ECE 327: Digital Systems Engineering
Solutions to Tutorial Problems

2017t1 (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'`.

Answer:

	In `std_logic_1164`?		
	Yes	*No*	*Description*
`'-'`	X		don't care
`'#'`		X	
`'0'`	X		strong 0
`'1'`	X		strong 1
`'A'`		X	
`'h'`		X	
`'H'`	X		weak 1
`'L'`	X		weak 0
`'Q'`		X	
`'X'`	X		strong unknown
`'Z'`	X		high impedance

NOTE: 'h' is not in the package, because characters are case sensitive. For example 'a' is different than 'A'.

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;
```

ILLEGAL: `if-then-else` is a statement, not an expression, you can't put it `if-then-else` on right-hand-side of assignment since it doesn't produce a value to assign to signal a.

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

ILLEGAL: `loop` statements are sequential, while architecture bodies contain concurrent statements.

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

ILLEGAL: condition for `if-generate` statements must be statically determined; testing the value of a signal is dynamic.

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;
```

LEGAL

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;
```

ILLEGAL: the argument to `inv` should be `(a => b)`. In function calls and component instantiations, when using named parameter instantiation (as opposed to positional parameter instantiation), the syntax is *`formal => actual`*. In the problem, the function definition is: `inv( a : std_logic )` and the function call is: `inv( a => b)`. Here `a` is the formal argument and `b` is the actual argument, so the correct function call would be: `inv( a => b )`

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;
```

ILLEGAL: case statements are sequential; but the body of an architecture contains concurrent statements.

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 XOR 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
    case l is
      when "00" =>
        wait until rising_edge(a);
        w <= b0 AND b1;
        x <= '0';
      when "01" =>
        wait until rising_edge(a);
        w <= '-';
        x <= '1';
      when "1-" =>
        wait until rising_edge(a);
        w <= c0 XOR c1;
        x <= '-';
    end case;
  end process;
  y <= c0 XOR c1;
  z <= x XOR w;
end main;
```

Answer:

	Latch	*Combinational*	*Flip-flop*
p		*X*	
q			*X*
r		*X*	
s	*X*		
t			*X*
u			*X*
v			*X*
w			*X*
x			*X*
y		*X*	
z		*X*	

Explanation of why `e`*, which is the output of a flip-flop, have a value at 5ns, which is before the first rising edge of the clock.*

Before the first rising edge of the clock, the following assignments will all happen:

```
a <= '0';
b <= '0';
...
----------- end of delta cycle
d <= '0'
...
----------- end of delta cycle
e <= d;
```

If you were to implement VHDL code in hardware, e would be the output of a flop, and as such would remain as 'U' until the first rising edge of the clock. This is a situation where simulating the VHDL code will have slightly different results than simulating the hardware. Most questions in ece327 that ask you to compare the behaviour of VHDL code with the behaviour of a circuit will say to focus on the steady-state behaviour and ignore any differences in the first few clock cycles.

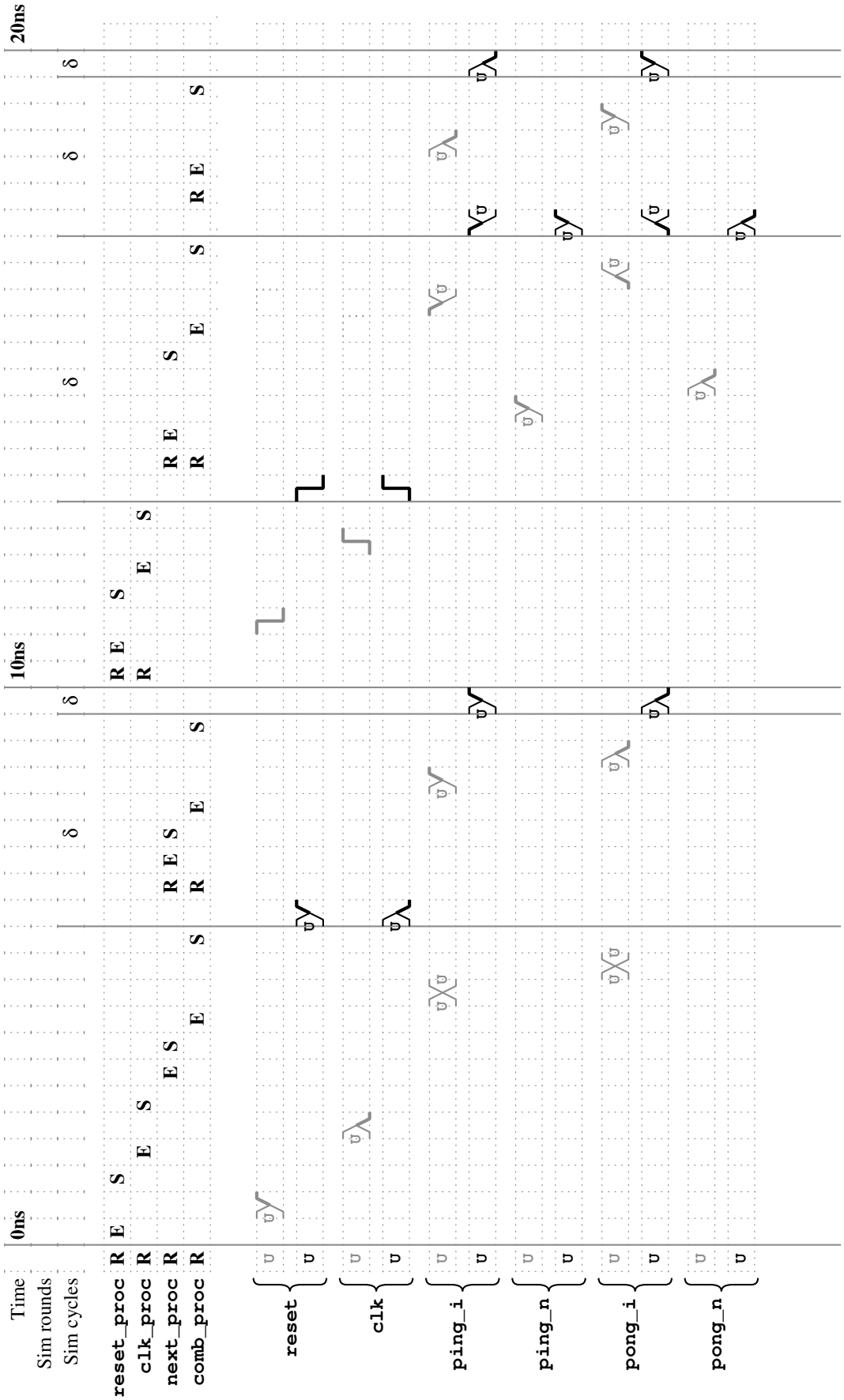
1.4 Delta-Cycle Simulation: Pong

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;
```

Answer:


Time
Sim rounds
Sim cycles
0ns
10ns
20ns
reset_proc
clk_proc
next_proc
comb_proc
reset
clk
ping_i
ping_n
pong_i
pong_n

1.5 Delta-Cycle Simulation: Befuddle

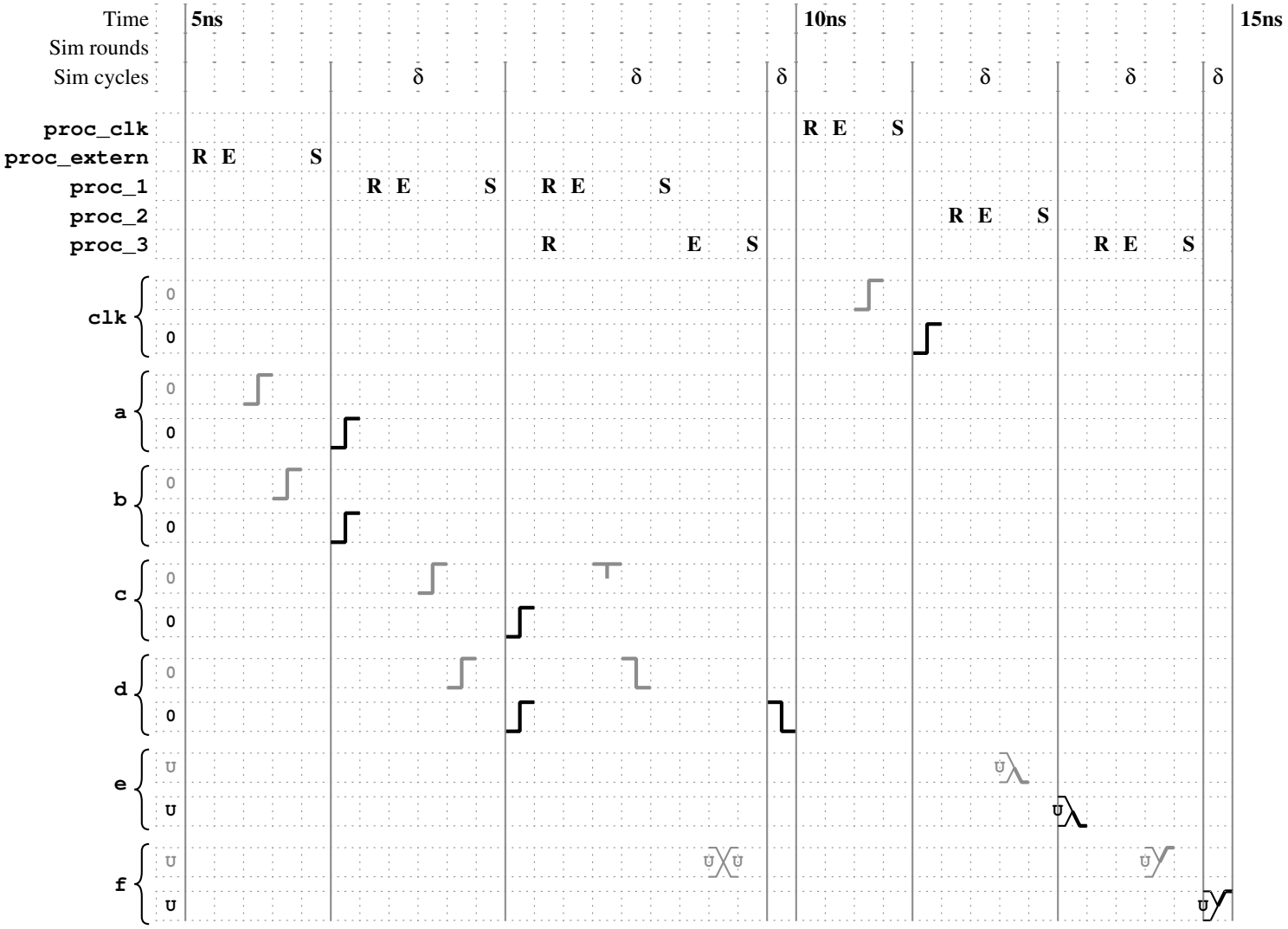
Perform a *delta-cycle simulation* of the following VHDL code by drawing a waveform diagram.

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;
```

Answer:



Note: the instruction to end just before 15 ns simply causes us to stop the simulation just before 15 ns. The values on the signals at the end of 10 ns will remain until the next event, which happens at 20 ns.

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?

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.

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?

Answer:

VHDL programs without combinational loops.

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.

Answer:

Rule 1 *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 2 *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.*

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;
```

SAME - Intermediate signals are optimized out.

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

SAME - Putting it in a process doesn't matter.

```
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;
```

SAME - Putting it in a seperate process doesn't matter due to the parallel nature of VHDL.

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.

1.8.0.1

```
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;
```

Answer:
legal, synth, good

1.8.0.2

```
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;
```

Answer:
legal, synth, bad : comb loop

1.8.0.3

```
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;
```

Answer:

illegal: if-genererate with dynamic condition

1.8.0.4

```
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;
```

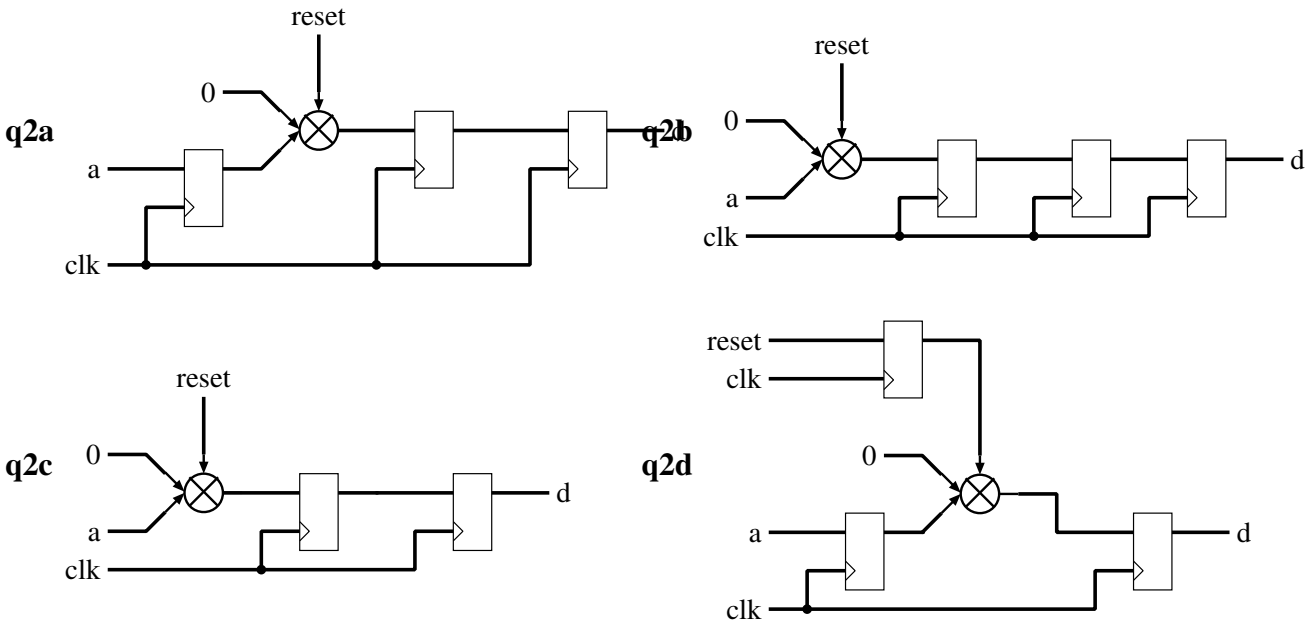
Answer:

