ECE 327: Digital Systems Engineering
Tutorial Problems

2018t1 (Winter)

Mark Aagaard

University of Waterloo
Department of Electrical and Computer Engineering

Contents

Chapter 1

Fundamentals of VHDL

1.1 IEEE 1164

For each of the values in the list below, answer whether or not it is defined in the `ieee.std_logic_1164` library. If it is part of the library, write a 2–3 word description of the value.

Values: `'-','#','0','1','A','h','H','L','Q','X','Z'`.

1.2 VHDL Syntax

Answer whether each of the VHDL code fragments q2a through q2f is legal VHDL code.

NOTES: 1) "`...`" represents a fragment of legal VHDL code.
2) **For full marks, if the code is illegal, you must explain why.**

q2a
```
architecture main of anchiceratops is
  signal a, b, c : std_logic;
begin
  process begin
    wait until rising_edge(c);
    a <= if (b = '1') then
           ...
         else
           ...
         end if;
  end process;
end main;
```

q2b
```
architecture main of tulerpeton is
begin
  lab: for i in 15 downto 0 loop
    ...
  end loop;
end main;
```

q2c
```
architecture main of metaxygnathus is
  signal a : std_logic;
begin
  lab: if (a = '1') generate
    ...
  end generate;
end main;
```

q2d
```
architecture main of temnospondyl is
  component compa
    port (
      a : in std_logic;
      b : out std_logic
    );
  end component;
  signal p, q : std_logic;
begin
  coma_1 : compa
    port map (a => p, b => q);
  ...
end main;
```

q2e
```
architecture main of pachyderm is
  function inv(a : std_logic)
    return std_logic is
  begin
    return(NOT a);
  end inv;
  signal p, b : std_logic;
begin
  p <= inv(b => a);
  ...
end main;
```

q2f
```
architecture main of apatosaurus is
  type state_ty is (S0, S1, S2);
  signal st : state_ty;
  signal p  : std_logic;
begin
  case st is
    when S0 | S1 => p <= '0';
    when others  => p <= '1';
  end case;
end main;
```

1.3 Flops, Latches, and Combinational Circuitry

For each of the signals p...z in the architecture main of montevido, answer whether the signal is a latch, combinational gate, or flip-flop.

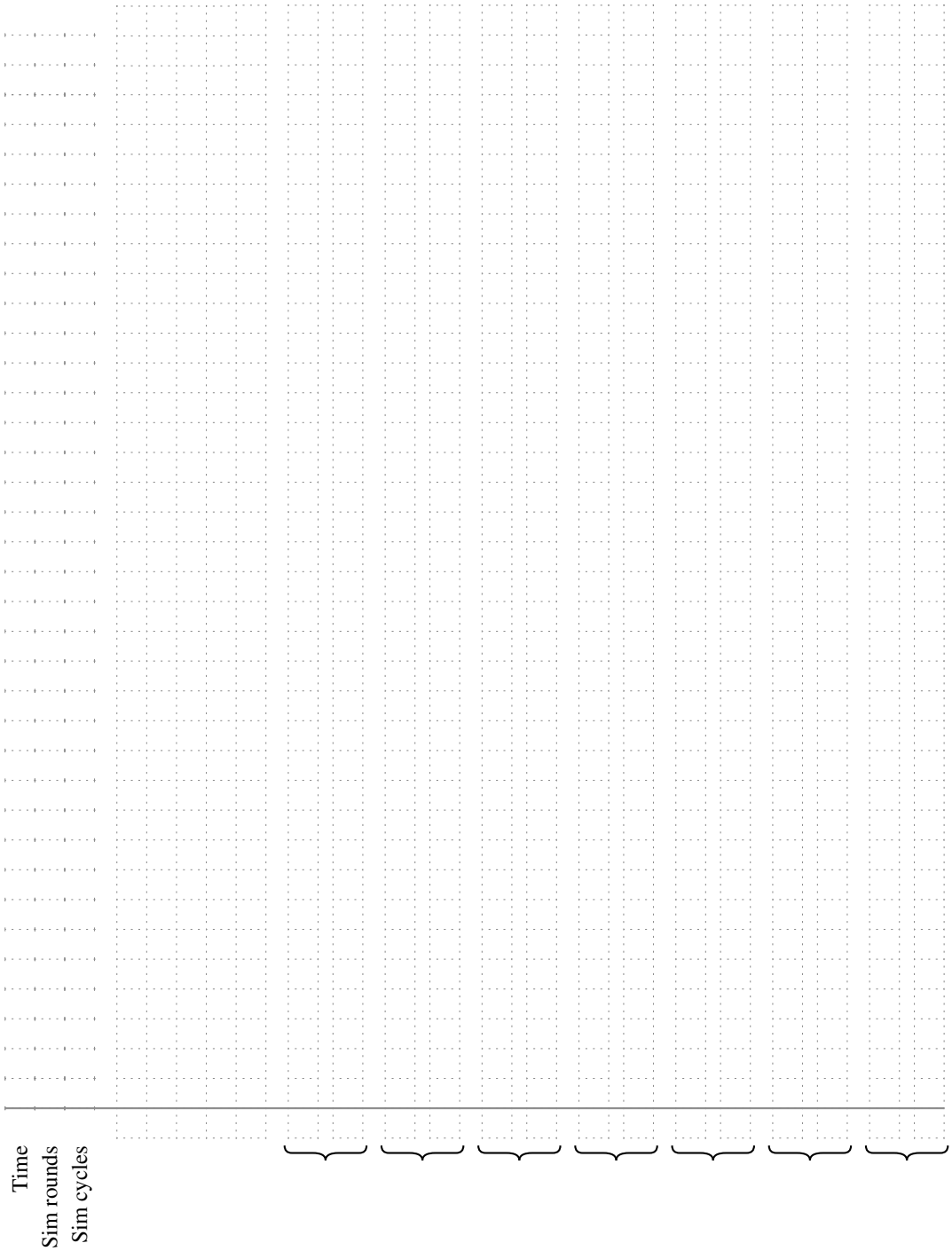
```
entity montevido is
  port (
    a, b0, b1, c0, c1, d0, d1, e0, e1 : in std_logic;
    l : in std_logic_vector (1 downto 0);
    p, q, r, s, u, v, y, z : out std_logic
  );
end montevido;


architecture main of montevido is
  signal i, j, t, w, x : std_logic;
begin

  i <= c0 xor c1;
  j <= c0 and c1;

  process (a, i, j) begin
    if (a = '1') then
      p <= i and j;
    else
      p <= not i;
    end if;
  end process;

  process (a, b0, b1) begin
    if rising_edge(a) then
      q <= b0 and b1;
    end if;
  end process;

  process
    (a, c0, c1, d0, d1, e0, e1)
  begin
    if (a = '1') then
      r <= c0 or c1;
      s <= d0 and d1;
    else
      r <= e0 xor e1;
    end if;
  end process;

  process begin
    wait until rising_edge(a);
    t <= b0 xor b1;
    u <= not t;
    v <= not x;
  end process;

  process begin
    wait until rising_edge(a);
    case l is
      when "00" =>
        w <= b0 and b1;
        x <= '0';
      when "01" =>
        w <= '-';
        x <= '1';
      when others =>
        w <= c0 xor c1;
        x <= '-';
    end case;
  end process;

  y <= c0 xor c1;
  z <= x xor w;

end main;
```

1.4 Delta-Cycle Simulation: Pong

Perform a *delta-cycle simulation* of the following VHDL code by drawing a waveform diagram.

NOTES:
1. End your simulation just before 20 ns.

```
architecture main of pong_machine is
  signal ping_i, ping_n, pong_i, pong_n : std_logic;
begin

  reset_proc: process
    reset <= '1';
    wait for 10 ns;
    reset <= '0';
    wait for 100 ns;
  end process;

  clk_proc: process
    clk <= '0';
    wait for 10 ns;
    clk <= '1';
    wait for 10 ns;
  end process;

  next_proc: process (clk)
  begin
    if rising_edge(clk) then
      ping_n <= ping_i;
      pong_n <= pong_i;
    end if;
  end process;

  comb_proc: process (pong_n, ping_n, reset)
  begin
    if (reset = '1') then
      ping_i <= '1';
      pong_i <= '0';
    else
      ping_i <= pong_n;
      pong_i <= ping_n;
    end if;
  end process;

end main;
```


Time
Sim rounds
Sim cycles

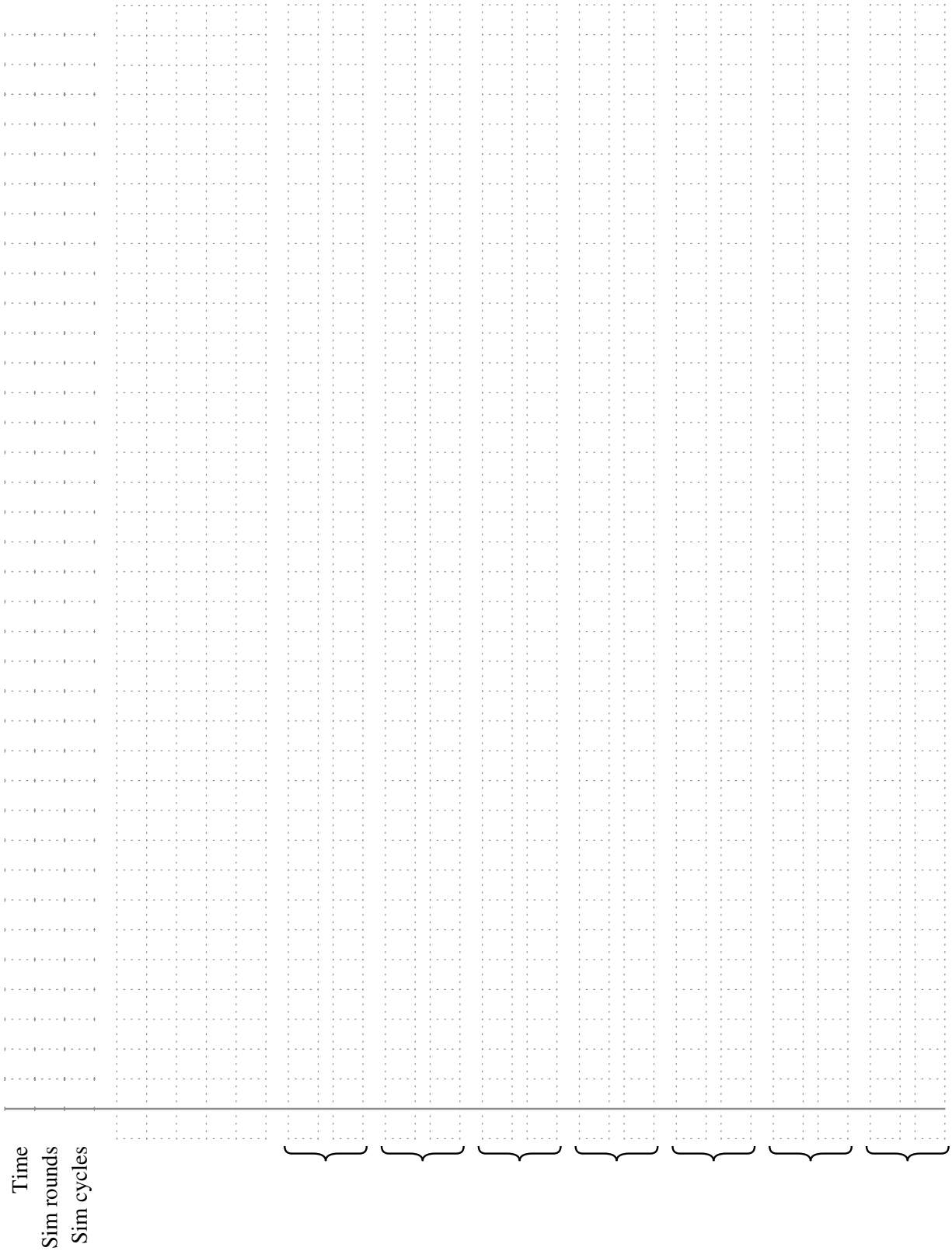
1.5 Delta-Cycle Simulation: Befuddle

Perform a *delta-cycle simulation* of the following VHDL code by drawing a waveform diagram.

