ECE-327 Solution to Midterm

2019t1 (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	20	15	2
Q2	FPGA Area	25	15	4
Q3	FSM	27	18	6
Q4	Code Review	28	20	8
Totals		100	68	

Q1 (20 Marks) VHDL Semantics

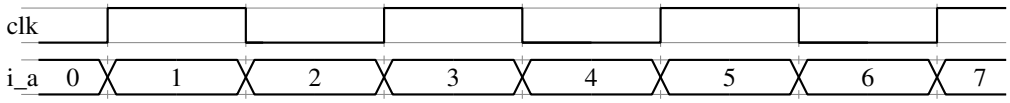
(*estimated time: 15 minutes*)

Which of the waveforms on the next page is/are the best choice(s) for a testbench waveform?

NOTES:

1. The testbench shall drive the clock signal (`clk`) and the data input (`i_a`), which shall increment.
2. Multiple waveforms may be best.
3. For each waveform, answer whether it is (one of) the best waveform(s).
4. For full marks, you must justify your answers in terms of VHDL simulation semantics.
5. If a waveform has the same justification as a previous waveform, write "Same as *x*", where *x* is the number of the previous waveform.

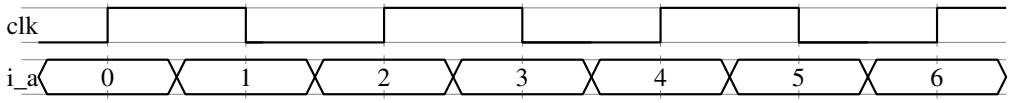
Waveform 1



	Yes	No
Best	☐	√

Justification: 1) The problem is that `i_a` transitions at the same time as `clk`. For example, `clk` has a rising edge at the same time that `i_a` transitions from 2 to 3. In the simulation cycle where `clk='1'` for `rising_edge(clk)`, `i_a` will be 3. Thus, a flop driven by `i_a` will see 3, not 2. This will cause the flop to act like a wire, not a flop. 2) Data transitions twice per clock cycle, so only half of the values will be read by registers in the circuit.

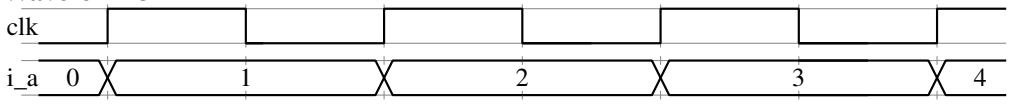
Waveform 2



	Yes	No
Best	☐	√

Justification: Data transitions twice per clock cycle (see reason 2 in Waveform 1).

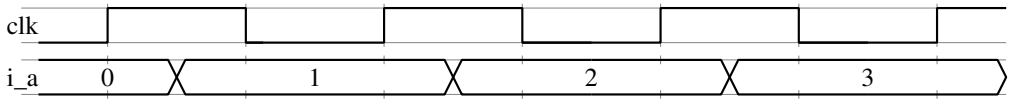
Waveform 3



	Yes	No
Best	☐	√

Justification: Data transitions at the same time as the rising edge on the clock (see reason 1 in Waveform 1).

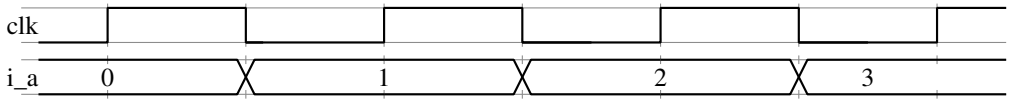
Waveform 4



	Yes	No
Best	√	☐

Justification: Data is stable at the rising edge of the clock and transitions once per clock cycle.

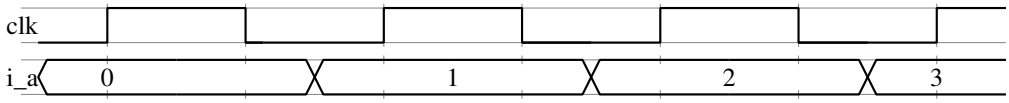
Waveform 5



	Yes	No
Best	√	☐

Justification: Same as 4.

Waveform 6



	Yes	No
Best	√	☐

Justification: Same as 4.

Marking:

+5 marks *`i_a` should not change at same time as a `clk` edge.*

+5 marks *`i_a` should change once per clock cycle*

+2 marks *A flip-flop will act as a wire.*

+2 marks *`clk='1'` in same* simulation cycle *as* new *value of `i_a`*

+6 marks *Correctly identify whether waveforms are best.*

Q2 (25 Marks) FPGA Area

(*estimated time: 15 minutes*)

Calculate the minimum number of FPGA cells needed to implement each of the VHDL code snippets below.