legal, synth, bad: two drivers of signals

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;
```



Answer:

q2a: a shouldn't be flopped. q2b: One too many FFs. q2c: Correct operation. q2d: This will work (i.e. it has the same input-output characteristics) but the internal description is different.

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.

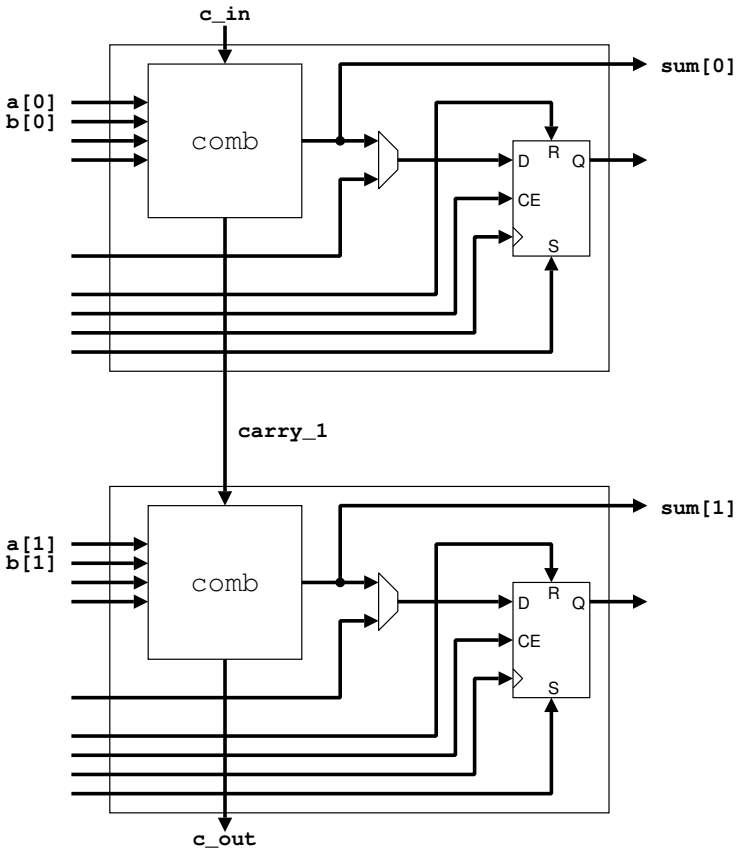
3.1.1 Generic Gates

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.

Answer:


c_in
a[0]
b[0]
comb
sum[0]
D
R
Q
CE
S
carry_1
a[1]
b[1]
comb
sum[1]
D
R
Q
CE
S
c_out

3.2 Number of FPGA Cells for Arithmetic Code

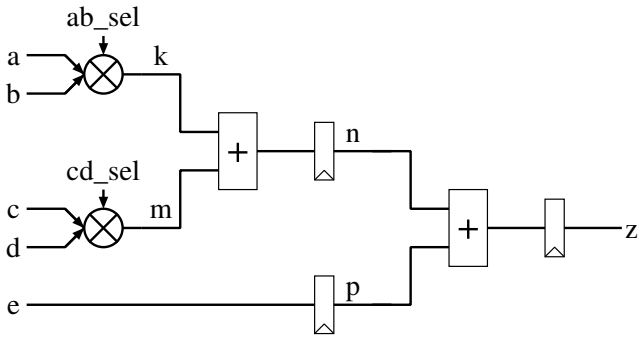
Calculate the minimum number of FPGA cells needed to implement the VHDL code below.

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. 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.

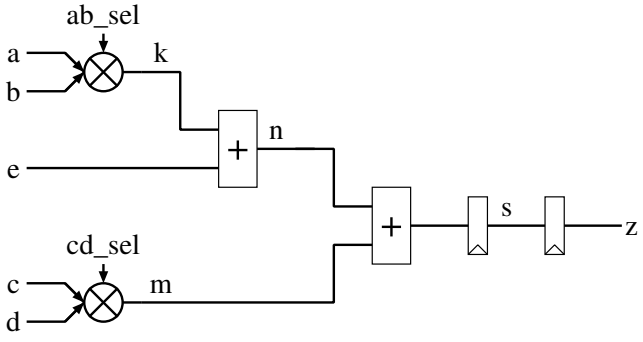
```
k <= a when ab_sel = '1' else b;
m <= c when cd_sel = '1' else d;
process (clk) begin
  if rising_edge(clk) then
    n <= k + m;
    p <= e;
    z <= n + p;
  end if;
end process;
```

Answer:

Original circuit:



Optimized circuit option #1:



signal	circuitry	LUTs per bit	flops per bit	num bits	LUTs total	flops total
`n`	mux and adder	1	0	12	12	0
`s`	mux and adder, reg	1	1	12	12	12
`z`	reg	0	1	12	0	12
				Total	**24**	**24**

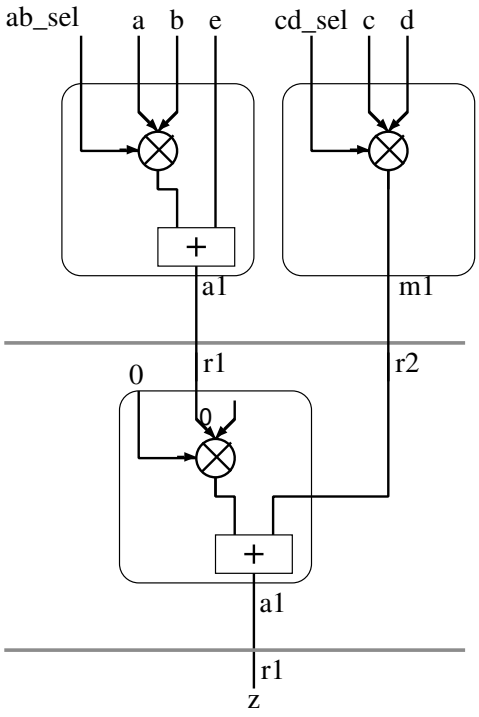
Total number of cells = Max(24 LUTs, 24 flops) = 24 cells.

We can put a 2:1 mux and an adder in the same LUT.

Each cell has a LUT and a flip-flop, so we can fit both the adder and the reg for `z` into one cell per bit.

Optimized circuit option #2:

Reuse the adder and mux in two clock cycles.



signal	circuitry	LUTs per bit	flops per bit	num bits	LUTs total	flops total
`a1`	mux and adder	1	0	12	12	0
`m1`	mux	1	0	12	12	0
`r1`	reg	0	1	12	0	12
`r2`	reg	0	1	12	0	12
				Total	**24**	**24**

3.3 Number of FPGA Cells for Schematic

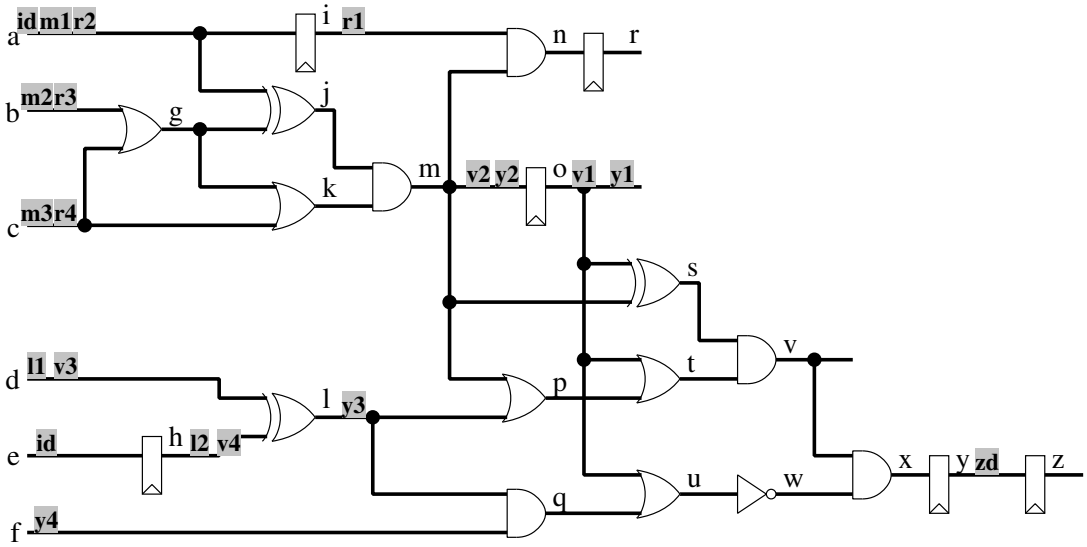
Design an FPGA implementation of the gate-level circuit shown below that uses the minimum number of FPGA cells. Use the FPGA cells on the following page to answer the question.

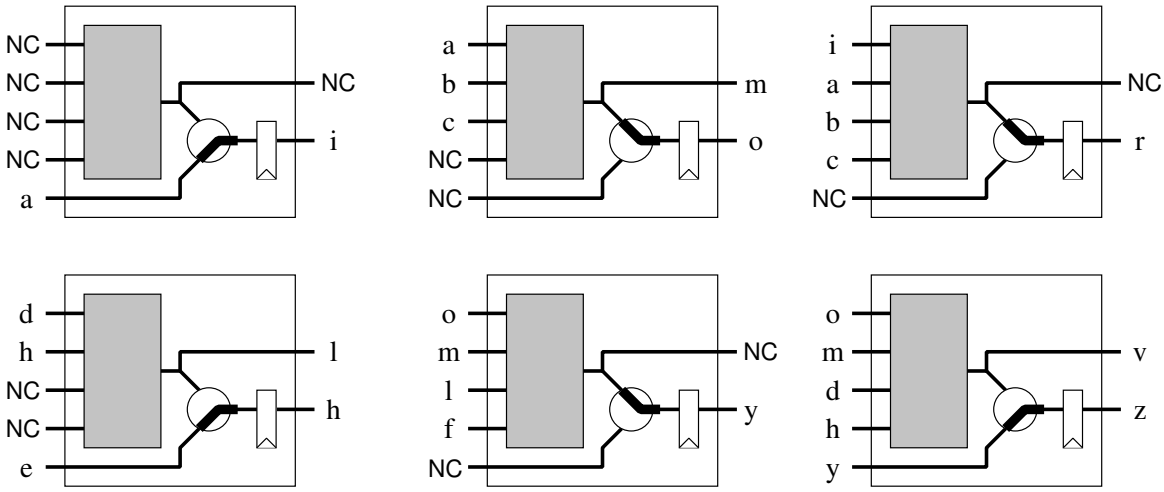
NOTES:

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.

Answer:

The gray boxes describe how the signals are used. For example, "m1" is the first input to the PLA that drives `m` *and "id" is the D-input to the flop that drives* `i`.




NC
NC
NC
NC
a
NC
i
a
b
c
NC
NC
m
o
i
a
b
c
NC
NC
r
d
h
NC
NC
e
l
h
o
m
l
f
NC
NC
y
o
m
d
h
y
v
z

Chapter 4

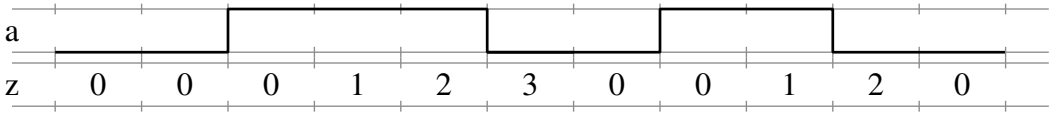
Introduction to RTL Design

4.1 FSM: A-Count

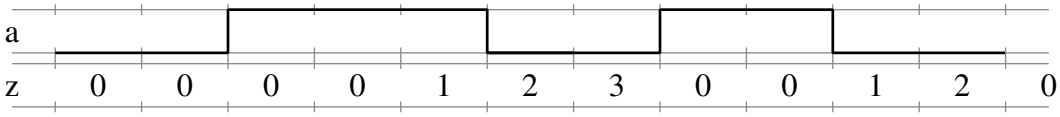
The state-machines below are supposed to count the number of contiguous clock cycles in which `a='1'`.

- 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.

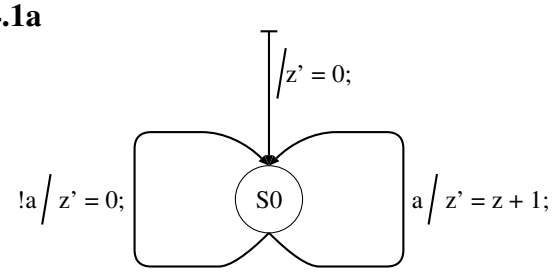
Sample waveform for a machine with a latency of 1 clock cycle.



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

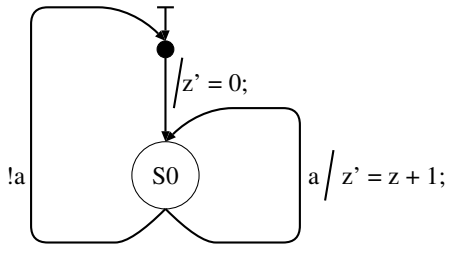


4.1a



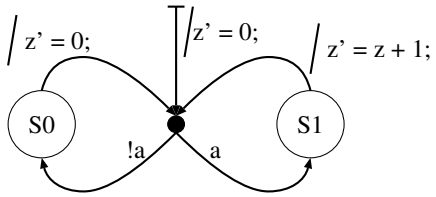
OK, latency = 1

4.1b



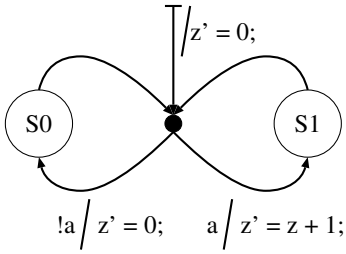
OK, latency = 1

4.1c



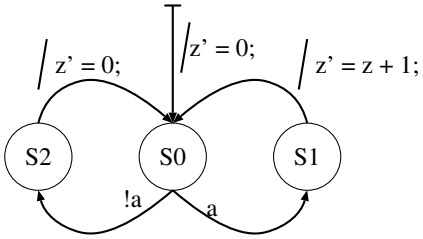
OK, latency = 2

4.1d



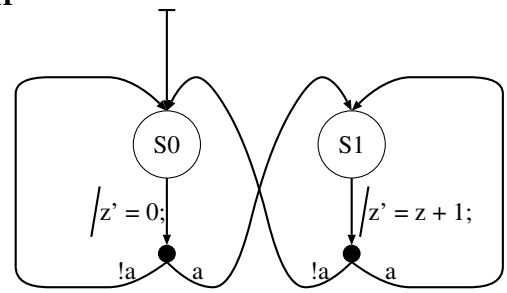
OK, latency = 1. NOTE: relies upon assumption that `a='0'` in first clock cycle, otherwise, `z` is not initialiazed for first sequence of `a='1'`.

4.1e



BUG: samples `a` every-other clock cycle.

4.1f



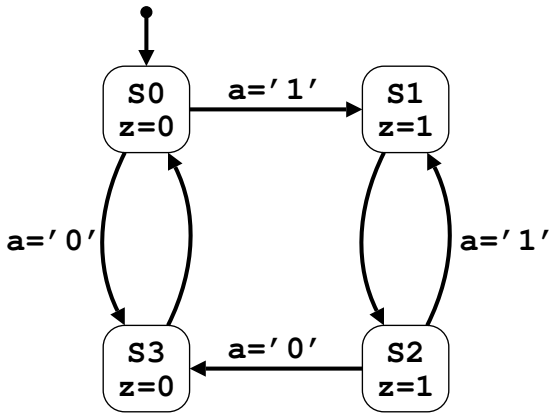
OK: latency = 2

4.2 State machines and VHDL

For each of the code fragments q3a...q3e, answer whether the code has the same input/output behaviour as the state machine below.

NOTES:

1. The signal `a` is the input and `z` is the output.
2. The arrows in the state machine are annotated with the input value that causes them to be taken. The input value of `'-'` means "don't care".
3. The circles in the state machine are annotated by the output value to be produced when in that state: `/0` means output a `'0'` and `/1` means output a `'1'`.
4. All of the code fragments are legal, synthesizable, and follow the E&CE 327 coding guidelines.
5. If the code has different behaviour than the state machine, explain how it differs.
6. Ignore any differences in behaviour in the first few clock cycles of running the VHDL code.



Answer:

We begin by drawing a waveform to analyze the behaviour of the state machine.

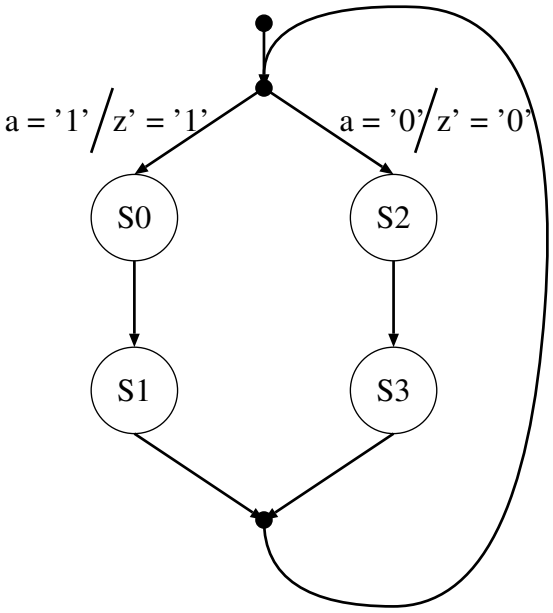
We start in `S0` and arbitrarily choose `a` to be `'1'`, which takes us to `S1`. From `S1` we always go to `S2`. In `S2`, we choose `a` to be `'0'`, so that we explore the previously unvisited state of `S3`. From `S3`, we return to `S0`.

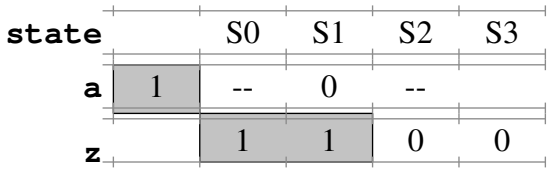
state	S0	S1	S2	S3	S0
a	1	--	0	--	
z		1	1	0	0

4.2.0.1