NOTES:

1. Begin your simulation at 5 ns using the value shown for each signal.
2. End your simulation just before 15 ns;

```
entity befuddle is
end entity;

architecture main of befuddle is
  signal clk, a, b, c, d, e, f : std_logic;
begin
  proc_clk: process
  begin
    clk <= '0';
    wait for 10 ns;
    clk <= '1';
    wait for 10 ns;
  end process;

  proc_extern : process
  begin
    a <= '0';
    b <= '0';
    wait for 5 ns;
    a <= '1';
    b <= '1';
    wait for 15 ns;
  end process;

  proc_1 : process (a, b, c)
  begin
    c <= a and b;
    d <= a xor c;
  end process;

  proc_2 : process
  begin
    wait until rising_edge(clk);
    e <= d;
  end process;

  proc_3 : process (c, e) begin
    f <= c xor e;
  end process;

end architecture;
```


Time
Sim rounds
Sim cycles

1.6 Synflopsys Simulation Algorithm

You have been hired by 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*.

1.6.1 Advantage

What would be the most important advantage of the Synflopsys simulator over a standard delta-cycle simulator?

1.6.2 Compatibility

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?

1.6.3 Zero-Delay Simulation Rules

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

1.7 VHDL — VHDL Behavioural Comparison: Teradactyl

For each of the VHDL architectures `q3a` through `q3c`, does the signal `v` have the same behaviour as it does in the `main` architecture of `teradactyl`?

NOTES: 1) **For full marks, if the code has different behaviour, you must explain why.**
2) Ignore any differences in behaviour in the first few clock cycles that is caused by initialization of flip-flops, latches, and registers.
3) All code fragments in this question are legal, synthesizable VHDL code.

```
entity teradactyl is
  port (
    a : in std_logic;
    v : out std_logic
  );
end teradactyl;
architecture main of teradactyl is
  signal m : std_logic;
begin
  m <= a;
  v <= m;
end main;
```

```
architecture q3a of teradactyl is
  signal b, c, d : std_logic;
begin
  b <= a;
  c <= b;
  d <= c;
  v <= d;
end q3a;
```

```
architecture q3b of teradactyl is
  signal m : std_logic;
begin
  process (a, m) begin
    v <= m;
    m <= a;
  end process;
end q3b;
```

```
architecture q3c of teradactyl is
  signal m : std_logic;
begin
  process (a) begin
    m <= a;
  end process;
  process (m) begin
    v <= m;
  end process;
end q3c;
```

1.8 The Good, the Bad, and the Synthesizably Challenged

For each of the code fragments below, answer the following questions.

NOTES:
1. Is the code legal VHDL?

2. If the code is legal VHDL:
   (a) Answer whether the code is synthesizable.
   (b) If the code is synthesizable, answer whether it adheres to good coding practices, according to the guidelines for E&CE 327.
3. If the the code is not legal, not synthesizable, or does not follow good coding practices, explain why.

P1.8.1

```
process begin
  wait until rising_edge(clk);
  if (sum = 5) or (reset = '1') then
    state <= S0;
    sum   <= to_unsigned(0, 8);
  else
    sum <= a + b;
    if state = S0 then
      state <= S1;
    end if;
  end if;
end process;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation if illegal, unsynthesizable, or bad practice:

P1.8.2

```
process (reset, sum) begin
  if (sum = 5) or (reset = '1') then
    state <= S0;
  else
    state <= S1;
  end if;
end process;
process (state, a, b) begin
  if state = S0 then
    sum <= to_unsigned(0, 8);
  else
    sum <= a + b;
  end if;
end process;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation if illegal, unsynthesizable, or bad practice:

P1.8.3

```
process begin
  wait until rising_edge(clk);
  if (sum = 5) or (reset = '1') then
    state <= S0;
  else
    state <= S1;
  end if;
end process;
gen0 : if state = S0 generate
  sum <= to_unsigned(0, 8);
end generate;
gen1 : if state /= S0 generate
  sum <= a + b;
end generate;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation if illegal, unsynthesizable, or bad practice:

P1.8.4

```
process begin
  wait until rising_edge(clk);
  if reset = '1' then
    state <= S0;
    sum   <= to_unsigned(0, 8);
  end if;
end process;
process (clk) begin
  if rising_edge(clk) then
    state <= next_state;
  end if;
end process;
process (sum) begin
  if sum = 5 then
    next_state <= S0;
  else
    next_state <= S1;
  end if;
end process;
process (state, a, b) begin
  if state = S0 then
    sum <= to_unsigned(0, 8);
  else
    sum <= a + b;
  end if;
end process;
```

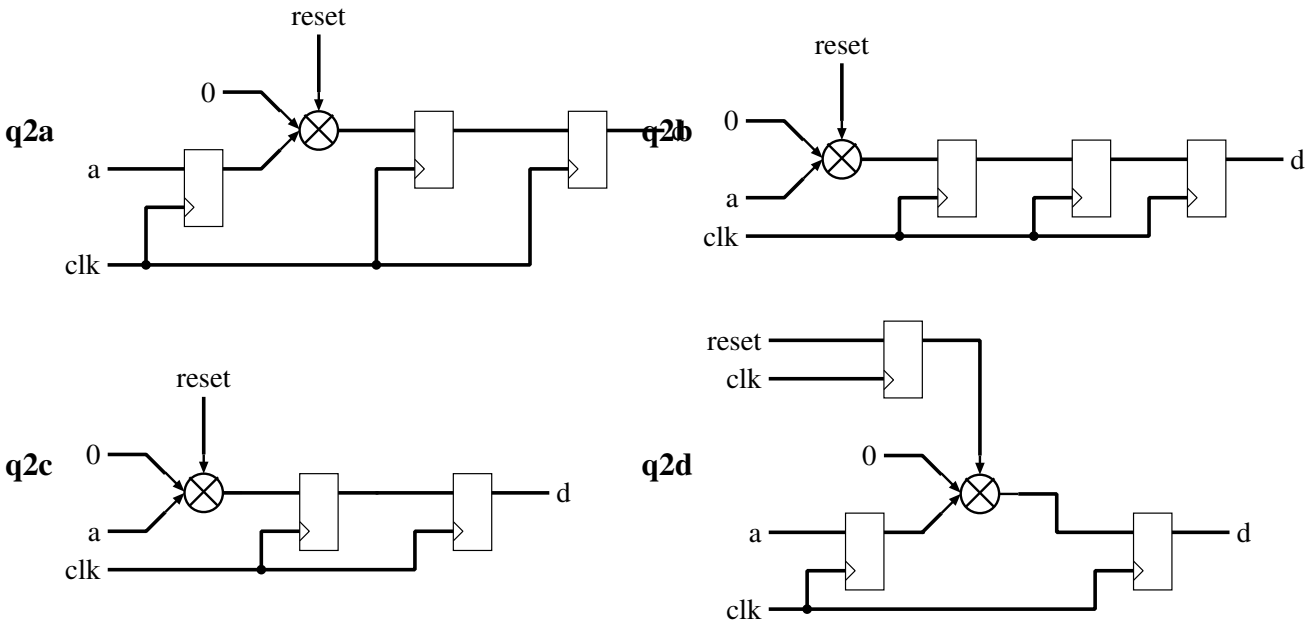
	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation if illegal, unsynthesizable, or bad practice:

1.9 Hardware — VHDL Comparison

For each of the circuits q2a–q2d, answer whether the signal `d` has the same behaviour as it does in the main architecture of q2.