NOTES:

1. The signals `a`, `b`, `c`, `d`, `e`, `f`, and `g` are `std_logic`.
2. The signals `m`, `n`, `p`, and `z` are 8-bit `unsigned`.
3. Optimizations are allowed, so long as the externally visible input-to-output behaviour of the system does not change.
4. For full marks, you must justify your answer with a drawing and/or text.

Q2a

```
g <= ((a and b) xor (c or d)) and (e xor f);

z <= m + n when g = '1'
else m + p;
```

	LUTs	Regs	Justification
`g`	2	0	Combinational function with 6 inputs and 1 output.
`z`	8	0	8-bit adder with a 2:1 mux on one input.
Total:	10	0	

Number of FPGA cells: 10

Marking:

- **+2 marks** *`g` requires 2 LUTs*
- **+2 marks** *`z` requires 1 LUT/bit*
- **+2 marks** *`z` requires 8 LUTs in total*
- **+2 marks** *circuit requires 8+2 cells.*

Q2b

```
process begin
  wait until rising_edge( clk );
  n <= m;
end process;

z <= n + p;
```

	LUTs	Regs	Justification
`n`	0	8	8-bit register
`z`	8	0	8-bit adder
Total:	8	8	Cells = max(LUTs, Regs) = 8

Number of FPGA cells: 8

Marking:

- **+2 marks** *`n` requires 1 flop/bit*
- **+2 marks** *`z` requires 1 LUT/bit*
- **+2 marks** *`n` requires 8 flops and `z` requires 8 LUTs*
- **+2 marks** *circuit requires max(8,8) = 8 cells.*

Q2c

```
z <=   m     when n = 0
  else m + 1;
```

	LUTs	Regs	Justification
`n=0`	3	0	Combinational function with 8 inputs and 1 output
`z`	8	0	8-bit adder
Total:	11	0	

Number of FPGA cells: 11

Marking:

+2 marks *`n=0` requires 3 LUTs*

+2 marks *`z` requires 1 LUT/bit*

+2 marks *`z` requires 8 LUTs*

+2 marks *circuit requires 3+8 = 11 LUTs = 11 cells*

+1 mark *answered each of a, b, c.*

Q3 (27 Marks) FSM

(*estimated time: 18 minutes*)

In this question, you will design and analyze a state machine that performs the computation defined in pseudocode below:

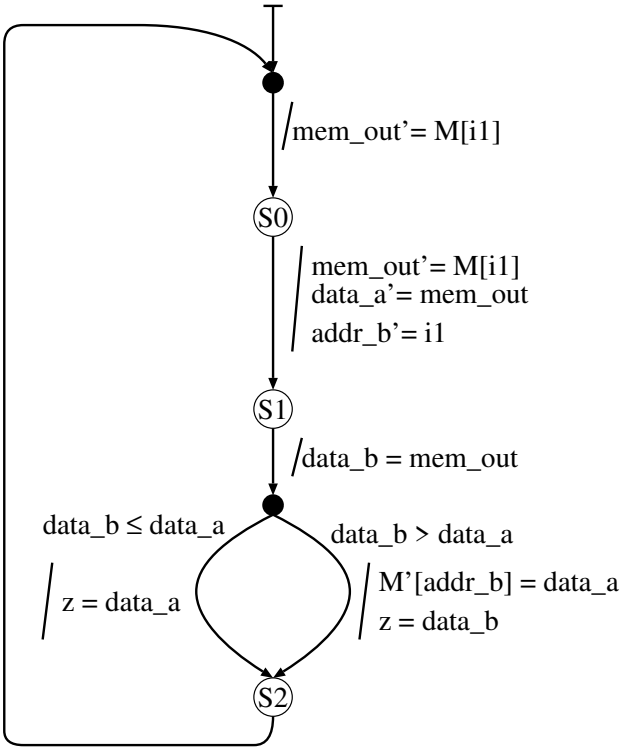
```
if M[a] > M[b] then
  M[b] = M[a];
  z    = M[b];
else
  z    = M[a];
```

NOTES:

1. The system has one input port: `i1`.
2. The input port `i1` is used for both of the input values (`a` and `b`) but each value is read in a different clock cycle. The values `a` and `b` may be read from `i1` in any order and in any clock cycle. Each value appears on `i1` for exactly one clock cycle.
3. The system has an internal memory array `M`, which shall be *single-port*.
4. The system has one output port: `z`.
5. The system shall use an ASAP parcel schedule.
6. Optimization goal: maximize throughput.
7. Marks will be earned for syntactic correctness, functional correctnes, maximized throughput, simplicity and elegance, and neatness.

Q3a (20 Marks) Design

Draw the state machine.