```
process begin
  if a = '1' then
    z <= '1';
    wait until rising_edge(clk);
    wait until rising_edge(clk);
  else
    z <= '0';
    wait until rising_edge(clk);
    wait until rising_edge(clk);
  end if;
end process;
```

Answer:



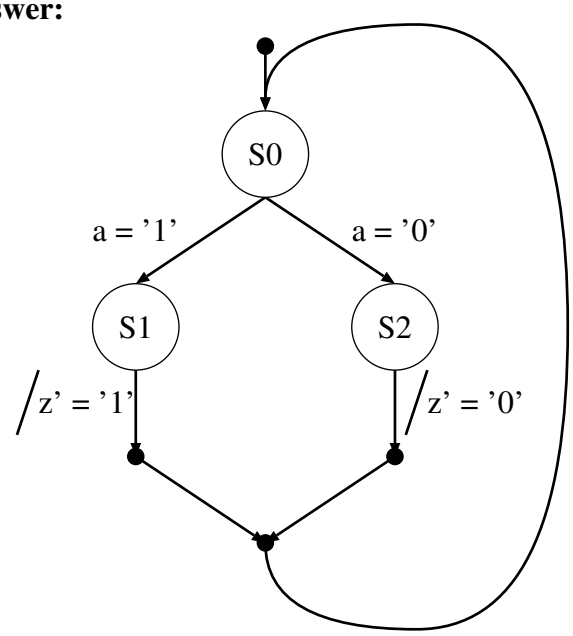


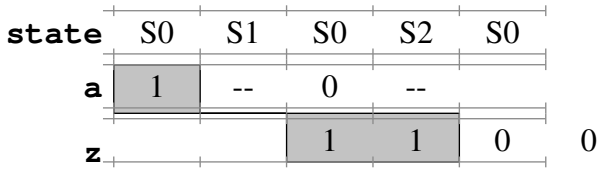
Same

4.2.0.2

```
process begin
  wait until rising_edge(clk);
  if a = '1' then
    wait until rising_edge(clk);
    z <= '1';
  else
    wait until rising_edge(clk);
    z <= '0';
  end if;
end process;
```

Answer:



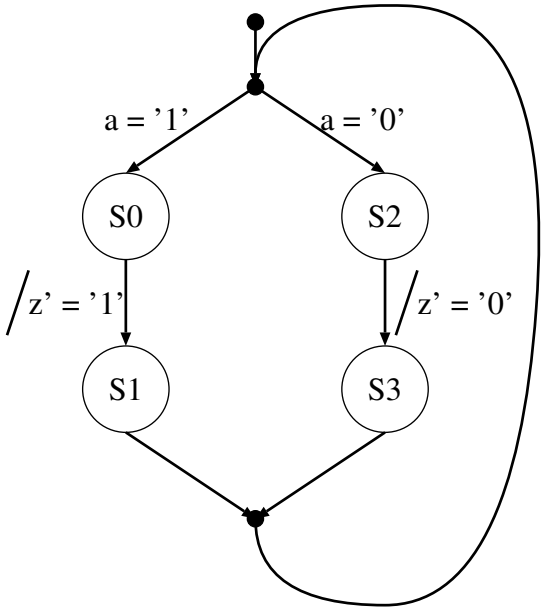


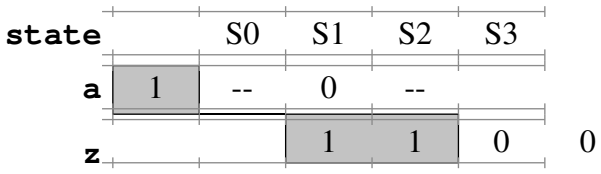
Different. In the VHDL code, there are two clock cycles between when `a` *changes and when* `z` *changes. In the diagram, there is just one clock cycle.*

4.2.0.3

```
process begin
  if a = '1' then
    wait until rising_edge(clk);
    z <= '1';
    wait until rising_edge(clk);
  else
    wait until rising_edge(clk);
    z <= '0';
    wait until rising_edge(clk);
  end if;
end process;
```

Answer:



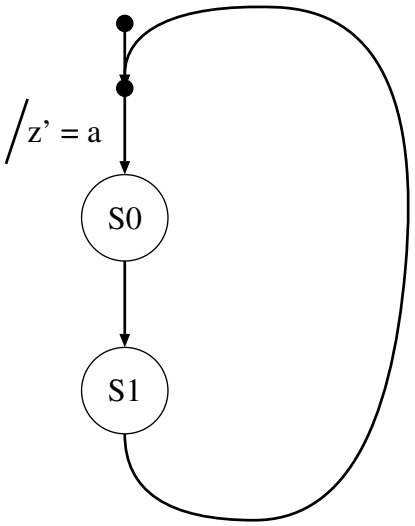


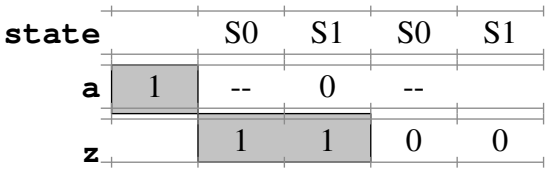
Different. In the VHDL code, there are two clock cycles between when `a` *changes and when* `z` *changes. In the diagram, there is just one clock cycle.*

4.2.0.4

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

Answer:



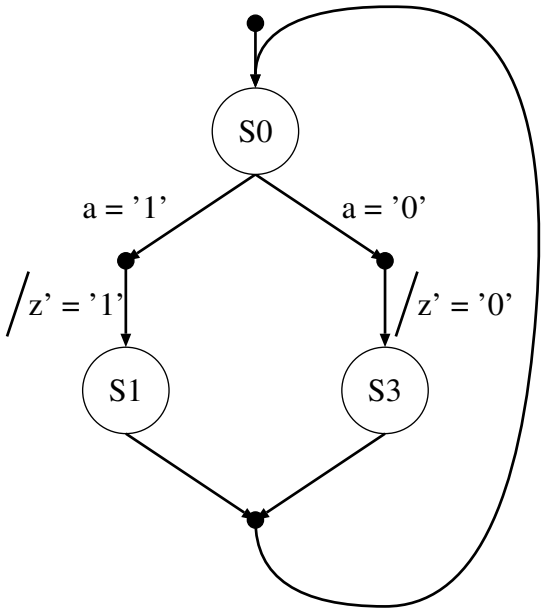


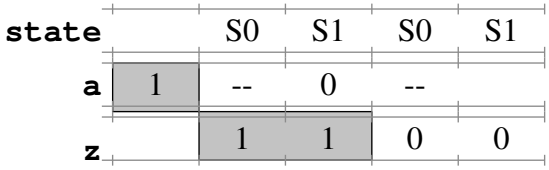
Same.

4.2.0.5