```
entity q2 is
  port (
    a, clk, reset : in std_logic;
    d             : out std_logic
  );
end q2;
architecture main of q2 is
  signal b, c : std_logic;
begin
  b <=   '0' when (reset = '1')
    else a;
  process (clk) begin
    if rising_edge(clk) then
      c <= b;
      d <= c;
    end if;
  end process;
end main;
```



Chapter 2

Additional Features of VHDL

The material in this chapter is useful for the labs. It will *not* be tested on exams.

Chapter 3

Overview of FPGAs

3.1 2-bit adder

This question compares an FPGA and generic-gates implementation of 2-bit full adder.

3.1.1 Generic Gates

Show the implementation of a 2 bit adder using NAND, NOR, and NOT gates.

3.1.2 FPGA

Show the implementation of a 2 bit adder using generic FPGA cells; show the equations for the lookup tables.

3.2 Number of FPGA Cells for Arithmetic Code

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

NOTES:

1. The signals `ab_sel` and `cd_sel` are `std_logic`.
2. The signals `a`, `b`, `c`, `d`, `e`, `k`, `m`, `n`, `p`, and `z` are 12-bit `unsigned`.
3. The inputs to the system are: `a`, `b`, `c`, `d`, `e`, `ab_sel`, `cd_sel`,
4. The output from the system is: `z`.
5. Optimizations are allowed, so long as the externally visible input-to-output behaviour of the system does not change.
6. For full marks, you must justify your answer with a drawing and/or text.

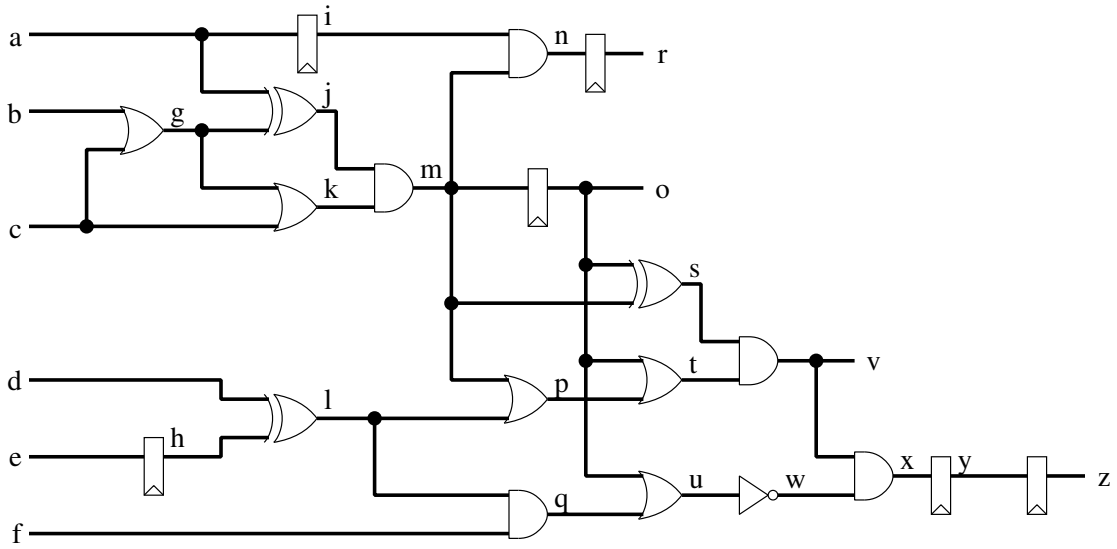
```
k <= a when ab_sel = '1' else b;
m <= c when cd_sel = '1' else d;
process (clk) begin
  if rising_edge(clk) then
    n <= k + m;
    p <= e;
    z <= n + p;
  end if;
end process;
```

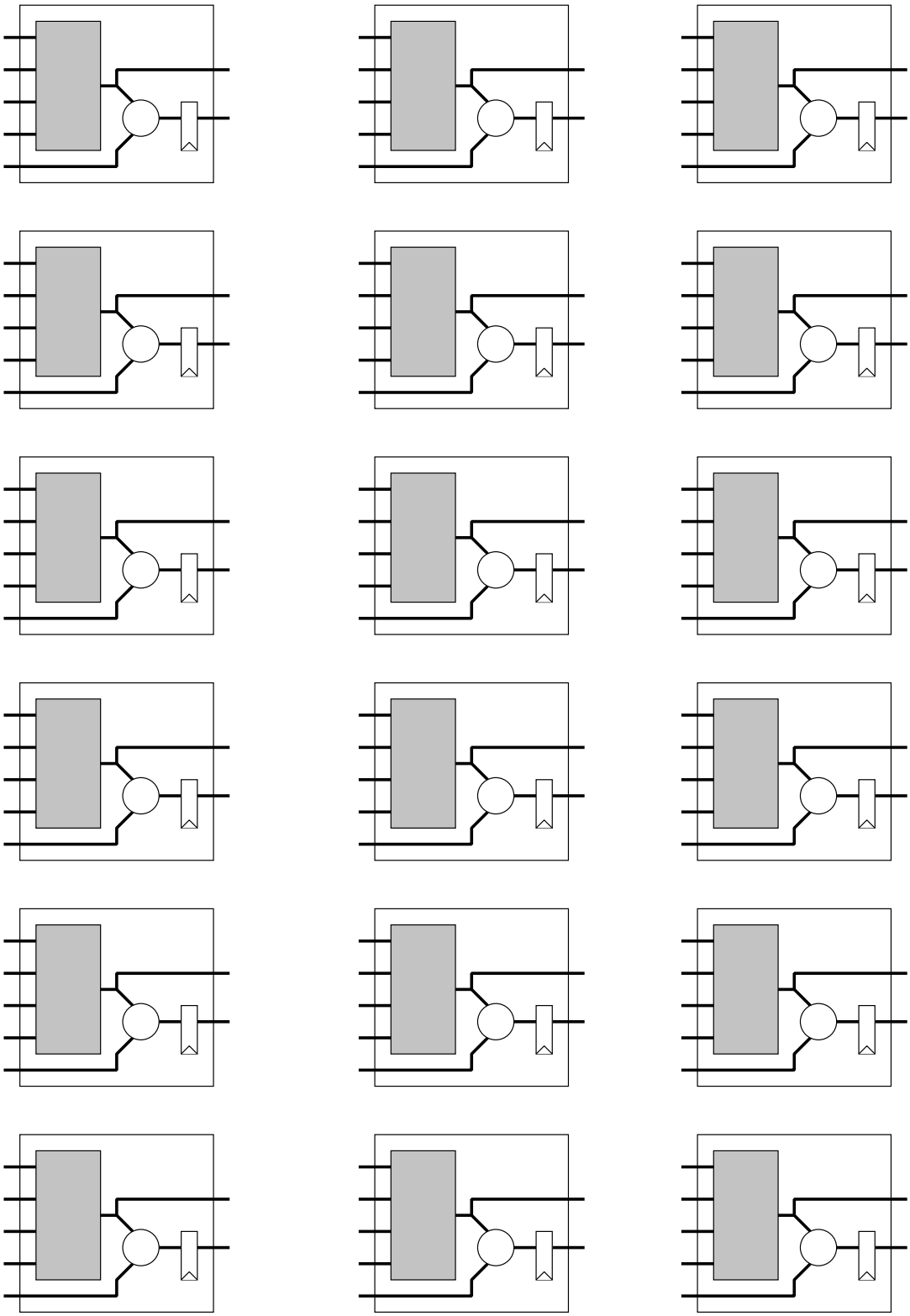
3.3 Number of FPGA Cells for Schematic

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.
2. The primary outputs of the circuit are: r, o, v, 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 don't use.
   - Show the configuration for the internal multiplexer by connecting either the PLA or input port of the cell to the flop.





Chapter 4

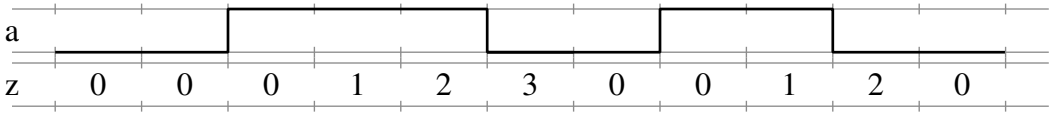
Introduction to RTL Design

4.1 FSM: A-Count

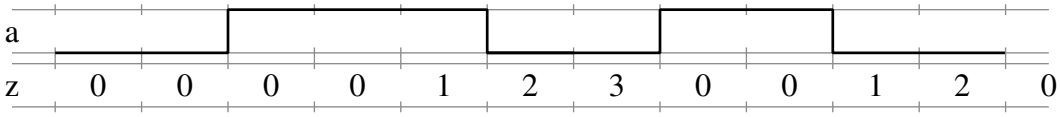
The state-machines below are supposed to count the number of contiguous clock cycles in which `a='1'`.

- Assume that `a='0'` for the first three clock cycles.
- For each state-machine, answer whether the machine works correctly.
- If the machine works correctly, answer what the *latency* through the system is.
- If the machine does *not* work correctly, describe how its behaviour differs from the specification.

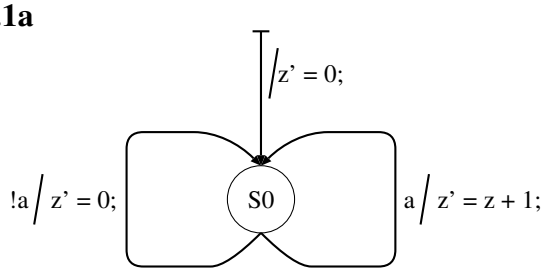
Sample waveform for a machine with a latency of 1 clock cycle.



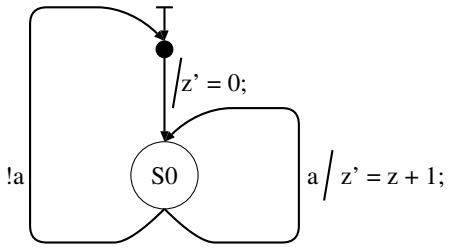
Sample waveform for a machine with a latency of 2 clock cycles.



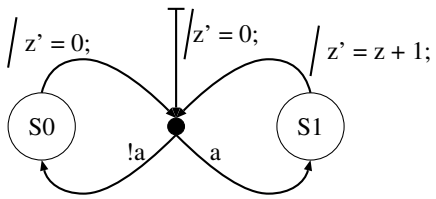
4.1a



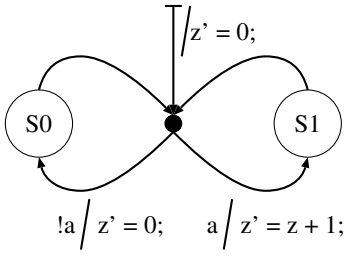
4.1b



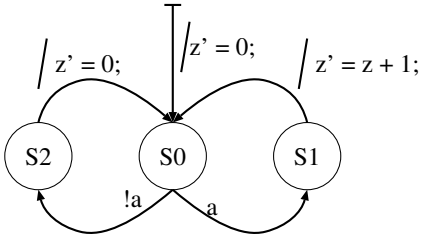
4.1c



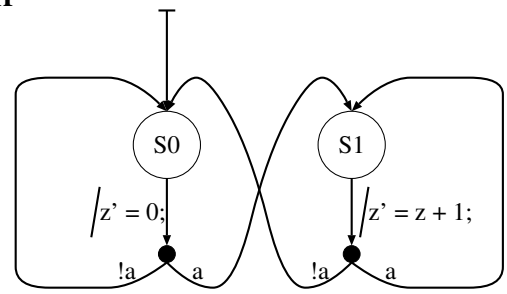
4.1d



4.1e



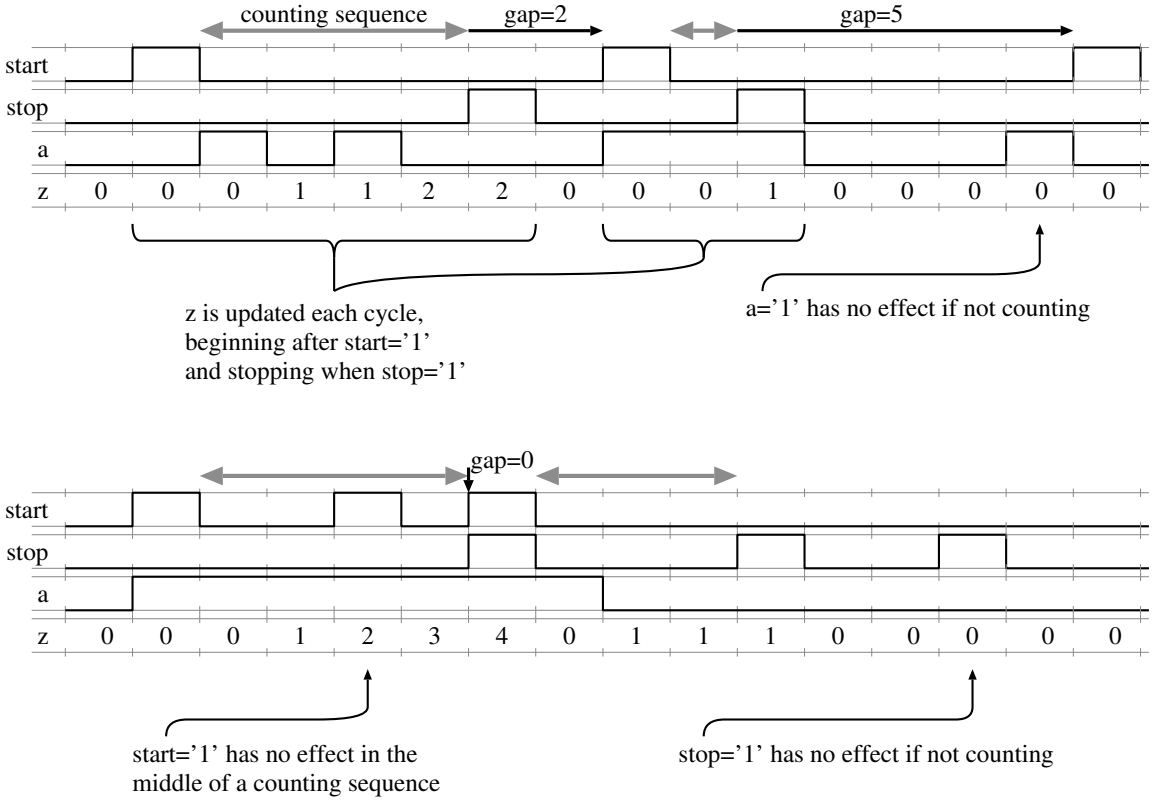
4.1f



4.2 Start-Stop A-Count

Each of the three state machines on the next page is intended to meet the system description below:

1. The inputs to the system are: `start`, `stop`, and `a`. The output is `z`.
2. In the first clock cycle, `start` and `stop` are both guaranteed to be `'0'`.
3. The counting shall begin in the clock cycle *after* `start='1'`.
4. The counting shall continue up to, but *not* including, the next clock cycle in which `stop='1'`.
5. The current values of `start`, `stop`, and `a` shall affect `z` in the *next* clock cycle.
6. In the clock cycle *after* `stop='1'`, the output `z` shall be set to 0 and shall remain 0 until the next sequence of counting begins.
7. The system may put a constraint on the inputs such that there is a *minimum gap* of clock cycles between `stop='1'` and the next `start='1'`. Each state machine may have its own value for the minimum gap. The minimum gap may be as small as 0 clock cycles.



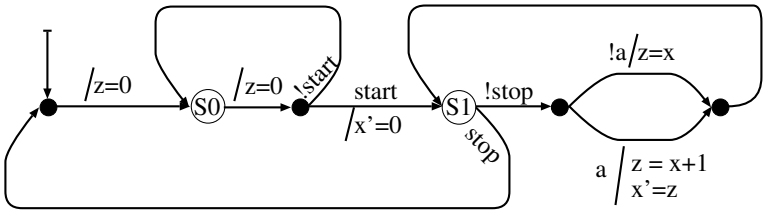
NOTES:

1. If the state machine is correct, answer what the minimum gap value is.
2. If the state machine is incorrect, explain either how the machines behaviour differs from the system description or how the state machine could be modified to fix the incorrect behaviour.

4.2.1

Answer:
Incorrect.

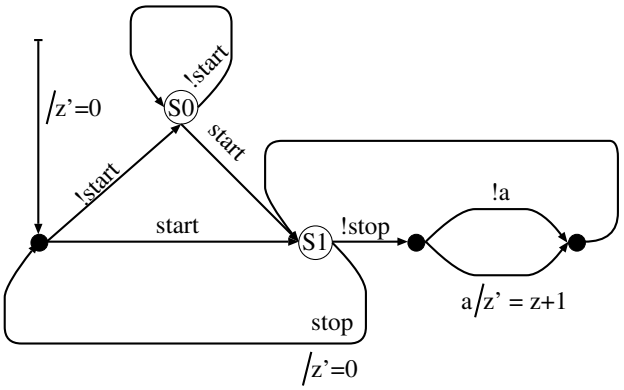
Explanation: When `a='1'`*, the output* `z` *is updated in the* current *clock cycle. It should be updated in the next clock cycle.*



4.2.2

Answer:
Correct. Gap=0.

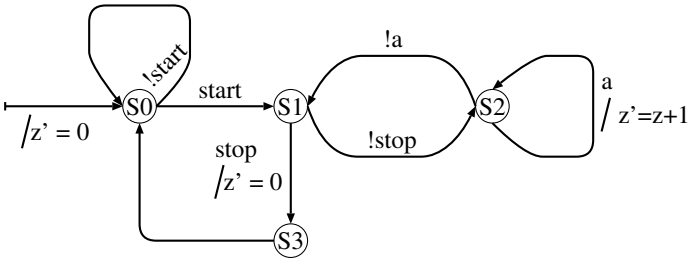
Explanation: `z` *is initialized in the first clock cycle and when* `stop='1'`*. When* `stop='1'`, `start` *is sampled in the same clock cycle.*



4.2.3

Answer:
Incorrect.

Explanation (required): The input `a` *is not sampled every clock cycle during the counting sequence.*

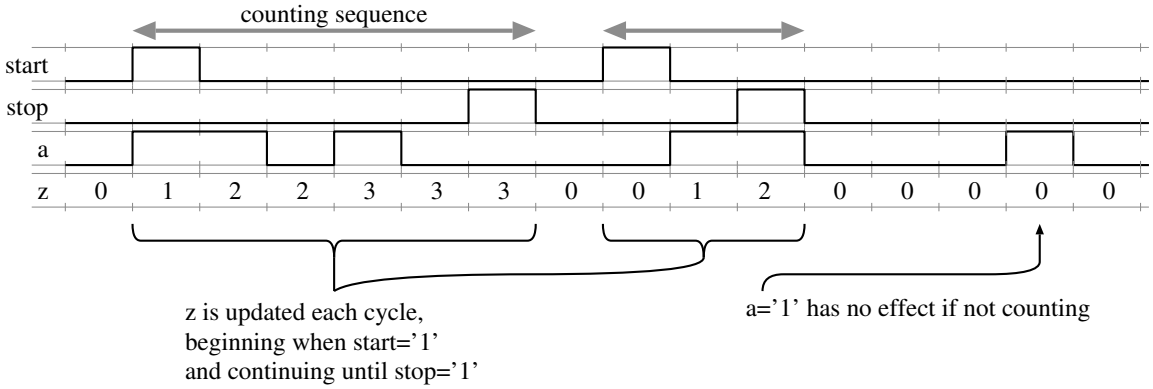


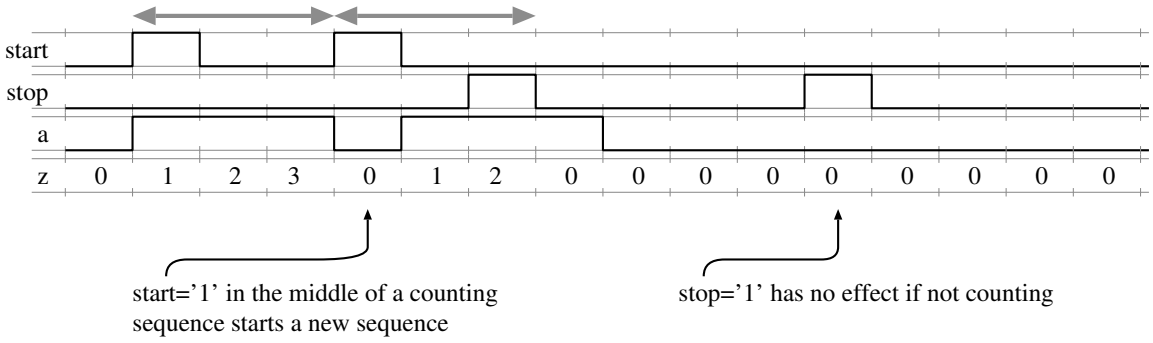
4.3 Count-Sequence Design

Design a state machine that counts the number of times `a='1'`, beginning when `start='1'` and ending when `stop='1'`.