Marking:

- **+2 marks** *reset is done correctly*
- **+2 marks** *state transitions cover all possibilities and are mutually exclusive*
- **+2 marks** *correct syntax and use of registered and combinational assignments*
- **+2 marks** *correct syntax and use of conditions and assignments*
- **+2 marks** *at most one memory operation per clock cycle*
- **+2 marks** *all memory reads have same target variable*
- **+2 marks** *throughput is maximized*
- **+2 marks** *uses ASAP parcel schedule*
- **+2 marks** *simplicity and elegance of design*
- **+2 marks** *neatness of drawing*
- **-4 marks** *incorrect overall functionality of algorithm*
- **-4 marks** *does not use `i1` correctly*

Q3b (7 Marks) Analysis

What is the maximum throughput of your system?

Answer:

1/3 parcels/clock-cycle

If you were to use a dual-port memory, could you increase the throughput? **For full marks, you must justify your answer.**

Answer:

Yes. With a dual port memory, we could remove S2. With a single-port memory, S2 is needed so that we do only one memory operation per clock cycle. With a dual port memory, we could overlap the writing and reading Removing S2 will reduce the latency of the system from 3 to 2. Throughput is 1/Latency, therefore reducing the latency will increase the throughput.

Marking:

2 marks *Correct throughput*

5 marks *Dual port analysis*

- **5 marks** *Correct answer with complete justification.*
- **4 marks** *Correct answer with mostly complete justification*
- **3 marks** *Incorrect answer with excellect justification*
- **2 marks** *Correct description of advantages of dual-port memory*

Q4 (28 Marks) Code Review

(*estimated time: 20 minutes*)

In this question, you will analyze the VHDL program `count_sum` on the next page for synthesizability, functional correctness, and optimality.

The purpose of the program is to compute `sum = sum + a - b` for each parcel, and count how many parcels it takes until `sum >= 17`. When `sum >= 17`, the system sets `o_valid='1'` and outputs the count on `o_count`. The functional specification in pseudocode is to the right.

```
sum = sum + a - b;
count = count + 1;
if sum >= 17 then {
  o_count = count; -- generate output
  sum = 0;
  count = 0;
}
```

NOTES:

1. The code is legal VHDL.
2. The system has 1 input data port (`i_data`). When `i_valid='1'`, `i_data` has the value of `a`. In the next clock cycle `i_data` has the value of `b`.
3. Write down the 5 highest priority comments. The priority of comments, in decreasing order of priority is:

Synth	Synthesizability
Fun	Functional correctness
Tput	Optimizations to increase throughput
Reg	Optimizations to decrease registers
Comb	Optimizations to decrease combinational circuitry
Good	Praise for something good in the code

4. For each comment, put a $\surd$ in the box for its category. Comments may give replacement code, explain how to change the existing code, or praise a good feature of the existing code.
5. For full marks, each comment must be *briefly* justified.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity count_sum is
6    port
7      ( clk, reset : in std_logic;
8        i_valid    : in std_logic;
9        i_data     : in signed( 7 downto 0 );
10       o_valid    : out std_logic;
11       o_count    : out unsigned( 7 downto 0 )
12     );
13 end entity;
14
15 architecture main of count_sum is
16   signal v           : std_logic_vector( 0 to 3 );
17   signal count       : unsigned( 7 downto 0 );
18   signal a, b, sum   : signed( 7 downto 0 );
19 begin
```

```
process begin
  wait until rising_edge( clk );
  if reset = '1' then
    v     <= ( others => '0' );
    count <= ( others => '0' );
  else
    v <= i_valid & v( 1 to 3 );
  end if;
end process;

process begin
  wait until rising_edge( clk );
  if v(0) = '1' then
    a     <= i_data;
    count <= count + 1;
  elsif v(1) = '1' then
    b     <= i_data;
  end if;
end process;

process (v) begin
  if v(2) = '1' then
    sum <= sum + a - b;
  elsif v(3) = '1' then
    if sum >= 17 then
      sum   <= (others => '0');
      count <= (others => '0');
    end if;
  end if;
end process;

o_valid <=   '1' when sum >= 17
        else '0';

o_count <= count;

end architecture;
```

Marking:

+6 marks *Multiple drivers on* `count`*. All assignments to* `count` *should be in the same process.*

+6 marks *Process for sum and count is a latch, it should be clocked*

+5 marks `v(0)` *should be a wire from* `i_valid`

+5 marks `v` *is not shifted correctly*

+5 marks *Throughput could be increased and registers could be decreased by changing* `b` *from a register to combinational and moving* `sum <= sum + a - b` *from* `v(3)` *to* `v(2)`*.*

+4 marks *Throughput could be increased by creating a combinational signal* `sum_next` *as follows:*

```
sum_next <= sum + a - b;

process begin
  wait until rising_edge(clk);
  if v(2) = '1';
    if sum_next >= 17 then
      sum <= (others => '0');
      count <= (others => '0');
    else
      sum <= sum_next;
    end if;
  end if;
end process;
```

+4 marks *Combinational loop*

+4 marks *Sum needs to be reset*

+1 mark *clarity and neatness*

NOTE: incomplete justifications earned partial marks

This page is for scratch work for any question.