```
process begin
  wait until rising_edge(clk);
  if a = '1' then
    z <= '1';
    wait until rising_edge(clk);
  else
    z <= '0';
    wait until rising_edge(clk);
  end if;
end process;
```

Answer:



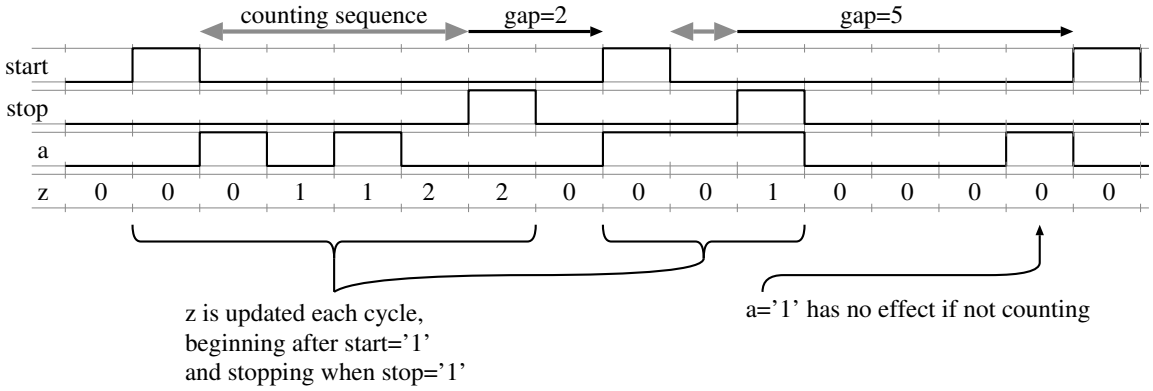


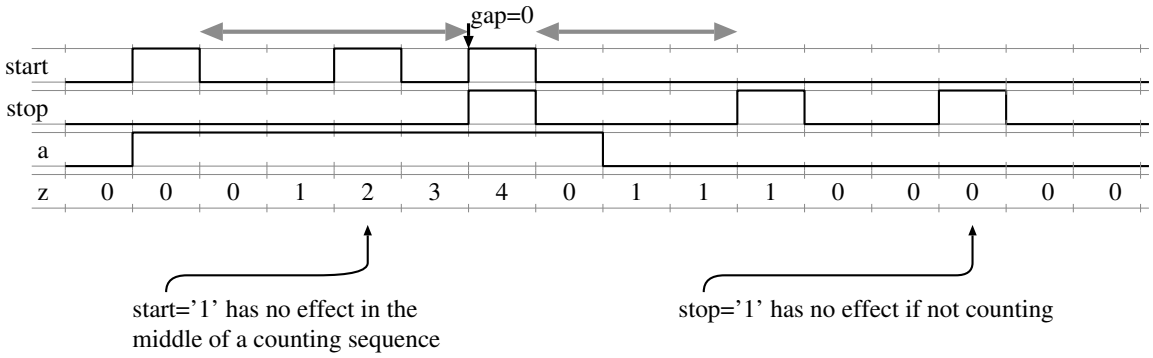
Same.

4.3 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.





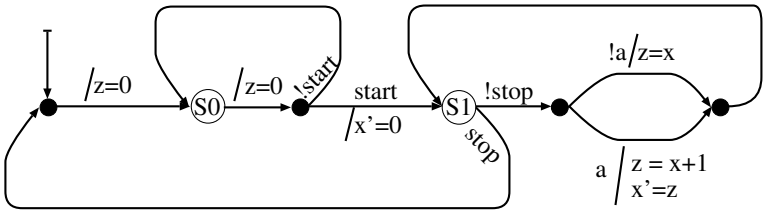
NOTES:

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.

4.3.1

Answer:
Incorrect.

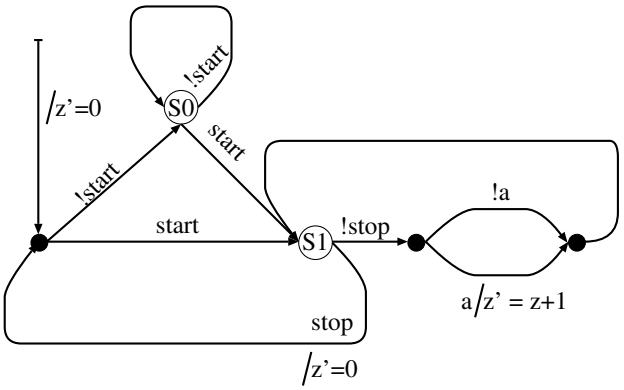
Explanation: When `a='1'`*, the output* `z` *is updated in the* current *clock cycle. It should be updated in the next clock cycle.*



4.3.2

Answer:
Correct. Gap=0.

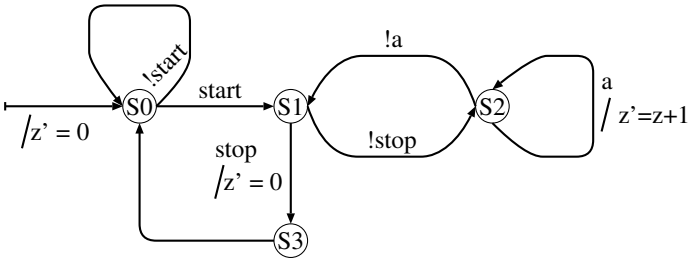
Explanation: `z` *is initialized in the first clock cycle and when* `stop='1'`*. When* `stop='1'`, `start` *is sampled in the same clock cycle.*



4.3.3

Answer:
Incorrect.

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

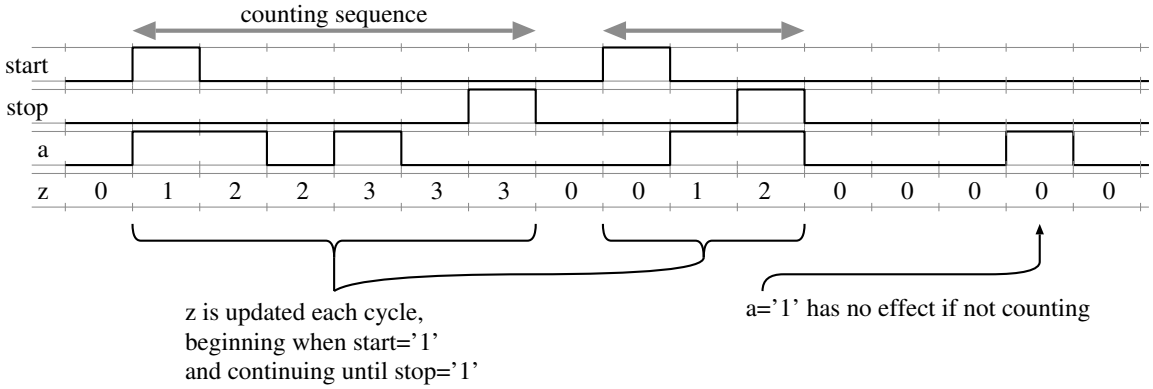


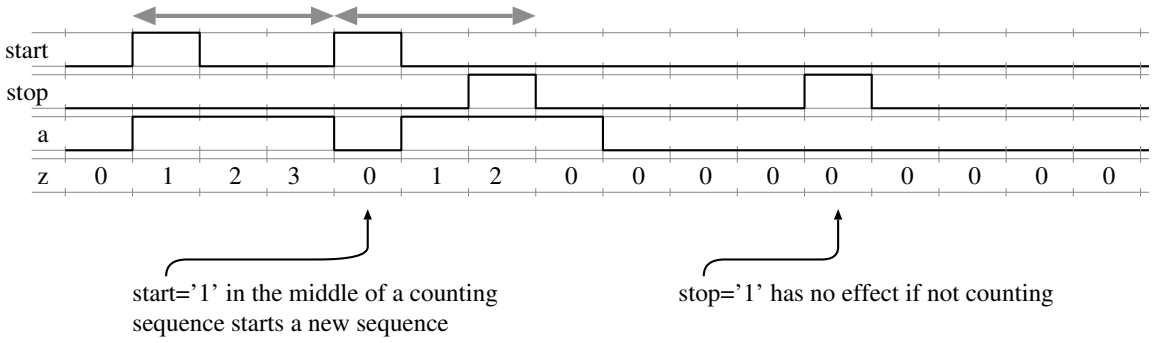
4.4 Count-Sequence Design

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

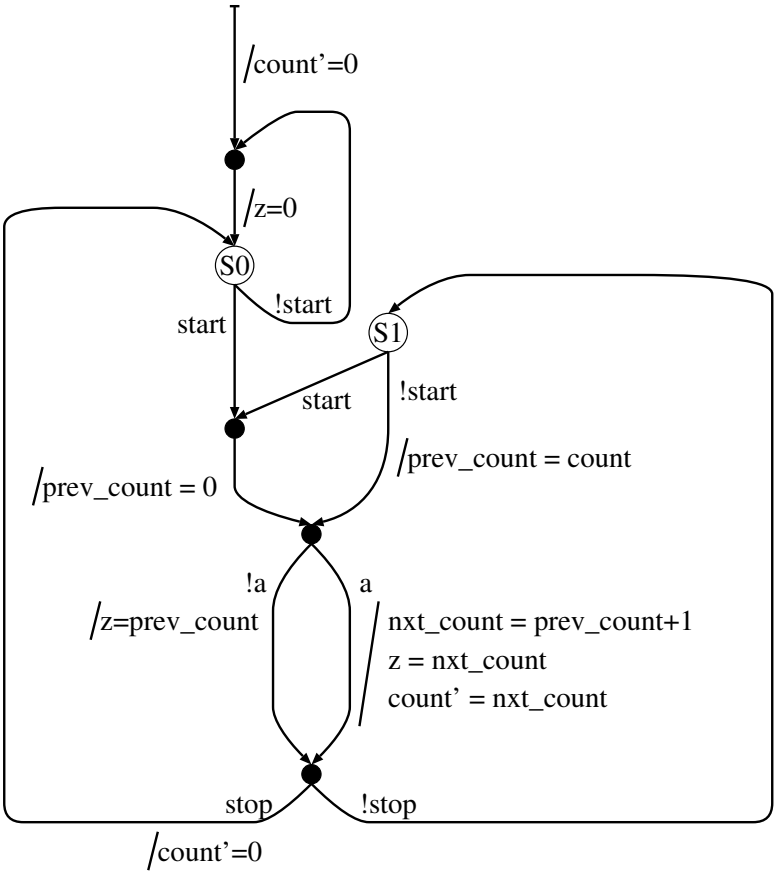
NOTES:

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.





Answer:

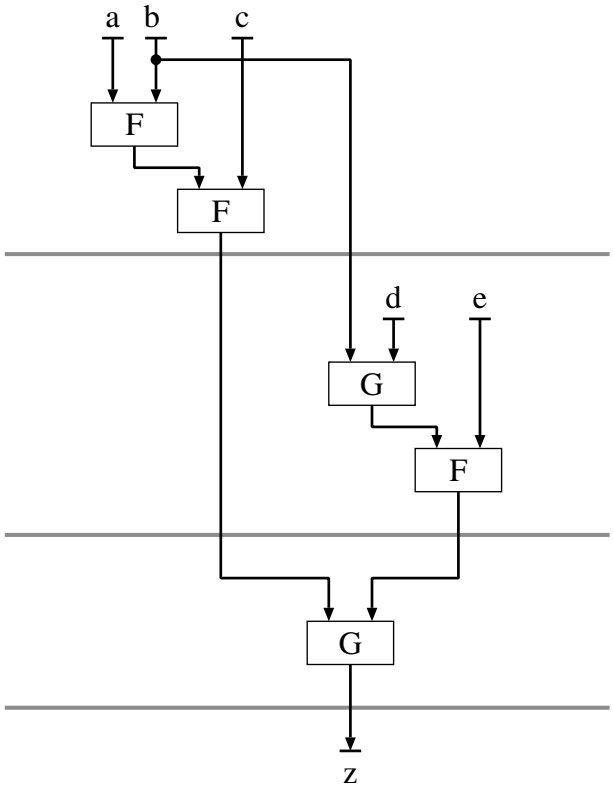

/count'=0
/z=0
S0
!start
start
S1
start
!start
/prev_count = count
/prev_count = 0
!a
a
/z=prev_count
nxt_count = prev_count+1
z = nxt_count
count' = nxt_count
stop
!stop
/count'=0

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.

Answer:

input ports	*3*
output ports	*1*
registers	*2*
f components	*2*
g components	*1*

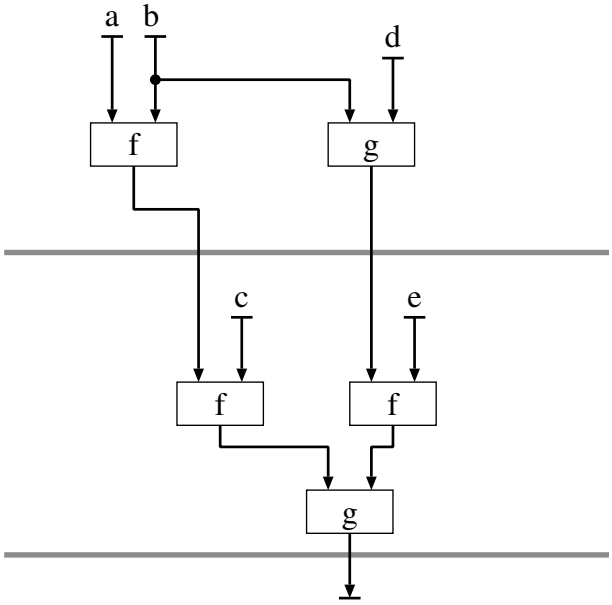
5.1.2 Optimization

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

Answer:



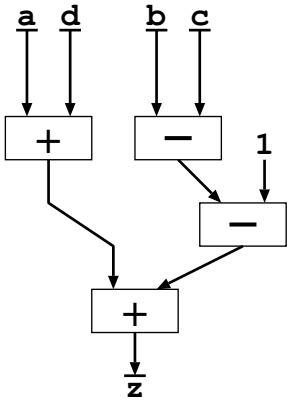
5.2 Michener: Design and Optimization

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

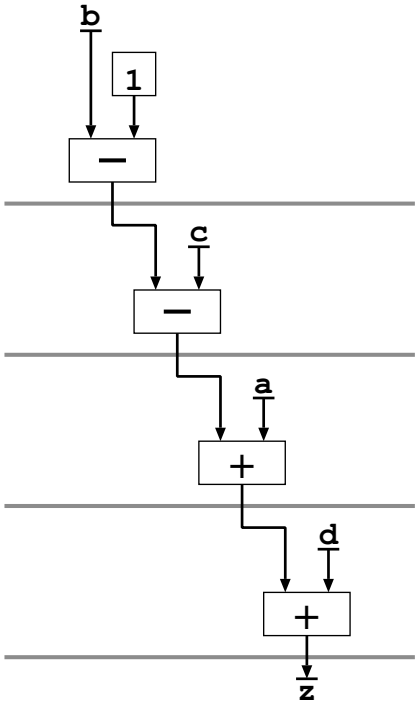
NOTES:

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.

Answer:



Data-dependency graph



Dataflow diagram

5.3 Allocation and Control Table

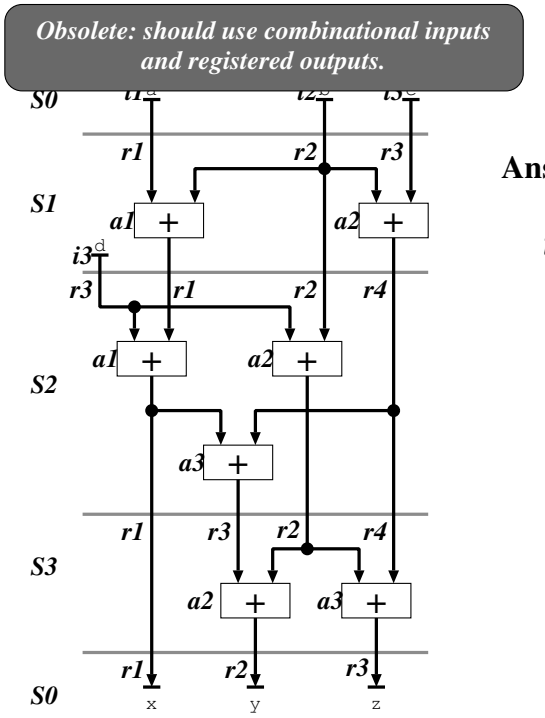
5.3.1 Allocation

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.

Answer:



Answer:

There are several allocations that arrive at the following result:

Number of input ports	*3*
Number of output ports	*3*
Number of adders	*3*
Number of registers	*4*
Number of 2:1 multiplexers	*5*

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.

Answer:

	r1		**r2**		**r3**		**r4**		**a1**		**a2**		**a3**	
	ce	*d*	*ce*	*d*	*ce*	*d*	*ce*	*d*	*src1*	*src2*	*src1*	*src2*	*src1*	*src2*
S0	*1*	*i1*	*1*	*i2*	*1*	*i3*	*–*	*–*	*–*	*–*	*–*	*–*	*–*	*–*
S1	*1*	*a1*	*0*	*–*	*1*	*i3*	*1*	*a2*	*r1*	*r2*	*r2*	*r3*	*–*	*–*
S2	*1*	*a1*	*1*	*a2*	*1*	*a3*	*0*	*–*	*r1*	*r3*	*r2*	*r3*	*a1*	*r4*
S3	*0*	*–*	*1*	*a2*	*1*	*a3*	*–*	*–*	*–*	*–*	*r2*	*r3*	*r2*	*r4*

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);
				...
				if r99_d_sel = '1' then
				  r99 <= r80;
				else
				  r99 <= r81;
				end if;

	r99 <= r81 when	:	others	

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";
```

Answer:

Original 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*	*–*	*–*

Use commutivity to swap operands to add so as to minimize number of different operands for each source.

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

Choose don't care values to have all *entries in a column are the same, or* all but one *of the entries in a column are the same.*

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

For sub1 when doing the don't care instantiation, could choose to instantiate the don't care values all with either "add1" (as was done) or with "r2". Both choices end up with one value appearing in only one cell and one other value

appearing in all of the other cells of the column. It is important to recognize here that we cannot swap operands to subtract.