NOTES:

1. The inputs to the system are: `start`, `stop`, and `a`. The output is `z`.
2. In the first clock cycle, `start` and `stop` are both guaranteed to be `'0'`.
3. In the first clock cycle, the output `z` shall be set to `0`.
4. The counting shall begin in a clock cycle in which `start='1'`.
5. The counting shall continue up to and including a clock cycle in which `stop='1'`.
6. The output `z` shall be updated according to the values of `start`, `stop`, and `a` in the current clock cycle.
7. In the clock cycle after `stop='1'`, the output `z` shall be set to 0 and shall remain 0 until the next sequence of counting begins.
8. Marks will be based on correct functionality, elegance of the design, and clarity of the drawing.



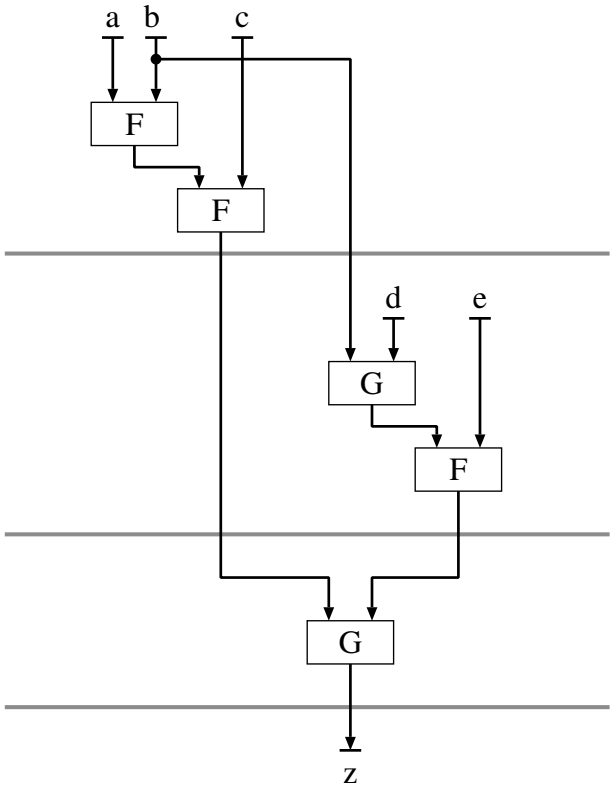


Chapter 5

Dataflow Diagrams

5.1 Dataflow Diagram Optimization

Use the dataflow diagram below to answer sections 5.1.1 and 5.1.2.



5.1.1 Resource Usage

List the number of items for each resource used in the dataflow diagram.

5.1.2 Optimization

Draw an optimized dataflow diagram that reduces the total execution time and produces the same output values without increasing the area. Or, if the performance cannot be improved without increasing area, describe the limiting factor on the performance.

NOTES:

- you may change the times when signals are read from the environment
- you may **not** increase the resource usage (input ports, registers, output ports, f components, g components)
- you may **not** increase the clock period

5.2 Michener: Design and Optimization

Design a circuit named michener that performs the following operation: `z = (a+d) + ((b - c) - 1)`

NOTES:

1. Inputs shall be combinational.
2. Outputs shall be registered.
3. Optimize your design for area.
4. You may schedule the inputs to arrive at any time.
5. You may do algebraic transformations of the specification.

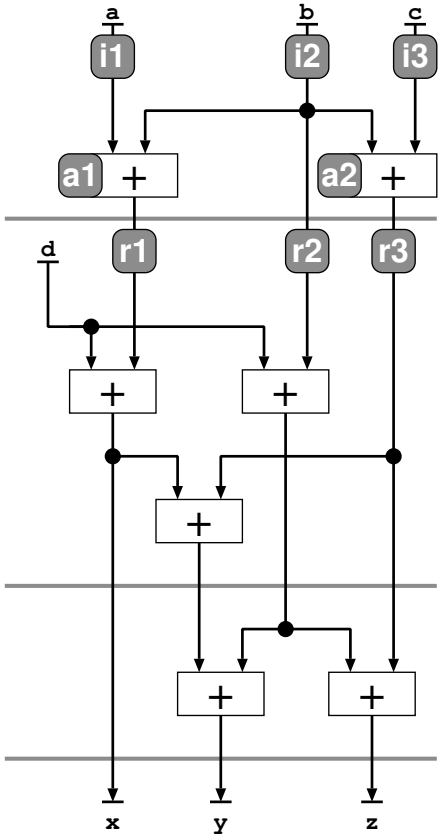
5.3 Allocation and Control Table

5.3.1 Allocation

For the dataflow diagram below, perform: input/output allocation, datapath allocation, and register allocation.

- You may *not* change the allocations that have already been done.

- You may use 2:1 multiplexers, and may combine 2:1 multiplexers to create larger multiplexers. (e.g. two 2:1 muxes can be combined to create a 3:1 multiplexer)
- Optimization goals in order of highest priority to lowest:
  1. Minimize the number of adders
  2. Minimize the number of input and output ports
  3. Minimize the number of registers
  4. Minimize the number of 2:1 multiplexers (including the 2:1 muxes used to build larger multiplexers)
- You may *not* perform any algebraic or scheduling optimizations on the dataflow diagram.



Number of input ports	
Number of output ports	
Number of adders	
Number of registers	
Number of 2:1 multiplexers	

5.3.2 Control Tables

Draw the register and datapath control tables for your dataflow diagram. Leave the "don't care" symbols in the table, that is, do *not* do don't-care optimization.

5.4 Control Table Optimization

In this question, you will optimize control tables, then design the hardware that implements the control table. section 5.4.1 is for a one-hot encoding and section 5.4.2 is for a binary encoding.

NOTES:

1. Inputs: `i1` and `i2`. Data registers: `r1`, `r2`, and `r3`. Datapath components: `add1`, `sub1`.
2. State signal: `st` (see encoding information with section 5.4.1 and section 5.4.2).
3. Your first goal is to minimize area, then optimize for clock speed without increasing area.
4. Your circuit may be implemented with *any number* of the following gates, where the area of each gate is shown in the table below:

Gate	**Area**	**Gate**	**Area**
= (2-bits)	8	AND	2
= (4-bits)	20	OR	2
		NOT	1

5. You may **not** change the behaviour of either the registers or the outputs of the adder or subtracter.
6. You may **not** add any registers.
7. As a guide when filling out your answers, the blank control tables contain the values from the original table printed in grey.

Original control table:

	r1		r2		r3		add1		sub1	
	d	ce	d	ce	d	ce	src1	src2	src1	src2
S0	i1	1	–	0	i2	1	r3	r2	–	–
S1	r3	1	add1	1	–	0	r1	r3	r2	add1
S2	r3	1	–	–	i2	1	r3	r1	add1	r2
S3	–	–	add1	1	r1	1	r1	r3	–	–

In each of section 5.4.1 and section 5.4.2, after optimizing the control table, use the VHDL functions AND, OR, and NOT to write expressions to implement each signal in terms of the state signal and, if helpful, the other control signals.

NOTES:

1. For the "else" case of if-then-elses used as multiplexers, write "others".
2. If a control signal or case for a mux is not needed, write "N/A".

Example

To illustrate how to write your answers, this example shows a sample answer and the equivalent VHDL code, which you do *not* have to write.

	Your Answer			Equivalent VHDL
r99_d_sel	r99 <= r80 when	:	not(a and b)	r99_d_sel = not(a and b); ...

	r99 <= r81 when	:	others	if r99_d_sel = '1' then
				  r99 <= r80;
				else
				  r99 <= r81;
				end if;

5.4.1 One-Hot Encoding

