	*second add in same clock cycle as read*
1 marks	*optimal latency*
1 marks	*anti-dependency arrows*
1 marks	*do 2 reads, do not store d in a register*