Implementation expressions:

```
r1_ce <= '1';
r1_d  <=   i1 when st(0) = '1'
      else r3;
r2_ce <= not( st(0) );
r2_d  <= add1;
r3_ce <= not( st(1) );
r3_d  <=   r1 when st(3)
      else i2;
add1_src1 <= r2 when st(0) = '1'
          else r1;
add1_src2 <= r3;
sub1_src1 <= r2 when st(1) = '1'
          else add1;
sub1_src2 <= r2 when st(2) = '1'
          else add1;
```

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";
```

Answer:

Use the same commutivity swapping of operands to minimize the number of different operands for each source.

Pick values for don't cares that minimize the amount of combinational circuitry. This can be done by quickly looking for patterns (e.g. same value for S0 and S2 and same value for S1 and S3 gives hardware that looks just at st(0)).

In the solution though, we illustrate a detailed process. We are looking for 1: simple Karnaugh maps, and 2: identical or similar equations for multiple signals, so that we can reuse hardware.

Draw Karnaugh maps with the following layout:

S0	*S1*
S2	*S3*

From the state encoding, we arrive at the following layout of the Karnaugh map:

00	*01*
10	*11*

Draw Karnaugh maps for each signal that has multiple values.

r1.d		***r2.ce***		***r3.d***		***r3.ce***		***add.src2***		***sub.src1***		***sub.src2***	
i1	*r3*	*0*	*1*	*i2*	*–*	*1*	*0*	*r2*	*r1*	*–*	*r2*	*–*	*add1*
r3	*–*	*–*	*1*	*i2*	*r1*	*1*	*1*	*r1*	*r1*	*add*	*–*	*r2*	*–*

Pick values for the don't care positions.

r1.d		***r2.ce***		***r3.d***		***r3.ce***		***add.src2***		***sub.src1***		***sub.src2***	
i1	*r3*	*0*	*1*	*i2*	***r1***	*1*	*0*	*r2*	*r1*	***add***	*r2*	*r2*	*add1*
r3	***r3***	***0***	*1*	*i2*	*r1*	*1*	*1*	*r1*	*r1*	*add*	***r2***	*r2*	*add1*

Use the K-maps to fill in the control table.

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

Signal	Case		Expression
r1_d_sel	r1 <= i1 when	:	st(0) or st(1)
	r1 <= r3 when	:	others
r1_ce		:	N/A
r2_d_sel	r2 <= add1 when	:	N/A
r2_ce		:	st(0)
r3_d_sel	r3 <= i2 when	:	others
	r3 <= r1 when	:	st(0)
r3_ce		:	not(st(0)) or st(1)
add1_src1_sel	add1_src1 <= r1 when	:	N/A
	add1_src1 <= r3 when	:	N/A
add1_src2_sel	add1_src2 <= r1 when	:	st(1) or st(0)
	add1_src2 <= r2 when	:	others
	add1_src2 <= r3 when	:	N/A
sub1_src1_sel	sub1_src1 <= r2 when	:	st(0)
	sub1_src1 <= add1 when	:	others
sub1_src2_sel	sub1_src2 <= r2 when	:	others
	sub1_src2 <= add1 when	:	st(0)

5.5 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

5.6 Inter-Parcel Variables

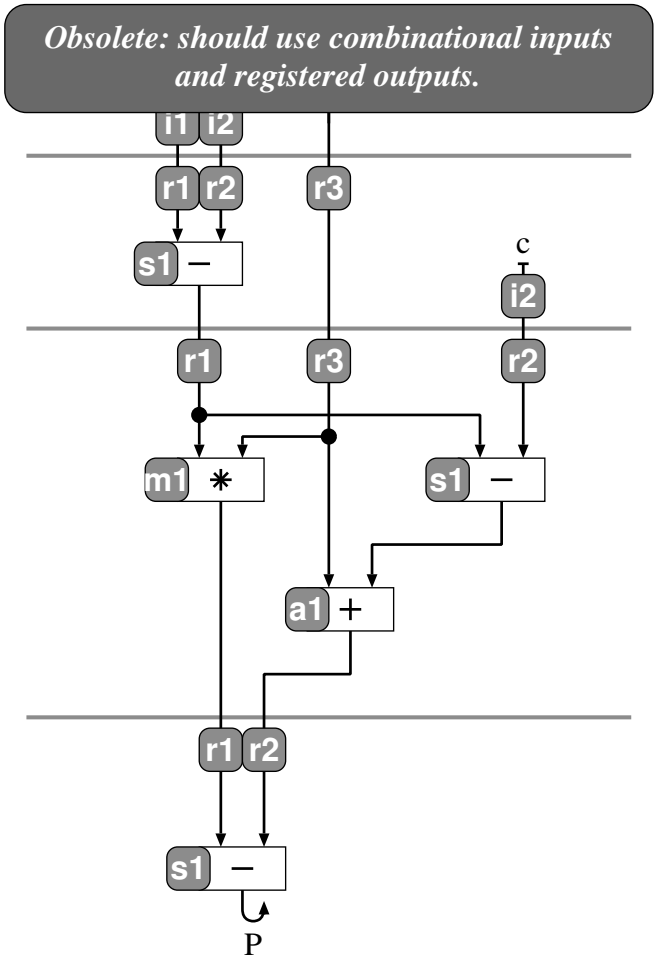
This question explores various aspects of the system described by the specification and dataflow diagram below.

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

5.6.1 Register Allocation

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

Answer:



5.6.2 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)*

5.6.3 Control Table

Draw the control table.

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

Answer:

		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	—	—

5.6.4 State Encoding

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

Answer:
Valid-bits. Having an unpredictable number of bubbles and having an inter-parcel variable (P) are each, by themselves, sufficient conditions to make a valid-bit encoding the best choice.

Q How many states does the system have?

Answer:
5: one for each of the 4 clock cycles of latency and 1 for the idle state.

5.7 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

Answer:

Bugs:

1. ***B:*** `gt_count` *should be incremented only when* `i_valid=1`
2. ***B:*** `o_count`, `gt_count`, *and* `data_count` *should all be 9 bits wide, so that they can count all the way up to 256. With 8 bits, we can count only up to 255.*
3. ***B:*** `gt_count` *keeps counting after* `o_done` *is true.*

Area improvements:

1. ***A:*** `data_count <= data_count` *will probably synthesize to mux. The* `else` *clause should be deleted. The code will then synthesize to a register with a chip-enable.*
2. ***A:*** *The condition* `i_data>63` *should be optimized to check just the two most significant bits. Similarly,* `data_count>=256` *should check just the MSB.*
3. ***A:*** `o_done` *could be done combinationally*

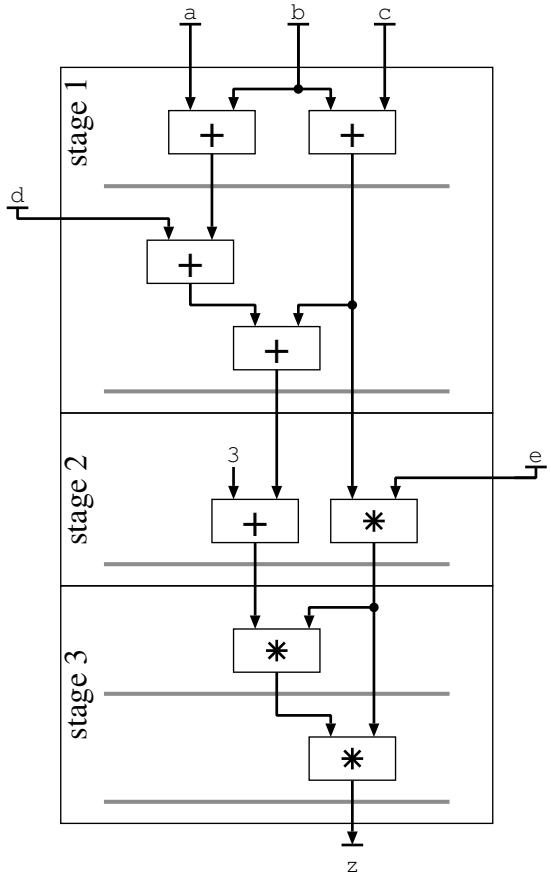
Chapter 6

Advanced Design

6.1 Pipelining and Dataflow Diagrams

6.1.1 Resource Usage

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



Answer:

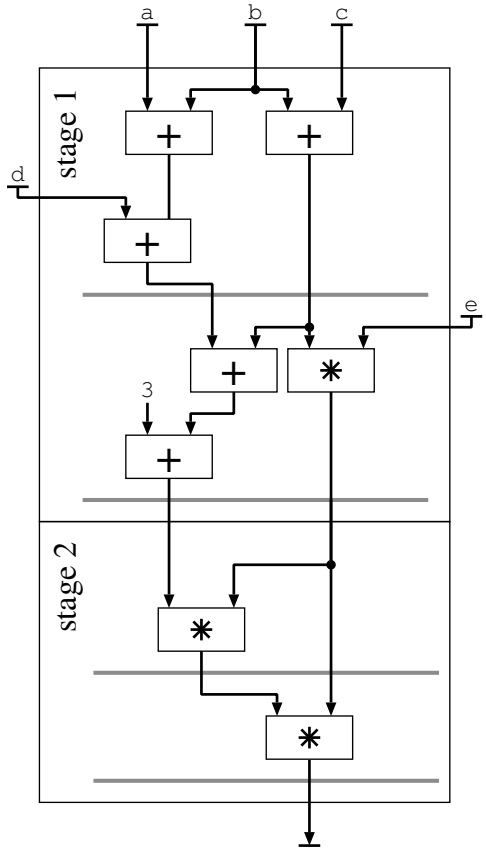
Latency	*2 + 1 + 2 = 5*
Throughput	*1/2*
Clock period	*flop + max(2 add, 1 mul)*
Number of input ports	*3 + 1 + 0 = 4*
Number of output ports	*0 + 0 + 1 = 1*
Number of adders	*2 + 1 + 0 = 3*
Number of multipliers	*0 + 1 + 1 = 2*
Number of registers	*2 + 2 + 2 = 6*

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

Optimization question:

Answer:



Latency	*2 + 2 = 4*
Throughput	*1/2*
Clock period	*flop + max(2 add, 1 mul)*
Number of input ports	*4 + 0 = 4*
Number of output ports	*0 + 1 = 1*
Number of adders	*3 + 0 = 3*
Number of multipliers	*1 + 1 = 2*
Number of registers	*2 + 2 = 4*

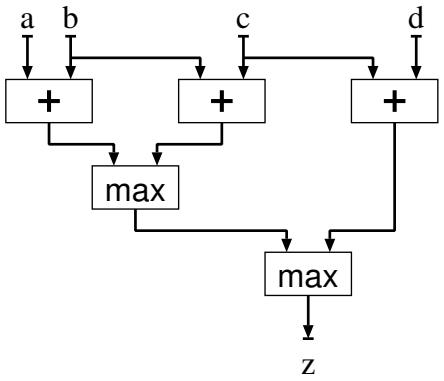
6.2 Overlapping Pipeline Stages

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

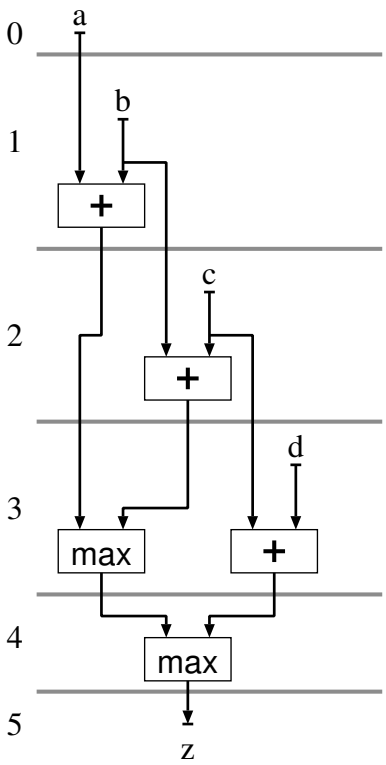
Goals:
1. Minimum area
2. Minimum latency

1. Data-dependency graph



2. First dataflow diagram

   To achieve a design with just one input port and minimum area, we will aim for a tall and narrow dataflow diagram.



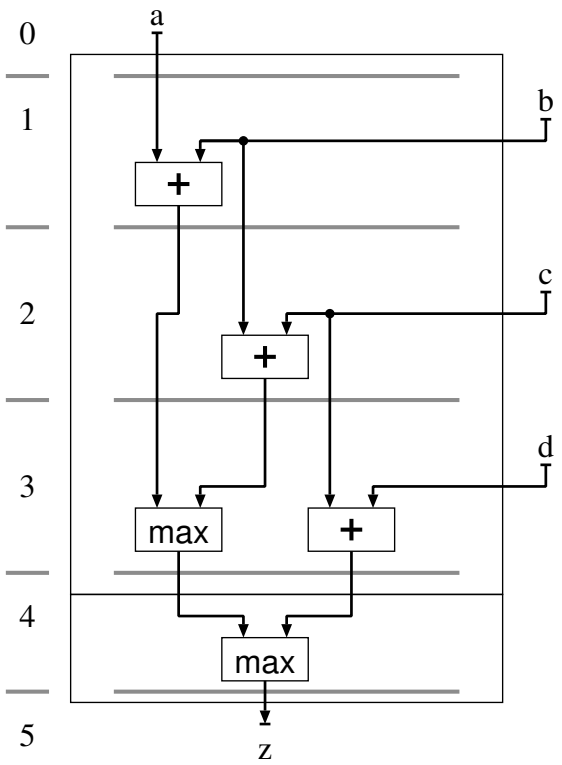
Our first design violates the requirement that the throughput be at least 1/4.

With a requirement of at most one input and path of four operations from inputs to output (three adds and one max) we have a minimum latency of five. Thus, we must pipeline our design.

As a small technical note, to satisfy the requirement of a maximum of 1 input port, we must store `a` in a register before we can do the addition. Because of this, we will say that the DFD has combinational inputs, even though `a` goes directly to a register.

3. First pipeline

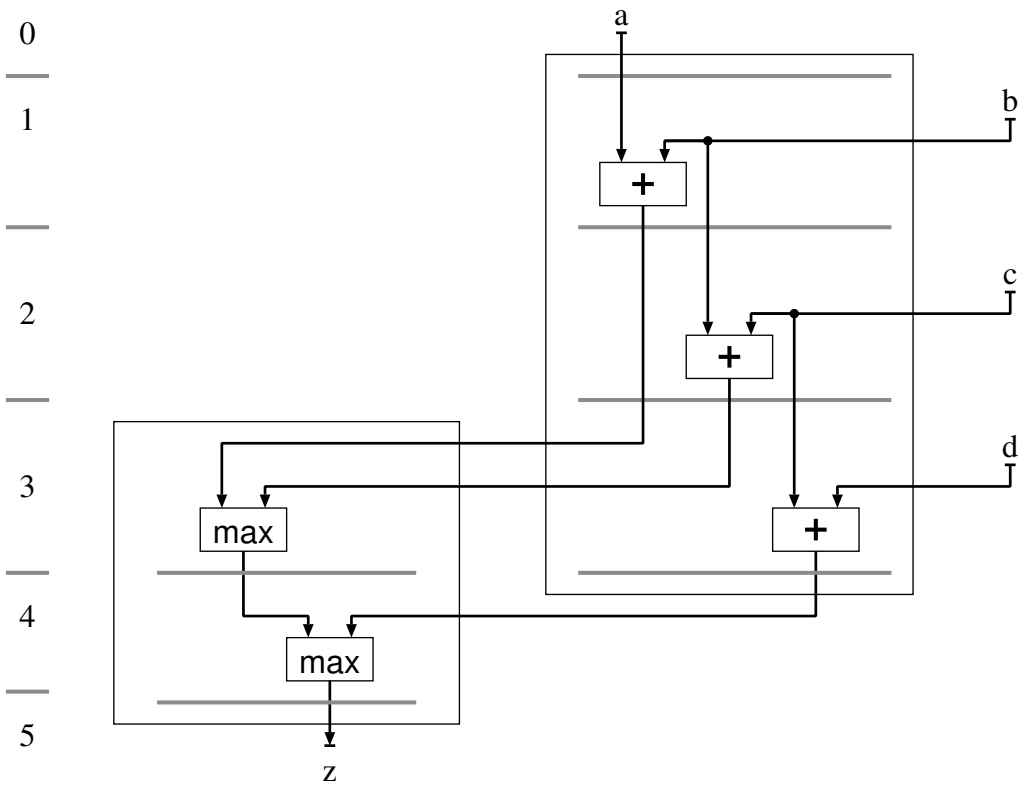
   Our first pipeline design is to simply add a stage boundary after clock-cycle four so as to allow a throughput of 1/4.



The disadvantage of this design is that we have two max components. If we keep the stage boundary after clock cycle four, we can move the first max to be in the second stage by postponing it to clock-cycle five. However, this will increase the latency to six.

4. Overlapping stages

   We must have a maximum latency through the first stage of four clock cycles.



5. Analysis

	Stage1	Stage2	Overall
Inputs	1	0	1
Registers	3	1	4
Add	1	0	1
Max	0	1	1
Latency			5
Throughput	1/4	1/2	1/4

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?*

Answer:

$$\begin{aligned}
\mathit{TimeTot} &= \mathit{NumTrips} \times (\mathit{TimeLoaded} + \mathit{TimeUnloaded}) \\
\\
\mathit{NumTrips} &= \lceil \mathit{Harvest}/\mathit{Capacity} \rceil
\end{aligned}$$

All trips are for the same distance, so distance cancels out of the equations:

$$\mathit{Time} \propto 1/\mathit{Speed}$$

$$\begin{aligned}
\mathit{TimeTotBig} &\propto \lceil 85/12 \rceil \times (1/15 + 1/38) \\
&\propto 8 \times 0.0930 \\
&\propto 0.7439 \\
\mathit{TimeTotSmall} &\propto \lceil 85/6 \rceil \times (1/30 + 1/70) \\
&\propto 15 \times 0.0477 \\
&\propto 0.7143
\end{aligned}$$

Small truck will take less time

$$\begin{aligned}
\mathit{PctFaster} &= \frac{\mathit{TimeSlow} - \mathit{TimeFast}}{\mathit{TimeFast}} \\
&= \frac{\mathit{TimeTotBig} - \mathit{TimeTotSmall}}{\mathit{TimeTotSmall}} \\
&= \frac{0.7439 - 0.7143}{0.7143} \\
&= 4.15\%
\end{aligned}$$

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.*

Answer:

Use two drivers

Use a combination of the small truck and large truck to improve his utilization.

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?

Answer:

Data throughput can be thought of as useful_data / time. So, often in these types of questions you will have to do the following:

*total_data/time * useful_data/total_data.*

The maximum data throughput of the two technologies in terms of bits can be calculated as follows:

1. *BigLan Network Protocol*

   *Maximum data throughput = 160 Mbps * (8000 useful data bits per packet / 8800 total d*
   = 145.45 Mbps

2. *DataChopper Router*

   Time required for a packet = 500 clock cycles
   *+ 0.5 cycles per bit * 8800 packet bits*
   = 500 clock cycles + 4400 clock cycles
   = 4900 clock cycles
   *= 4900 clock cycles * 13.33 ns per cycle*
   = 65333 ns per packet

   Time required for a data bit = 65333 ns per packet / 8000 data bits
   = 8.167 ns per data bit

   Maximum data throughput = 1 / 8.167 ns per data bit
   = 122.46 Mbps

*You could also use the previous method: = cycles/sec * total_bytes/cycle * useful_bytes/t*

The network has a higher maximum throughput.

What percentage higher?

n% higher performance = (perf_high - perf_low) / perf_low