```
subtype state_ty : std_logic_vector( 3 downto 0 );
signal st : state_ty;
constant S0 : state_ty := "0001";
constant S1 : state_ty := "0010";
constant S2 : state_ty := "0100";
constant S3 : state_ty := "1000";
```

Optimized control table:

	r1		r2		r3		add1		sub1	
	d	ce	d	ce	d	ce	src1	src2	src1	src2
S0	i1	1	–	0	i2	1	r3	r2	–	–
S1	r3	1	add1	1	–	0	r1	r3	r2	add1
S2	r3	1	–	–	i2	1	r3	r1	add1	r2
S3	–	–	add1	1	r1	1	r1	r3	–	–

Implementation expressions:

Signal	Case	Expression
r1_d_sel	r1 <= i1 when	:
	r1 <= r3 when	:
r1_ce		:
r2_d_sel	r2 <= add1 when	:
r2_ce		:
r3_d_sel	r3 <= i2 when	:
	r3 <= r1 when	:
r3_ce		:
add1_src1_sel	add1_src1 <= r1 when	:
	add1_src1 <= r3 when	:
add1_src2_sel	add1_src2 <= r1 when	:
	add1_src2 <= r2 when	:
	add1_src2 <= r3 when	:
sub1_src1_sel	sub1_src1 <= r2 when	:
	sub1_src1 <= add1 when	:
sub1_src2_sel	sub1_src2 <= r2 when	:
	sub1_src2 <= add1 when	:

5.4.2 Binary Encoding