= (145 - 122)/122
= 19%

The network has 19% higher maximum performance. Therefore, the router can't keep up with the network.

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.

Answer:

As packet size increases, the overhead associated with the constant routing delay will become less significant.

The data rate of the router will slowly approach that of the network but it will never surpass the network throughput. If there was not any overhead for routing, the peak data rate for the router would be 150 Mbps compared to 160 Mbps of the network.

It shoud be noted that even though a giant packet size would seem like an ideal solution in this question, in reality lost packets, latency, and small data sizes would make this impractical. For example, if each packet was 1 GB and the network was transmitting a cell-phone conversation, you would have to wait a very long time for the first packet to arrive before you could hear the other person. Also, if a packet was lost, you'd have to wait a long time to see if the other person is still on the phone.

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.

Answer:

$$\begin{aligned} P &= 2^{t/24} (\textit{where } t \textit{ is measured in months}) \\ &= 2^{1/24} \\ &= 1.029 \end{aligned}$$

Therefore, performance goes up by 2.9% each month.

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?*

Answer:

Use the following subscripts: `Yme` *1*
`Y!v1` *2*
`Y!u2` *3*

The `Yme` *is 10% faster than the* `Y!v1`*.*

$$\begin{aligned} NumInst_2 &= NumInst_1 \\ ClockSpeed_1 &= 200MHz \\ ClockSpeed_2 &= 150MHz \\ CPI_1 &= 4 \end{aligned}$$

Solve for CPI_2.

$$\begin{aligned} Time &= \frac{NumInst \times CPI}{ClockSpeed} \\ \frac{Time_2 - Time_1}{Time_1} &= 0.10 \\ \frac{Time_2}{Time_1} &= 1.10 \\ Time_2 &= 1.10 \times Time_1 \\ \frac{NumInst_2 \times CPI_2}{ClockSpeed_2} &= 1.10 \times \frac{NumInst_1 \times CPI_1}{ClockSpeed_1} \\ CPI_2 &= 1.10 \times \frac{ClockSpeed_2 \times NumInst_1 \times CPI_1}{NumInst_2 \times ClockSpeed_1} \\ &= 1.10 \times \frac{ClockSpeed_2 \times CPI_1}{ClockSpeed_1} \\ &= 1.10 \times \frac{150MHz \times 4}{200MHz} \\ &= 3.3 \end{aligned}$$

Common mistakes:

- *Swapping performance of* `Yme` *and* `Y!v1`*.*

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?*

Answer:

$$
\begin{aligned}
1.3 \times \mathit{Time}_3 &= \mathit{Time}_2 \\
1.3 \times \frac{\mathit{NumInst}_3 \times \mathit{CPI}_3}{\mathit{ClockSpeed}_3} &= \frac{\mathit{NumInst}_2 \times \mathit{CPI}_2}{\mathit{ClockSpeed}_2}
\end{aligned}
$$

Solve for CPI_3:

$$
\begin{aligned}
\mathit{CPI}_3 &= \frac{\mathit{ClockSpeed}_3 \times \mathit{NumInst}_2 \times \mathit{CPI}_2}{1.3 \times \mathit{NumInst}_3 \times \mathit{ClockSpeed}_2} \\
&= \frac{180\mathit{MHz} \times 3.3}{1.3 \times 0.9 \times 150\mathit{MHz}} \\
&= 3.38
\end{aligned}
$$

Common mistakes:

- *Comparing performance of* `Y!u2` *to* `Yme`*, rather than* `Y!v1`*.*
- *Saying that time for* `Y!u2` *is 70% of* `Y!v1`*.*
- *Forgeting to take into account reduced number of instructions.*

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

Answer:

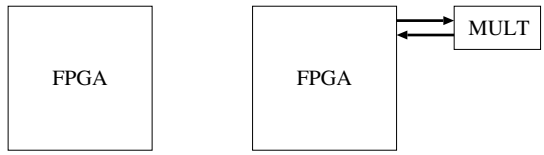
The most likely analysis is that the `Y!u2` *is basically the same as the* `Y!v1` *except for the multiply. This is because the* `Y!u2` *has a slightly larger CPI than the* `Y!v1`*, this is in keeping with the addition of a multiply instruction. A multiply instruction probably has a larger-than-average CPI.*

The increase in clock speed likely comes from a new fabrication process, and would not have required significant changes to the design of the chip.

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?

Answer:

MIPs for FPGA option:

$$\begin{aligned} MIPs_{FPGA} &= \frac{MHz_{FPGA}}{CPI_{FPGA}} \\ &= \frac{200}{5} \\ &= 40 \end{aligned}$$

Find MIPs for FPGA+MULT option:

$$MIPs_{FM} = \frac{MHz_{FM}}{CPI_{FM}}$$

Find CPI for MIPS+FPGA option:

$$CPI_{FM} = PctInst_{mult} \times CPI_{mult} + PctInst_{other} \times CPI_{other}$$

Find CPI for non-multiply (other) instructions. Key insight is that the CPI for non-multiply instructions is the same for both the FPGA and FPGA+MULT.

$$\begin{aligned} CPI_{FPGA} &= PctInst_{mult} \times CPI_{mult} + PctInst_{other} \times CPI_{other} \\ CPI_{other} &= \frac{CPI_{FPGA} - PctInst_{mult} \times CPI_{mult}}{PctInst_{other}} \\ &= \frac{5 - 0.1 \times 20}{0.9} \\ &= 3.333 \end{aligned}$$

$$\begin{aligned} CPI_{FM} &= PctInst_{mult} \times CPI_{mult} + PctInst_{other} \times CPI_{other} \\ &= 0.1 \times 6 + 0.9 \times 3.333 \\ &= 3.6 \end{aligned}$$

$$\begin{aligned} MIPs_{FM} &= \frac{MHz_{FM}}{CPI_{FM}} \\ MIPs_{FM} &= \frac{160}{3.6} \\ &= 44.4 \end{aligned}$$

$MIPs_{FM} > MIPs_{FPGA}$, therefore the FPGA+MULT is the higher performance option.

$$\begin{aligned} N\%Faster &= \frac{Perf_{FM} - Perf_{FPGA}}{Perf_{FPGA}} \\ &= \frac{44.4 - 40}{40} \\ &= 11.1\% \end{aligned}$$

The FPGA+MULT option is 11% faster than the FPGA option.

7.5.2 Performance Metrics

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

Answer:

- *MIPs is a good metric for this example, because we are comparing two microprocessors that use the same instruction set and will be used in the same environment.*

  In general, the disadvantage of MIPs is that it doesn't take into account that different instructions accomplish different amounts of work. This causes problems when comparing microprocessors that use different instruction sets (e.g. one with a cosine instruction and one without).

  On an exam, you need to explain whether or not MIPS is a good choice to use based on what is stated in the question. For example, if a question states that two processors have the same instruction set, are running the same program, give some information relating CPI and clock speed, then you can state that MIPS would be an okay comparison for the stated reasons.

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?

Answer:

Calculate time to produce each pixel in hardware and software, set times equal, solve for clock frequency of FPGA:

$$
\begin{aligned}
T_{hw} &= \frac{1cyc}{1pix} \times \frac{1}{F_{hw}} \\
\\
T_{sw} &= \frac{NumInst \times CPI_{avg}}{F_{sw}} \\
&\quad (observe: \quad \%_i = Num_i / NumInst) \\
&= \frac{\sum_{i=1}^{n} Num_i \times CPI_i}{F_{sw}} \\
&= \frac{Num_{Ld} \times CPI_{Ld} + Num_{St} \times CPI_{St} + Num_{Br} \times CPI_{Br} + Num_{Ad} \times CPI_{Ad} + Num_{Mu} \times CPI_{M}}{F_{sw}} \\
&= \frac{(7 \times 1.4) + (2 \times 1.1) + (1 \times 1.2) + (8 \times 1.0) + (4 \times 2.2) cycs/pix}{900MHz} \\
&= \frac{30cycs/pix}{900MHz} \quad (CPI_{avg} = 1.3636)
\end{aligned}
$$

$$
\begin{aligned}
T_{sw} &= T_{hw} \\
(30cycs/pix)/900\ MHz &= \frac{1}{F_{hw}} \\
F_{hw} &= \frac{1}{T_{hw} \times 2} \\
&= 900/30 \\
&= 30MHz
\end{aligned}
$$

The FPGA must run at 30MHz to have the same performance as the software-based solution.

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:

$CP_{avg}(t)$ = cost/performance ratio of the average Waterluvian filter on the market at time t.

CP_{hw} = cost/performance ratio of hardware option

CP_{sw} = cost/performance ratio of software option

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

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

Answer:

For the software option, the development cost per unit is:

$$= \frac{3months \times \$100,000/month}{20,000units}$$
$$= \$15$$

For the hardware option, the development cost per unit is:

$$= \frac{6months \times \$100,000/month}{20,000units}$$
$$= \$30$$

In the time between the software option would reach the market and when the hardware option would reach the market the average performance of Waterluvian filters on the market increases by a factor of:

$$= 2^{(6months-3months)/24months}$$
$$= 2^{(1/8)}$$
$$= 1.09$$

The cost of Waterluvian filters remains constant, but the performance increases over time, therefore the average cost/performance ratio ***decreases****.*

$$CP_{avg}(6months) = \frac{1}{1.09} \times CP_{avg}(3months)$$

Cost/performance ratio of software option:

$$\begin{aligned} CP_{sw} &= \frac{C_{sw}}{P_{sw}} \\ &= \frac{\$100 + \$15}{P_{sw}} \end{aligned}$$

For the relative cost/performance ratio of hardware option at time it reaches the market to be the same as the software option when it reached the market, need:

$$CP_{hw} = \frac{CP_{sw}}{1.09}$$

Both hardware and software have the same performance, so relative cost-performance ratios depend solely upon cost:

$$\begin{aligned} C_{hw} &= \frac{C_{sw}}{1.09} \\ C_{fpga} + \$30 &= \frac{\$115}{1.09} \\ C_{fpga} &= \frac{\$115}{1.09} - \$30 \\ &= \$75 \end{aligned}$$

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.

Answer:

Mathematically, the answer would be different.

The relative cost-performance ratio remains the same: 1.09, because the time difference remains at 3 months.