```
subtype state_ty : std_logic_vector( 1 downto 0 );
signal st : state_ty;
constant S0 : state_ty := "00";
constant S1 : state_ty := "01";
constant S2 : state_ty := "10";
constant S3 : state_ty := "11";
```

Optimized control table:

	r1		r2		r3		add1		sub1	
	d	ce	d	ce	d	ce	src1	src2	src1	src2
S0	i1	1	–	0	i2	1	r3	r2	–	–
S1	r3	1	add1	1	–	0	r1	r3	r2	add1
S2	r3	1	–	–	i2	1	r3	r1	add1	r2
S3	–	–	add1	1	r1	1	r1	r3	–	–

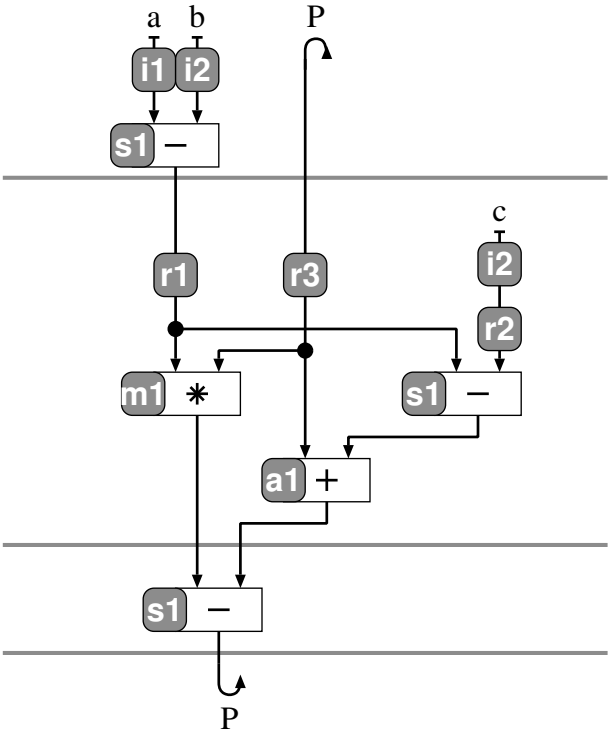
5.5 Inter-Parcel Variables

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)`

5.5.1 Register Allocation

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



5.5.2 Dataflow diagram analysis

Fill in the boxes in the table below.

Latency	
Throughput	
Clock period	

5.5.3 Control Table

Draw the control table.

NOTES:

1. The system shall support an *unpredictable number of bubbles* between valid parcels.

5.5.4 State Encoding

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

Q How many states does the system have?

5.6 Code Review

You are supervising an intern from some UTher school and are responsible for reviewing the intern's VHDL code. The intern has compiled the `count_gt_63` program on the next page, but has not simulated or synthesized it yet. Your task is to write comments that describe the five most important changes that should be made to the code.

NOTES:

1. The purpose of the code is to receive a sequence of 256 data values and count the number of data values that are greater than 63.
2. The input data are unsigned 8-bit numbers.
3. For each data value, `i_valid` is asserted ('1') for exactly one clock cycle, followed by 0 or more clock cycles of bubbles.
4. The input data `i_data` is valid only when `i_valid` is asserted.
5. The types of `i_data` and `o_count` must be `std_logic_vector`.
6. If a comment applies to multiple signals or processes, list the comment just once.
7. Keep the comments focused and try to preserve as much of the intern's code as possible. Try to avoid large structural changes to the code, such as "Combine the three processes into a single process."
8. If you cannot find five changes that will improve the code, then give positive comments about the good features in the code.