The development cost of each option increases by a constant amount ($10 for the additional 2 months of development).

*Because the development costs increase by a **constant** amount, but the **ratio** of total costs must remain the same, the cost of the FPGA will be different than the previous calculation.*

7.7 Waterluvian on New FPGA Chip

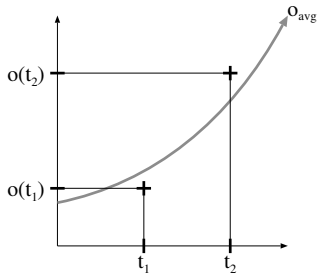
You are developing a new design of the Waterluvian filter that will be implemented on an FPGA chip (the definition of a Waterluvian filter is irrelevant to this question). Your project leader just returned from a conference where she learned about a new FPGA chip. She has asked you to analyze the tradeoffs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

Answer:



$$o_{avg}(t_1) = \frac{1.25}{1.05} \times o(t_1) \tag{7.1}$$

$$o_{avg}(t_2) = o_{avg}(t_1) \times 2^{\frac{t_2 - t_1}{24}} \tag{7.2}$$

$$o(t_2) = \frac{1.2}{0.85} \times o(t_1) \tag{7.3}$$

$$\frac{o(t_2)}{o_{avg}(t_2)} = 1.07 \tag{7.4}$$

Solve for $t_2 - t_1$.

To get $t_2 - t_1$, we will need to use (7.2), which means that we first need to solve for $\frac{o_{avg}(t_2)}{o_{avg}(t_1)}$.

Because we are solving for a ratio, setup chain of ratios:

$$\begin{array}{rcccccc} & & (7.4) & & (7.3) & & (7.1) \\ \frac{o_{avg}(t_2)}{o_{avg}(t_1)} & = & \frac{o_{avg}(t_2)}{o(t_2)} & \times & \frac{o(t_2)}{o(t_1)} & \times & \frac{o(t_1)}{o_{avg}(t_1)} \\ & = & \frac{1}{1.07} & \times & \frac{1.2}{0.85} & \times & \frac{1.05}{1.25} \\ & = & 1.1080 & & & & \end{array}$$

Substitute into (7.2):

$$\begin{aligned} \frac{o_{avg}(t_2)}{o_{avg}(t_1)} &= 2^{\frac{t_2-t_1}{24}} \\ 1.1080 &= 2^{\frac{t_2-t_1}{24}} \\ \log_2 1.1080 &= \frac{t_2-t_1}{24} \\ 24 \times \log_2 1.1080 &= t_2 - t_1 \\ 24 \times \frac{\log_{10} 1.1080}{\log_{10} 2} &= t_2 - t_1 \\ t_2 - t_1 &= 3.55 months \end{aligned}$$

If we switch to the new FPGA chip, we can delay the project by 3.55 months and still achieve an optimality that is 7% more than average.

Chapter 8

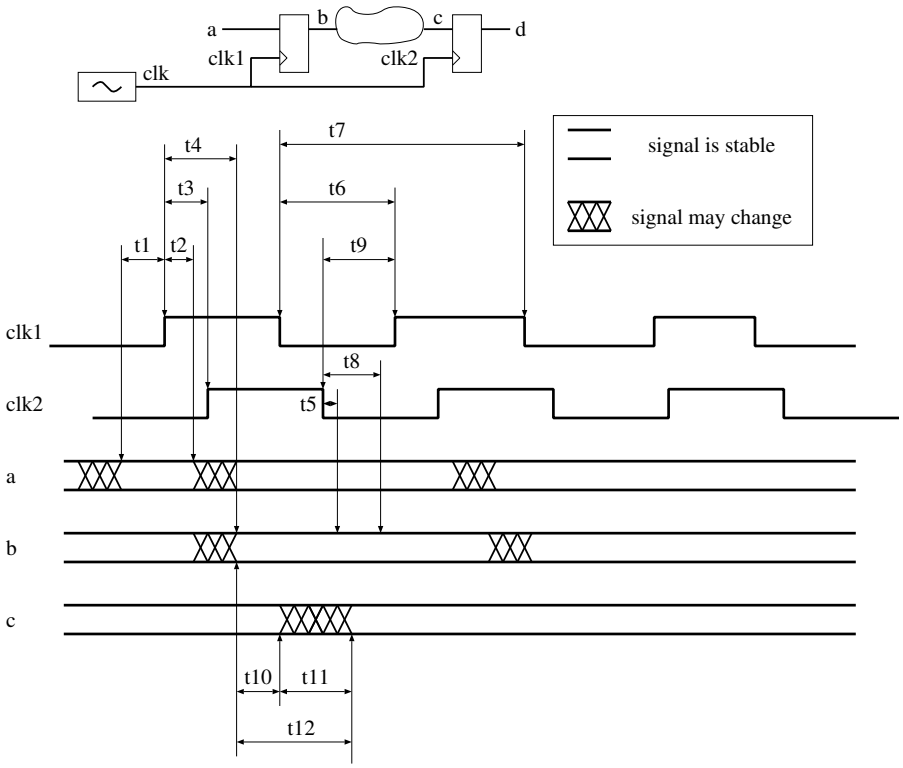
Timing Analysis

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.



Answer:

clock skew	t_3
clock period	t_7
setup time	t_1
hold time	t_2
clock to Q	t_4

8.2 Hold Time Violations

8.2.1 Cause

What is the cause of a hold time violation?

Answer:

The cause of a hold time violation is that new data reaches the gate that enables the input to affect the output before the gate is turned off.

8.2.2 Behaviour

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

Answer:

The bad behaviour that results from a hold time violation is that the new data will corrupt the contents of the storage loop, which is trying to store the previous data. If the new data arrives early enough to satisfy the setup constraint, then the new data will overwrite the previous data and will slip through the latch or flop.

8.2.3 Rectification

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

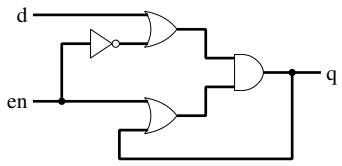
Answer:

A hold time violation can be corrected by adding a delay (buffer) to the data path before the input gate.

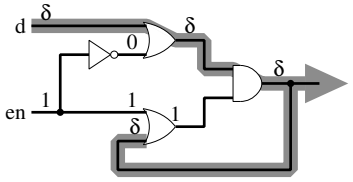
8.3 Latch Analysis

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.

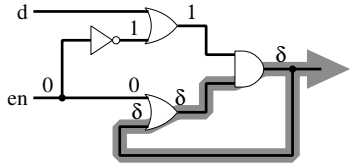
Gate Delays	
AND	4
OR	2
NOT	1



Answer:



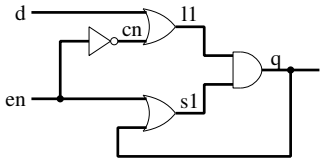
Load mode



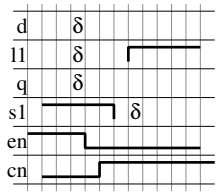
Store mode

*From the mode diagrams, if the circuit is a latch, it is **active high**, because latch is in load mode when* `en='1'`.

Now check if timing of circuit is correct. The critical transition is from load mode to store mode.



Node labels



Timing diagram for transition from load to store mode.

Clock-to-Q: 6 (1 NOT*, 1* AND*, and 1* OR *gate from en to q.)*

Setup: 6 (1 AND *and 1* OR *from d to controlling gate for storage loop, 0 gates from enable to controlling gate for storage loop.)*

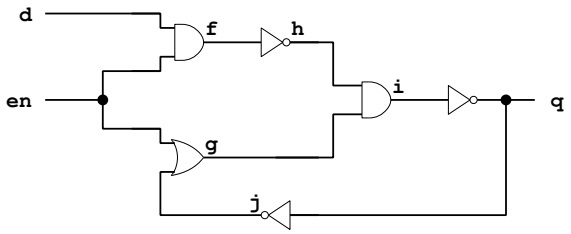
Hold:

- *Hold time constraint must prevent new value arriving at* `d` *before* `en` *sets* `l1` *to* `'1'`.

- *Delay along data path is 0.*
- *Delay along clock path is 1.*
- *Hold time is 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?

Answer:

Yes, the circuit is a correct implementation of a latch.

- *It has a storage loop.*
- *Can turn on and off load path*
- *Can turn on and off storage loop*
- *Even number of inverters on storage loop.*
- *When transition from load to store, the storage loop turns on before the load path turns off (this prevents a glitch from entering the storage loop)*

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.

Answer:

Active-high or active-low?	*Active high*
Clock-to-Q	*4*
Setup	*5*
Hold	*0*

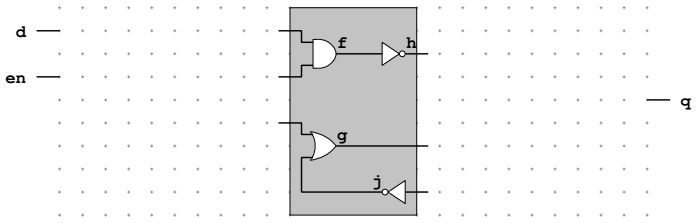
If the circuit is *not* a correct implementation of a latch, explain why.

Answer:
Irrelevant, because circuit is a latch.

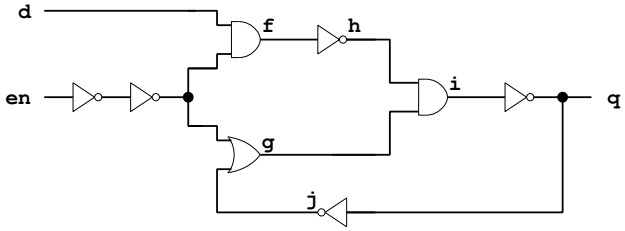
8.4.3 Modification

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.



Answer:
(From before, circuit is a latch)



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.

The new setup time is 3 ns.

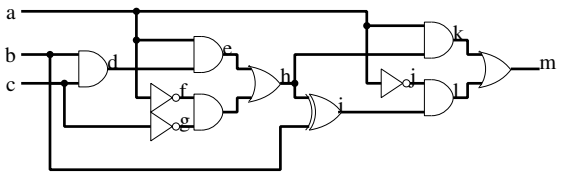
If the circuit is a correct implementation of a latch, for each of the timing parameters below, answer whether your modification to the latch increases, does not change, or decreases the value of the timing parameter.

Answer:

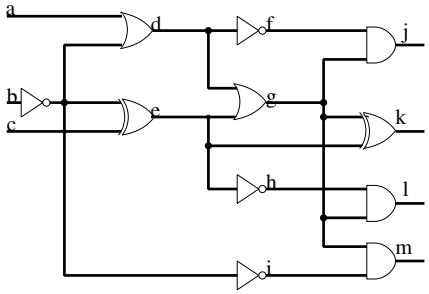
	Increase	*No change*	*Decrease*
Clock-to-Q	$\surd$		
Hold	$\surd$		

8.5 Critical Path and False Path

Find the critical path through the following circuit:



8.6 Critical Path



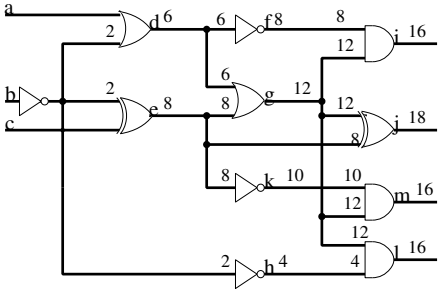
gate	delay
NOT	2
AND	4
OR	4
XOR	6

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.

Answer:



Longest path is: b, e, g, j

8.6.2 Delay

What is the combinational delay along the longest path?

Delay: 18

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?

Answer:

- *false paths*
- *wire delay*
- *clock skew*
- *clock jitter*

8.6.4 Critical Path or False Path?

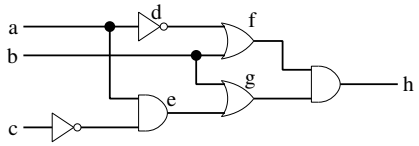
This is an advanced section.
It is <u>not</u> covered in the course
and will <u>not</u> be tested.

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 <u>not</u> covered in the course
and will <u>not</u> be tested.

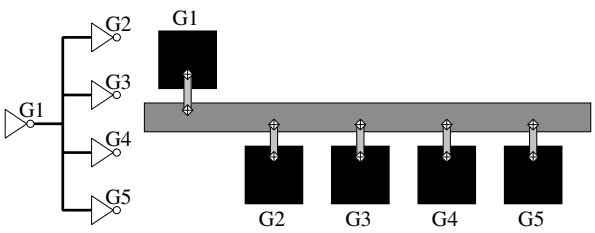
Find the critical path in the circuit below.



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.



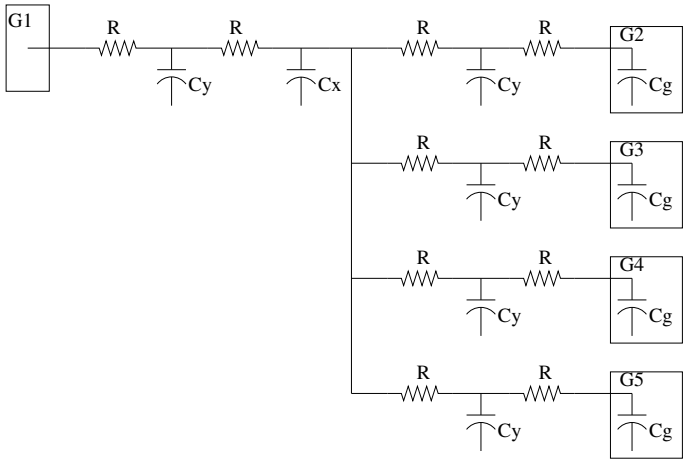
Symbol	Description	Capacitance	Resistance
	Interconnect level 2	Cx	0
	Interconnect level 1	Cy	0
	Gate	Cg	0
	Antifuse	0	R

Assumptions:

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

Answer:

Equivalent Circuit:



$$\begin{aligned} t_{DG2} &= R\times C_y + 2R\times C_x + 3R\times C_y + 4R\times C_g \\ &+ 2R(C_y + C_g) \\ &+ 2R(C_y + C_g) \\ &+ 2R(C_y + C_g) \\ &= 2R\times C_x + 4R\times C_y + 4R\times C_g \\ &+ 6R(C_y + C_g) \end{aligned}$$

In general, for a similar circuit with fanout n:

$$\begin{aligned} t_{DGn} &= 2R\times C_x + 4R\times C_y + 4R\times C_g \\ &+ 2(n-1)R\times(C_y + C_g) \\ &= 2R\times C_x + 2(n+1)R\times(C_y + C_g) \end{aligned}$$

There are two components in the delay equation:

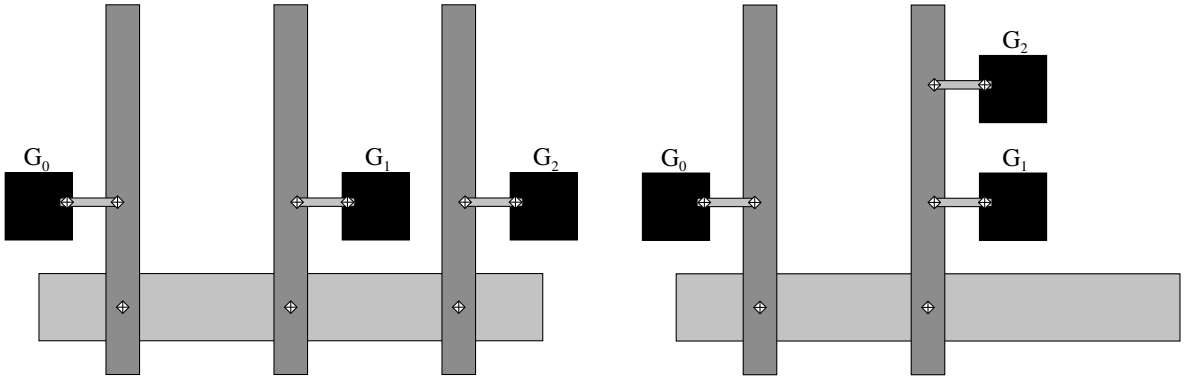
1. A fixed component that is not a function of the fanout ($2R\times C_x$).

2. A component that varies linearly with the fanout ($2(n+1)R\times(C_y + C_g)$).

***Yes**, the fanout model closely matches the timing predicted by the Elmore model.*

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.**

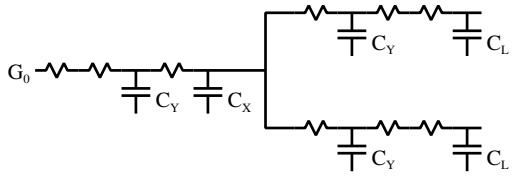
NOTES:

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

Answer:

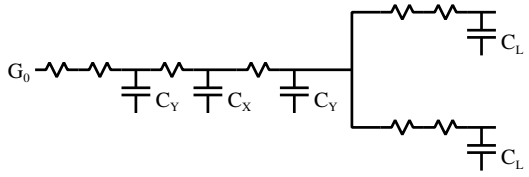
1. *RC network for layout-A*



2. *Delay from G_0 to G_2 for layout-A*

$$\begin{aligned} & 2RC_Y + 3RC_X + 3RC_Y + 3RC_L + 4RC_Y + 6RC_L \\ = \; & 9RC_Y + 3RC_X + 9RC_L \end{aligned}$$

3. *RC network for layout-B*



4. *Delay from G_0 to G_2 for layout-B*

$$\begin{aligned} & 2RC_Y + 3RC_X + 4RC_Y + 4RC_L + 6RC_L \\ = \; & 6RC_Y + 3RC_X + 10RC_L \end{aligned}$$

5. *We cannot determine which layout has a greater delay, A has a delay of $9RC_y$*
6. *Equate the two delays and simplify*

$$\begin{aligned} 9RC_Y + 3RC_X + 9RC_L &= 6RC_Y + 3RC_X + 10RC_L \\ 9C_Y + 9C_L &= 6C_Y + 10C_L \\ 3C_Y &= C_L \end{aligned}$$

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.**

Answer:

There would be no change. As can be seen by the RC-network and delay equations, nodes G_1 and G_2 are symmetric.

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

Answer:

The antifuse closest to the source should be the low-resistance antifuse. This resistance appears in the term for each of the capactitors, hence it has the largest effect on the delay.

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.

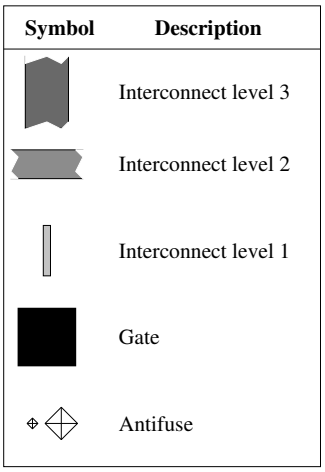
Your first task is to analyze the circuit layouts A and B shown below.

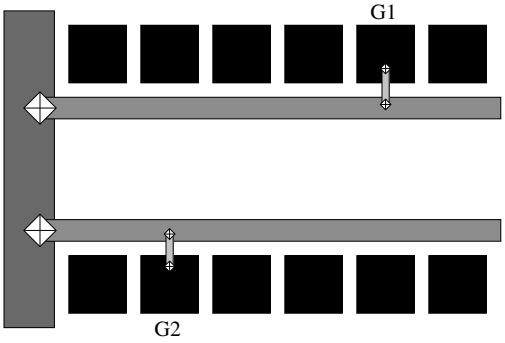