```
1)  library ieee;
2)  use ieee.std_logic_1164.all;
3)  use ieee.numeric_std.all;
4)
5)  entity count_gt is
6)    port (
7)      reset, clk, i_valid : in  std_logic;
8)      i_data              : in  std_logic_vector( 7 downto 0 );
9)      o_done              : out std_logic;
10)     o_count             : out std_logic_vector( 7 downto 0 )
11)   );
12) end entity;
13)
14) architecture main of count_gt is
15)   signal gt_count, data_count : unsigned( 7 downto 0 );
16) begin
17)
18)   process (clk) begin
19)     if rising_edge( clk ) then
20)       if reset = '1' then
21)         data_count <= (others => '0');
22)       elsif i_valid = '1' then
23)         data_count <= data_count + 1;
24)       else
25)         data_count <= data_count;
26)       end if;
27)     end if;
28)   end process;
29)
30)   process (clk) begin
31)     if rising_edge( clk ) then
32)       if reset = '1' then
33)         gt_count <= (others => '0');
34)       elsif unsigned(i_data) > 63 then
35)         gt_count <= gt_count + 1;
36)       end if;
37)     end if;
38)   end process;
39)
40)   process begin
41)     wait until rising_edge( clk );
42)     if reset = '1' then
43)       o_done <= '0';
44)     elsif data_count >= 256 then
45)       o_done <= '1';
46)     end if;
47)   end process;
48)
49)   o_count <= std_logic_vector(gt_count);
50)
51) end architecture;
```

The table below lists the different categories of comments in their order of importance, from most important to least important.

Each category has a key (B, S, A, C, or P). For each comment, write the key of the comment's category. If a comment fits into more than one category, write down the most important category that the comment fits into.

Key	Category
B	*bug* fixes
S	making the code *synthesizable*
A	decrease the *area*
C	making the *code* simpler and more elegant
P	a *positive* comment

5.7 Sketches of Problems

1. calculate resource usage for a dataflow diagram (input ports, output ports, registers, datapath components)
2. calculate performance data for a dataflow diagram (clock period and number of cycles to execute (CPI))
3. given a dataflow diagram, calculate the clock period that will result in the optimum performance
4. given an algorithm, design a dataflow diagram
5. given a dataflow diagram, design the datapath and finite state machine
6. optimize a dataflow diagram to improve performance or reduce resource usage
7. given fsm diagram, pick VHDL code that "best" implements diagram — correct behaviour, simple, fast hardware — or critique hardware

Chapter 6

Advanced Design

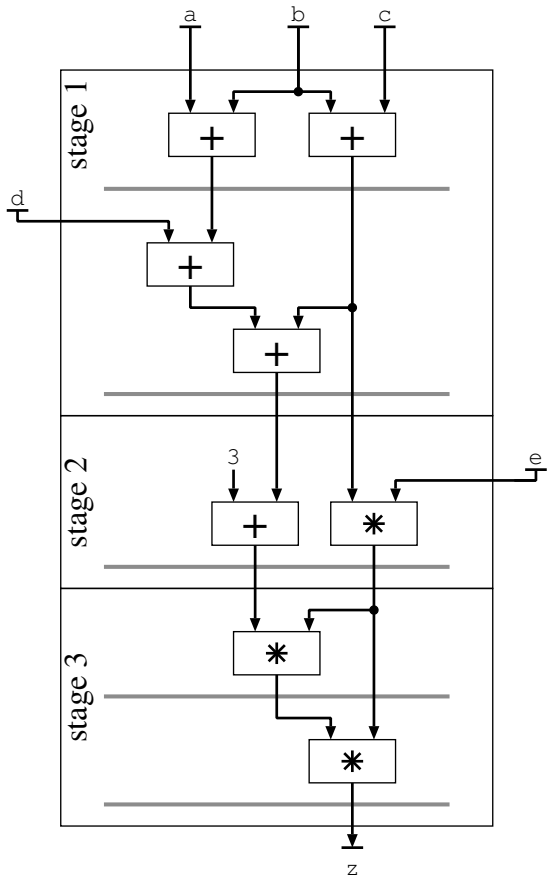
6.1 Pipelining and Dataflow Diagrams

6.1.1 Resource Usage

For the pipelined dataflow diagram below, answer the given questions.

6.1.2 Optimization

Optimize the dataflow diagram to reduce the latency without hurting any other design parameter. Or, describe the constraints that prevent you from improving the latency. You must keep the same data-dependency graph as the original dataflow diagram



Latency	
Throughput	
Clock period	
Number of input ports	
Number of output ports	
Number of adders	
Number of multipliers	
Number of registers	

6.2 Overlapping Pipeline Stages

Your task is to design a dataflow diagram for the following system:

Requirements:

1. Implement the equation $z = \max(a+b, b+c, c+d)$
2. Maximum of 1 input port
3. Minimum throughput of 1/4
4. Maximum clock period of flop + MAX(add, max), where "MAX" is the mathematical function and "max" is the hardware component.
5. Combinational inputs
6. Registered outputs

Goals:

1. Minimum area
2. Minimum latency

Chapter 7

Performance Analysis and Optimization

7.1 Farmer

A farmer is trying to decide which of his two trucks to use to transport his apples from his orchard to the market.

Facts:

	capacity of truck	speed when loaded with apples	speed when unloaded (no apples)
big truck	12 tonnes	15kph	38kph
small truck	6 tonnes	30kph	70kph

distance to market	120 km
amount of apples	85 tonnes

NOTES:

1. All of the loads of apples must be carried using the same truck
2. Elapsed time is counted from beginning to deliver first load to returning to the orchard after the last load
3. Ignore time spent loading and unloading apples, coffee breaks, refueling, etc.
4. For each trip, a truck travels either its fully loaded or empty speed.

Question: *Which truck will take the least amount of time and what percentage faster will the truck be?*

Question: *In planning ahead for next year, is there anything the farmer could do to decrease his delivery time with little or no additional expense? If so, what is it, if not, explain.*

7.2 Network and Router

In this question there is a network that runs a protocol called BigLan. You are designing a router called the DataChopper that routes packets over the network running BigLan (i.e. they're BigLan packets).

The BigLan network protocol runs at a data rate of 160 Mbps (Mega **bits** per second). Each BigLan packet contains 100 Bytes of routing information and 1000 Bytes of data.

You are working on the DataChopper router, which has the following performance numbers:

75MHz	clock speed
4	cycles for a byte of either data or header
500	number of additional clock cycles to process the routing information for a packet

7.2.1 Maximum Throughput

Which has a higher maximum throughput (as measured in **data** bits per second — that is only the payload bits count as useful work), the network or your router, and how much faster is it?

7.2.2 Packet Size and Performance

Explain the effect of an increase in packet length on the performance of the DataChopper (as measured in the maximum number of bits per second that it can process) assuming the header remains constant at 100 bytes.

7.3 Performance Short Answer

If performance doubles every two years, by what percentage does performance go up every month? This question is similar to compound growth from your economics class.

7.4 Microprocessors

The `Yme` microprocessor is very small and inexpensive. One performance sacrifice the designers have made is to not include a multiply instruction. Multiplies must be written in software using loops of shifts and adds.

The `Yme` currently ships at a clock frequency of 200MHz and has an average CPI of 4.

A competitor sells the `Y!v1` microprocessor, which supports exactly the same instructions as the `Yme`. The `Y!v1` runs at 150MHz, and the average program is 10% faster on the `Yme` than it is on the `Y!v1`.

7.4.1 Average CPI

Question: *What is the average CPI for the* `Y!v1`*? If you don't have enough information to answer this question, explain what additional information you need and how you would use it?*

A new version of the `Y!`, the `Y!u2` has just been announced. The `Y!u2` includes a multiply instruction and runs at 180MHz. The `Y!u2` publicity brochures claim that using their multiply instruction, rather than shift/add loops, can eliminate 10% of the instructions in the average program. The brochures also claim that the average performance of `Y!u2` is 30% better than that of the `Y!v1`.

7.4.2 Why not you too?

Question: *Assuming the advertising claims are true, what is the average CPI for the* `Y!u2`*? If you don't have enough information to answer this question, explain what additional information you need and how you would use it?*

7.4.3 Analysis

Question: *Which of the following do you think is most likely and why.*

1. the `Y!u2` is basically the same as the `Y!`v1 except for the multiply
2. the `Y!u2` designers made performance sacrifices in their design in order to include a multiply instruction
3. the `Y!u2` designers performed other significant optimizations in addition to creating a multiply instruction

7.5 Multiply Instruction

You are part of the design team for a microprocessor implemented on an FPGA. You currently implement your multiply instruction completely on the FPGA. You are considering using a specialized multiply chip to do the multiplication. Your task is to evaluate the performance and optimality tradeoffs between keeping the multiply circuitry on the FPGA or using the external multiplier chip.

If you use the multipliplier chip, it will reduce the CPI of the multiply instruction, but will not change the CPI of any other instruction. Using the multiplier chips will also force the FPGA to run at a slower clock speed.

	FPGA option	FPGA + MULT option
average CPI	5	???
% of instrs that are multiplies	10%	10%
CPI of multiply	20	6
Clock speed	200 MHz	160 MHz

7.5.1 Highest Performance

Which option, FPGA or FPGA+MULT, gives the higher performance (as measured in MIPs), and what percentage faster is the higher-performance option?

7.5.2 Performance Metrics

Explain whether MIPs is a good choice for the performance metric when making this decision.

7.6 FPGA vs CPU

You are the project leader for a team that will design and implement a Waterluvian filter. (Do not worry if you've never heard of this obscure filter, the details are irrelevant to this question.)

Your task is to decide whether to implement the Waterluvian filter using a component on an FPGA (the *hardware option*) or using a program running on a microprocessor (the *software option*).

The following notes are relevant to both section 7.6.1 and 7.6.2.

NOTES:

1. The average performance of Waterluvian filters on the market doubles every 24 months.
2. Both the input and output of your system will be gray-scale images of 1000×1000 pixels, with 8 bits per pixel.
3. The marketing department predicts that they can sell 20,000 Waterluvian filters.
4. The average cost of Waterluvian filters remains constant.
5. Information for *software option*:
   - Clock speed for microprocessor: 900MHz
   - Cost of microprocessor: $100
   - For each instruction in the microprocessor's instruction set, the table below gives the cycles-per-instruction and the number of each instruction needed to compute one pixel in the output image.

	CPI	**Instrs/Pixel**
Load	1.4	7
Store	1.1	2
Branch	1.2	1
Add	1.0	8
Multiply	2.2	4

   - Development effort (time to market): 3 months
   - Labour cost for development: $100,000 per month
6. Information for *hardware option*:
   - The clock speed of the FPGA has not yet been determined.
   - The cost of the FPGA chip has not yet been determined.
   - The hardware design will produce 1 output pixel per clock cycle.
   - Development effort (time to market): 6 months.
   - Labour cost for development: $100,000 per month.
7. The two options (hardware and software) have the same costs, except for the differences noted above.

7.6.1 FPGA Clock Speed

Ignoring the difference in time-to-market, what clock speed must the FPGA design have in order for the hardware option to have the same performance as the software option?

7.6.2 FPGA cost

Calculate the cost of the FPGA chip such that the hardware option and the software option have the same cost/performance ratio, relative to the average Waterluvian filter on the market, at the time when each option would reach the market.

More formally, define the following:

$$
\begin{aligned}
CP_{avg}(t) &= \text{cost/performance ratio of the average Waterluvian filter on the market at time } t. \\
CP_{hw} &= \text{cost/performance ratio of hardware option} \\
CP_{sw} &= \text{cost/performance ratio of software option}
\end{aligned}
$$

Find the cost of the FPGA chip such that: $\dfrac{CP_{sw}}{CP_{avg}(3months)} = \dfrac{CP_{hw}}{CP_{avg}(6months)}$

If you were unable to answer section 7.6.1, you may assume that the FPGA clock speed is F when answering this question.

7.6.3 Delay in Schedule

Would your answer to the fpga cost change if both the hw and sw schedules were delayed by 2 months? — that is, their development times became 5 months and 8 months, rather than 3 months and 6 months.

7.7 Waterluvian on New FPGA Chip

You are developing a new design of the Waterluvian filter that will be implemented on an FPGA chip (the definition of a Waterluvian filter is irrelevant to this question). Your project leader just returned from a conference where she learned about a new FPGA chip. She has asked you to analyze the tradeoffs between remaining with your current FPGA chip, or delaying the project and switching to the new chip.

- The *optimality* of Waterluvian filters is measured by the ratio of performance to area.
- The current prediction is that the average Waterluvian filter at the time you complete the project will be 25% faster and have 5% more area than your filter with the current chip.
- The average *performance* of Waterluvian filters doubles every 24 months.

- The average *area* of the circuits that implement Waterluvian filters has remained constant over time.
- The new FPGA chip claims that it will provide a 20% improvement in clock speed with just 85% of the area of the FPGA chip you are using now.

If you keep your design the same and switch to the new chip, how long can you delay your project and have an *optimality* that is at least 7% more than the average *optimality* at the time you complete the project?

If you cannot achieve an optimality that is 7% more than average at the time you complete the project, then assume that using the new FPGA chip will delay your project by 4 months, and then calculate the optimality of your filter compared to the average at the time you complete the project.

Chapter 8

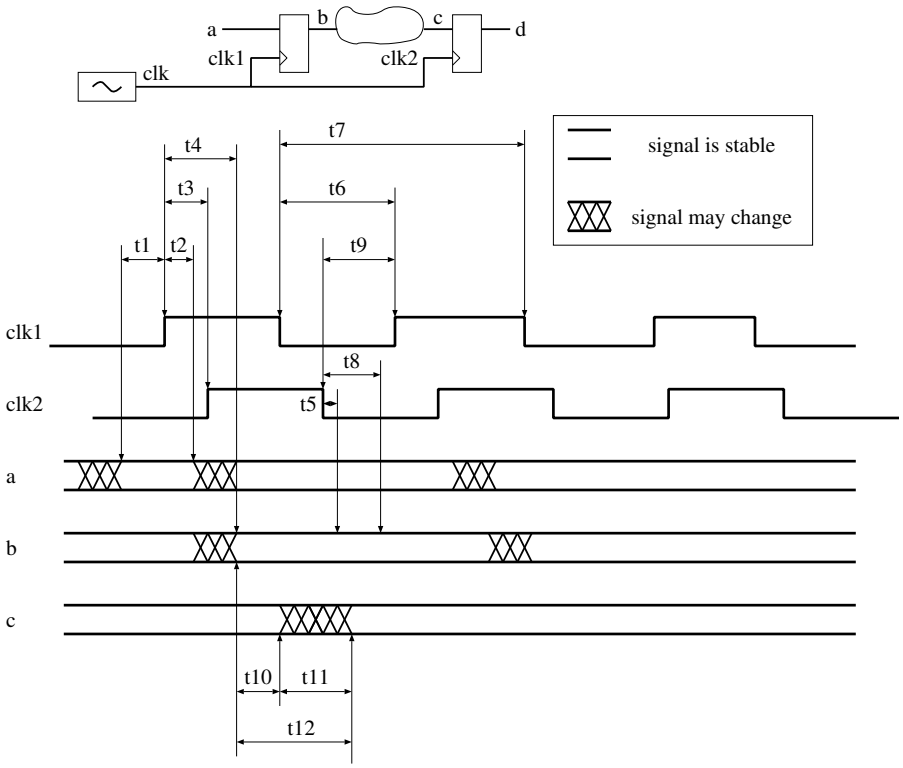
Timing Analysis

8.1 Terminology

For each of the terms: clock skew, clock period, setup time, hold time, and clock-to-q, answer which time periods (one or more of t1 – t9 or NONE) are examples of the term.

NOTES:

1. The timing diagram shows the limits of the allowed times (either minimum or maximum).
2. All timing parameters are non-negative.



8.2 Hold Time Violations

8.2.1 Cause

What is the cause of a hold time violation?

8.2.2 Behaviour

What is the bad behaviour that results if a hold time violation occurs?

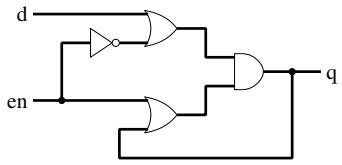
8.2.3 Rectification

If a circuit has a hold time violation, how would you correct the problem with minimal effort?

8.3 Latch Analysis

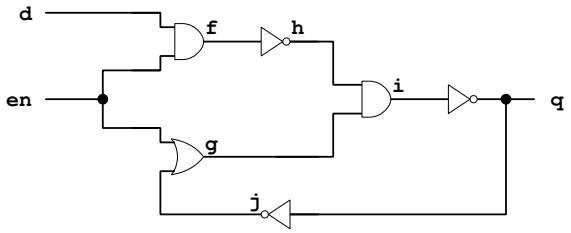
Does the circuit below behave like a latch? If not, explain why not. If so, calculate the clock-to-Q, setup, and hold times; and answer whether it is active-high or active-low.

Gate Delays	
AND	4
OR	2
NOT	1



8.4 Latch Analysis

In this question, you will analyze the circuit below to determine if it is correct implementation of a latch.



NOTES:
1. The delay through each gate is 1 ns.

8.4.1 Good or Bad?

Is the circuit a correct implementation of a latch?

The two remaining parts of this question (8.4.2 and 8.4.3) are divided into two columns. Use the left column if you answered *yes*. Use the right column if you answered *no* above.

8.4.2 Analysis

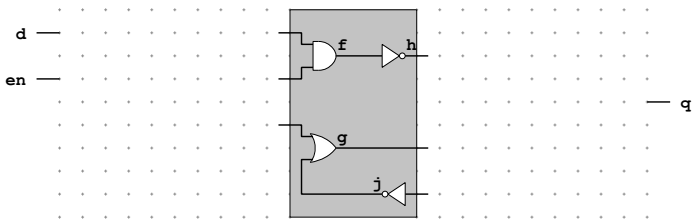
If the circuit is a correct implementation of a latch, determine if it is active-high or active-low and calculate the timing parameters below.

Active-high or active-low?	
Clock-to-Q	
Setup	
Hold	

If the circuit is *not* a correct implementation of a latch, explain why.

8.4.3 Modification

If the circuit is a correct implementation of a latch, modify the circuit using the diagram below to *decrease the setup time* by 2 ns.
NOTES:
1. You may *not* modify the part of the circuit in the gray rectangle.
2. If you cannot decrease the delay by 2 ns, then decrease the delay to the minimum amount such that the latch still works correctly.



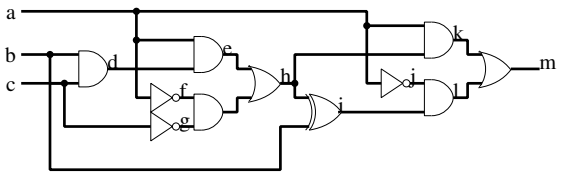
If the circuit is *not* a correct implementation of a latch, modify the circuit using the diagram below to turn it into a correctly functioning latch.
NOTES:
1. Make the minimum modifications necessary.
2. You may *not* modify the part of the circuit in the gray rectangle.

If the circuit is a correct implementation of a latch, for each of the timing parameters below, answer whether your modification to the latch increases, does not change, or decreases the value of the timing parameter.

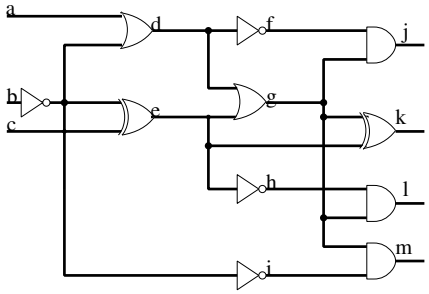
	Increase	No change	Decrease
Clock-to-Q	☐	☐	☐
Hold	☐	☐	☐

8.5 Critical Path and False Path

Find the critical path through the following circuit:



8.6 Critical Path



gate	delay
NOT	2
AND	4
OR	4
XOR	6

Assume all delay and timing factors other than combinational logic delay are negligible.

8.6.1 Longest Path

List the signals in the longest path through this circuit.

8.6.2 Delay

What is the combinational delay along the longest path?

8.6.3 Missing Factors

What factors that affect the maximum clock speed does your analysis for parts 1 and 2 not take into account?

8.6.4 Critical Path or False Path?

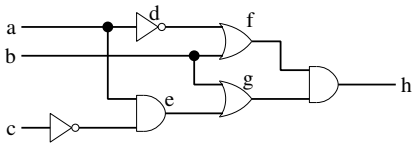
This is an advanced section.
It is not covered in the course
and will not be tested.

Is the longest path that you found a real critical path, or a false path? If it is a false path, find the real critical path. If it is a critical path, find a set of assignments to the primary inputs that exercises the critical path.

8.7 YACP: Yet Another Critical Path

This is an advanced section.
It is not covered in the course
and will not be tested.

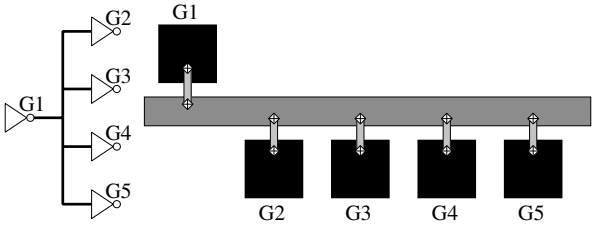
Find the critical path in the circuit below.



8.8 Timing Models

In your next job, you have been told to use a "fanout" timing model, which states that the delay through a gate increases linearly with the number of gates in the immediate fanout. You dimly recall that a long time ago you learned about a timing model named Elmo, Elmwood, Elmore, El-Morre, or something like that.

For the circuit shown below as a schematic and as a layout, answer whether the fanout timing model closely matches the delay values predicted by the Elmore delay model.



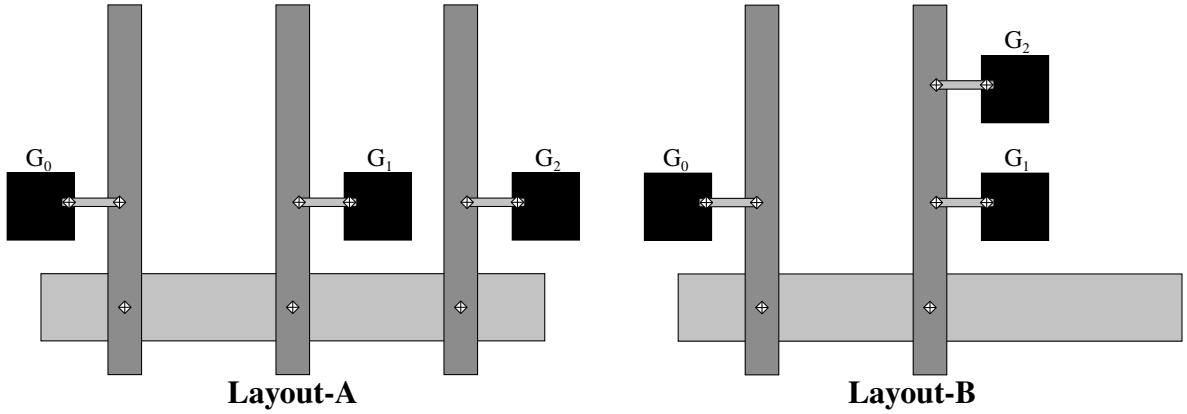
Symbol	Description	Capacitance	Resistance
	Interconnect level 2	Cx	0
	Interconnect level 1	Cy	0
	Gate	Cg	0
	Antifuse	0	R

Assumptions:

- The capacitance of a node on a wire is independent of where the node is located on the wire.

8.9 Elmore Analysis of Super-Vias

In this question you will analyze the 2 layouts below using the Elmore delay model.



Layout-A **Layout-B**

Symbol	Description	Capacitance	Resistance
	Interconnect level 3	C_X	0
	Interconnect level 2	C_Y	0
	Interconnect level 1	0	0
	Gate	C_L	0
	Antifuse	0	R

8.9.1 Minimum Delay

Which of the layouts has the least delay from G_0 to G_2? **Justify your answer in terms of the Elmore delay model.**

NOTES:

1. If you do not have sufficient information to answer the question, find an equation for the relationship between C_X, C_Y, and C_L such that the two layouts have the same delay (An example equation is: $C_X = C_Y + C_L$).
2. The capacitance of a wire is independent of distance and location on the wire.
3. Write your answer on the next page

8.9.2 Alternative layout

In Layout A, how would the delay from G_0 to G_2 be affected by swapping the location of G_1 and G_2? **Justify your answer in terms of the Elmore delay model.**

8.9.3 Low-res antifuse

The FPGA layout rules allow one of the antifuses to be a special low-resistance antifuse which has significantly less resistance than a normal antifuse. For layout-A, which antifuse would you choose to have low-resistance to minimize the delay from G_0 to G_2?

NOTES:

1. Indicate your choice for the low-resistance antifuse by circling it in the picture below

8.10 Zeraf

Your next work term is with a large FPGA company whose place-and-route software is written in Toronto. This company has just developed a new antifuse technology named ZERAF (zero resistance antifuse), that reduces dramatically the resistance through antifuses, or vias. The reduction is so great that antifuse resistance is now negligible in comparison to the resistance through wires and gates.

The current place-and-route software is based upon the assumption that resistance through wires is negligible compared to the resistance through antifuses. Your job is to modify the place-and-route software to take into account the ZERAF technology.

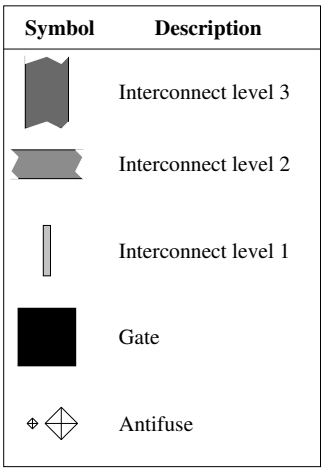
Your first task is to analyze the circuit layouts A and B shown below.

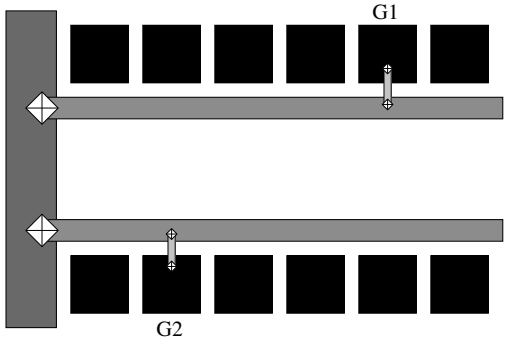
On FPGAs without ZERAF, layout A and B have approximately the same delay. The company wants to know whether the use of ZERAF will cause one of the layouts to have a smaller delay than the other.

Answer whether on an FPGA with ZERAF, the delay for layout A is likely to be the same as the delay for layout B. If the delays are different, decide which layout will have the smaller delay.

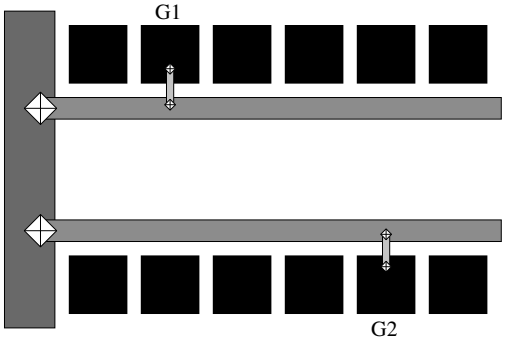
NOTES:

1. G1 is the source gate and G2 is the load gate.
2. The capacitance of a node on a wire is independent of the location of the node on the wire.
3. The resistance between two nodes on a wire is proportional to the distance between them.
4. The resistance of level-1 interconnect is negligible, because the wires are extremely short.
5. For the circuits shown, the resistance of the level-3 interconnect is negligible, because the cross-section of the wire is so large in comparison to the distance between the antifuses on the wire.





Layout A



Layout B

8.11 Short Answer

8.11.1 Wires in FPGAs

In an FPGA today, what percentage of the clock period is typically consumed by wire delay?

8.11.2 Age and Time

If you were to compare a typical digital circuit from 5 years ago with a typical digital circuit today, would you find that the percentage of the total clock period consumed by capacative load has increased, stayed the same, or decreased?

8.11.3 Temperature and Delay

As temperature increases, does the **delay** through a typical combinational circuit increase, stay the same, or decrease?

8.12 Worst Case Conditions and Derating Factor

Assume that we have a 'Std' speed grade Actel A1415 (an ACT 3 part) Logic Module that drives 4 other Logic Modules:

8.12.1 Worst-Case Commercial

Estimate the delay under worst-case commercial conditions (assume that the junction temperature is the same as the ambient temperature)

8.12.2 Worst-Case Industrial

Find the derating factor for worst-case industrial conditions and calculate the delay (assume that the junction temperature is the same as the ambient temperature).

8.12.3 Worst-Case Industrial, Non-Ambient Junction Temperature

Estimate the delay under the worst-case industrial conditions (assuming that the junction temperature is 105C).

Chapter 9

Power Analysis and Power-Aware Design

9.1 Short Answers

9.1.1 Power and Temperature

As temperature increases, does the **power** consumed by a typical combinational circuit increase, stay the same, or decrease?

9.1.2 Leakage Power

The new vice president of your company has set up a contest for ideas to reduce leakage power in the next generation of chips that the company fabricates. The prize for the person who submits the suggestion that makes the best tradeoff between leakage power and other design goals is to have a door installed on their cube. What is your door-winning idea, and what tradeoffs will your idea require in order to achieve the reduction in leakage power?

9.1.3 Clock Gating

In what situations could adding clock-gating to a circuit increase power consumption?

9.1.4 Gray Coding

What are the tradeoffs in implementing a program counter for a microprocessor using Gray coding?

9.2 VLSI Gurus

The VLSI gurus at your company have come up with a way to decrease the average rise and fall time (0-to-1 and 1-to-0 transitions) for signals. The current value is 1ns. With their fabrication tweaks, they can decrease this to 0.85ns .

9.2.1 Effect on Power

If you implement their suggestions, and make no other changes, what effect will this have on power? **(NOTE: Based on the information given, be as specific as possible.)**

9.2.2 Critique

A group of wannabe performance gurus claim that the above optimization can be used to improve performance by at least 15%. Briefly outline what their plan probably is, critique the merits of their plan, and describe any affect their performance optimization will have on power.

9.3 Advertising Ratios

One day you are strolling the hallways in search of inspiration, when you bump into a person from the marketing department. The marketing department has been out surfing the web and has noticed that companies are advertising the MIPs/mm^2, MIPs/Watt, and Watts/cm^3 of their products. This wide variety of different metrics has confused them.

Explain whether each metric is a reasonable metric for customers to use when choosing a system.

If the metric is reasonable, say whether "bigger is better" (e.g. 500 MIPs/mm^2 is better than 20 MIPs/mm^2) or "smaller is better" (e.g. 20 MIPs/mm^2 is better than 500 MIPs/mm^2), and which **one** type of product (cell phone, desktop computer, or compute server) is the metric **most** relevant to.

- MIPs/mm^2
- MIPs/Watt
- Watts/cm^3

9.4 Vary Supply Voltage

As the supply voltage is scaled down (reduced in value), the maximum clock speed that the circuit can run at decreases.

The scaling down of supply voltage is a popular technique for minimizing power. The maximum clock speed is related to the supply voltage by the following equation:

$$\mathsf{MaxClockSpeed} \propto \frac{(\mathsf{VoltSup} - \mathsf{VoltThresh})^2}{\mathsf{VoltSup}}$$

Where VoltSup is supply voltage and VoltThresh is threshold voltage.

With a supply voltage of 3V and a threshold voltage of 0.8V, the maximum clock speed is measured to be 200MHz. What will the maximum clock speed be with a supply voltage of 1.5V?

9.5 Clock Gating

You have been asked to design a clock-gating circuit for the circuit described below. Without clock gating, the circuit is 45% over its power budget.

Area of main circuit	200 cells
Activity factor	60%
Percentage of input data that is a parcel	40%
Average length of continuous sequence of parcels	50 clock cycles
Latency	20 clock cycles

9.5.1 Maximum Area

What is the maximum area that your clock gating circuit can use without going over the power budget?

NOTES:

1. If you do not have enough information to answer the question, make reasonable assumptions about the values, or describe the additional information that you need and how you would use it to calculate the answer.
2. If you cannot reduce the power to within the power budget, calculate the minimum percentage over the power budget that you can reduce the power to.

9.5.2 Clock Gating Design

Briefly describe your design for the clock gating circuit. You may describe the design in words, draw a circuit, or use both text and a picture.

9.6 Clock Speed Increase Without Power Increase

The following are given:

- You need to increase the clock speed of a chip by 10%
- You must not increase its dynamic power consumption
- The only design parameter you can change is supply voltage
- Assume that short-circuiting current is negligible

9.6.1 Supply Voltage

How much do you need to decrease the supply voltage by to achieve this goal?

9.6.2 Supply Voltage

What problems will you encounter if you continue to decrease the supply voltage?

9.7 Power Reduction Strategies

In each low power approach described below identify which component(s) of the power equation is (are) being minimized and/or maximized:

9.7.1 Supply Voltage

Designers scaled down the supply voltage of their ASIC

9.7.2 Transistor Sizing

The transistors were made larger.

9.7.3 Adding Registers to Inputs

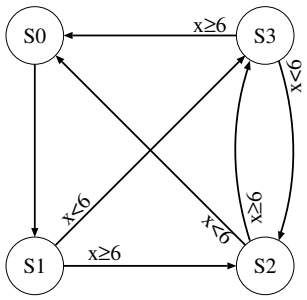
All inputs to functional units are registered

9.7.4 Gray Coding

Gray coding of signals is used for address signals.

9.8 State Encoding

Design a state encoding for the state machine below so that power consumption is minimized.



NOTES:

1. The state machine is to be implemented using generic FPGA cells with a 4-input LUT and a flip-flop.
2. The capacitance of a LUT is twice that of a flip-flop.
3. The signal `x` is declared as `unsigned(2 downto 0)`. All values of `x` are equally likely.
4. You may use a standard encoding, or create a custom encoding.
5. The edges of the diagram are annotated by the condition under which the transition is taken.
6. Your circuit does not need a reset signal.

9.9 Power Consumption on New Chip

While you are eating lunch at your regular table in the company cafeteria, a vice president sits down and starts to talk about the difficulties with a new chip.

The chip is a slight modification of existing design that has been ported to a new fabrication process. Earlier that day, the first sample chips came back from fabrication. The good news is that the chips appear to function correctly. The bad news is that they consume about 10% more power than had been predicted.

The vice president explains that the extra power consumption is a very serious problem, because power is the most important design metric for this chip.

The vice president asks you if you have any idea of what might cause the chips to consume more power than predicted.

9.9.1 Hypothesis

Hypothesize a likely cause for the surprisingly large power consumption, and justify why your hypothesis is likely to be correct.

9.9.2 Experiment

Briefly describe how to determine if your hypothesized cause is the real cause of the surprisingly large power consumption.

9.9.3 Reality

The vice president wants to get the chips out to market quickly and asks you if you have any ideas for reducing their power without changing the design or fabrication process. Describe your ideas, or explain why her suggestion is infeasible.