On FPGAs without ZERAF, layout A and B have approximately the same delay. The company wants to know whether the use of ZERAF will cause one of the layouts to have a smaller delay than the other.

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.

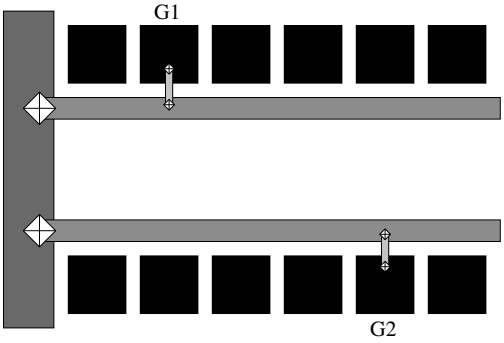
NOTES:

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?

Answer:
40–60%

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?

Answer:
Decreased.

Justification:
- *Transistors have gotten smaller, die size has remained roughly the same size or even increased, clock speeds are increasing.*
- *Signals are travelling roughly the same distance as before, but driving smaller capactive loads. Thus, wire delay is not decreasing much, but capacitive load is decreasing.*
- *The clock period is decreasing, so the wire delay is taking up a larger percentage of the clock period and capacitive load delay is taking up a smaller percentage.*

8.11.3 Temperature and Delay

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

Answer:
Increase.

Justification:
- *As temperature increases, atoms vibrate more, and so have greater probability of colliding with electrons flowing with current.*
- *This increases resistivity, which increases delay.*

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)

Answer:

For worst-case commercial condition, assuming that TA = TJ, Logic Module delay, tPD, for ACT 3 'Std' with 4 fanout is 5.7 ns (see Smith Table 5.2).

Assume this is the slowest path, then estimated critical path delay between registers, tCRIT (worst-case commercial) is:

$$\begin{aligned} tCRIT &= tPD + tSUD + tCO \\ &= 5.7\text{ns} + 0.8\text{ns} + 3.0\text{ns} \\ &= 9.5\text{ns} \end{aligned}$$

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).

Answer:

For worst-case industrial conditions, assuming that TA = TJ, the derating factor is 1.07 (see Table 5.3). Hence the delay tCRIT (worst-case industrial) is: 7% greater than worst case commercial delay: $1.07 \times 9.5 = 10.2\text{ns}$

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).

Answer:

For worst-case industrial conditions, the derating factor at 105C is found by linear interpolation between the values for 85C (1.07) and 125C (1.17).

The interpolated derating factor is 1.12. Hence the delay is: tCRIT (worst-case industrial, TJ = 105 0C) $1.12 \times 9.5 = 10.6$ns.

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?

Answer:

Power will increase.

Justification:

- *Leakage power will increase, because the equation for the leakage power is:*

$$ILeak \propto e^{\frac{-q \times VoltThresh}{k \times T}}$$

  where T is temperature.
- *Short circuiting power will increase because:*
  - *As temperature increases, atoms vibrate more, and so have greater probability of colliding with electrons flowing with current.*
  - *This increases resistivity, which increases delay.*
  - *Signals will rise and fall more slowly, which will increase the short circuiting time, and hence increase short circuiting power*

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?

Answer:

Increase transistor size so as to increase threshold voltage. This will require an increase in supply voltage, which will likely increase total power.

Alternative: *when increase transistor size, keep the supply voltage the same, but decrease performance.*

Alternative: *change fabrication process and materials to reduce leakage current. This will likely be expensive.*

Alternative: *Use dual-Vt fabrication process.*

9.1.3 Clock Gating

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

Answer:

- *If the circuitry has a high utilization rate, then the power consumed by the clock gating circuit could be more than that saved in the main circuit.*

Alternative: *Even if the utilization rate is low, the utilization pattern could prevent the clock gating circuitry from turning off the clock to main circuit. For example, if the circuit receives new data every other clock cycle, it would have a utilization rate of 50%, but might need to be powered up 100% of the time.*

9.1.4 Gray Coding

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

Answer:

- *Gray coding is designed to reduce power, because only one bit changes when incrementing or decrementing.*
- *Program counters usually increment, rather than jump to completely different values. So, using gray coding should reduce power consumption.*
- *The downside is that the memory system probably doesn't use gray-coded addresses, so additional circuitry would be needed to convert between gray and binary codes. This will increase area and likely decrease performance.*
- *Additionally, the extra circuitry to do the translation might require more power than is saved by 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.)**

Answer:

Reducing short circuit time from 1 ns to 0.85 ns means reducing raising/falling time. Hence, the new short circuit power is 85% of original.

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.

Answer:

The plan was probably to increase clock speed by 15%. However reducing Tshort by 0.15 ns can at most decrease clock period by $2 \times 0.15 = 0.30$ *ns, while clock period* $\gg$ *1 ns. Therefore, it does not work.*

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

  Answer:
  Unreasonable*: with performance we care about the volume of a system, not its area.*

- MIPs/Watt

  Answer:
  reasonable*: bigger is better; e.g. cell-phone*

- Watts/cm^3

  Answer:
  reasonable*; smaller is better; server farm*

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?

Answer:

$$
\begin{aligned}
\mathit{MaxClockSpeed} &\propto \frac{(\mathit{VoltSup} - \mathit{VoltThresh})^2}{\mathit{VoltSup}} \\
\frac{\mathit{MaxClockSpeed}_1}{\mathit{MaxClockSpeed}_2} &= \left(\frac{(\mathit{VoltSup}_1 - \mathit{VoltThresh})^2}{\mathit{VoltSup}_1}\right)\left(\frac{\mathit{VoltSup}_2}{(\mathit{VoltSup}_2 - \mathit{VoltThresh})^2}\right) \\
\mathit{MaxClockSpeed}_1 &= \mathit{MaxClockSpeed}_2\left(\frac{(\mathit{VoltSup}_1 - \mathit{VoltThresh})^2}{\mathit{VoltSup}_1}\right)\left(\frac{\mathit{VoltSup}_2}{(\mathit{VoltSup}_2 - \mathit{VoltThresh})^2}\right. \\
\mathit{MaxClockSpeed}_1 &= 200\mathit{MHz}\left(\frac{(1.5V - 0.8V)^2}{1.5V}\right)\left(\frac{3V}{(3V - 0.8V)^2}\right) \\
\mathit{MaxClockSpeed}_1 &= 40\mathit{MHz}
\end{aligned}
$$

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.

Answer:

1. *Calculate original power consumption (activity factor and area)*

$$\begin{aligned} & 0.6 \times 200 \\ = \; & 120 \end{aligned}$$

2. *Calculate power budget*

$$\begin{aligned} & 120/1.45 \\ = \; & 82.76 \end{aligned}$$

3. *Calculate length of window*

$$\begin{aligned} & 50\text{cycles}/40\% \\ = \; & 125\text{cycles} \end{aligned}$$

4. *Calculate activity factor of main circuit with 100% effective clock gating scheme*

$$\begin{aligned} & 0.60 \times (50\text{cycles} + 20\text{cycles})/125 cycles \\ = \; & 0.34 \end{aligned}$$

5. *Calculate power consumption of main circuit with clock gating*

$$\begin{aligned} & 0.34 \times 200 \\ = \; & 68 \end{aligned}$$

6. *Calculate available power for clock gating circuit*

$$\begin{aligned} & 82.76 - 68 \\ = \; & 14.76 \end{aligned}$$

7. *Calculate available area for clock gating circuit*

$$\begin{aligned} & 14.76/0.60 \\ = \; & 24.6\text{cells} \end{aligned}$$

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.

Answer:

Use a cycle-count style of design. Each time the valid data arrives, the counter is set to 0. The counter increments if valid data is not input. The counter saturates at 20, at which time we know that the valid data has left the circuit, and the clock can be turned off.

The clock is enabled/disabled by AND*ing it with the output of the cycle-count state machine.*

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?

Answer:

Total power:

$$\begin{aligned} \mathit{Power} &= (\mathit{ActFact} \times \mathit{ClockSpeed} \times \tfrac{1}{2}\mathit{CapLoad} \times \mathit{VoltSup}^2) \\ &+ (\mathit{ActFact} \times \mathit{ClockSpeed} \times \mathit{TimeShort} \times \mathit{IShort} \times \mathit{VoltSup}) \\ &+ (\mathit{ILeak} \times \mathit{VoltSup}) \end{aligned}$$

Only need to reduce dynamic power, therefore neglect static (leakage) power.

Neglect short circuiting current.

$$\mathit{Power} = (\mathit{ActFact} \times \mathit{ClockSpeed} \times \frac{1}{2}\mathit{CapLoad} \times \mathit{VoltSup}^2)$$

$$\begin{aligned} ClockSpeed' &= 1.1\,ClockSpeed \\ Power' &= Power \end{aligned}$$

$$\begin{aligned} Power' &= Power \\ (ActFact \times ClockSpeed' \times \tfrac{1}{2} CapLoad \times VoltSup'^2) &= (ActFact \times ClockSpeed \times \tfrac{1}{2} CapLoad \times VoltSup^2) \\ (ClockSpeed' \times VoltSup'^2) &= (ClockSpeed \times VoltSup^2) \\ (1.1\,ClockSpeed \times VoltSup'^2) &= (ClockSpeed \times VoltSup^2) \\ VoltSup' &= \sqrt{\frac{(ClockSpeed \times VoltSup^2)}{1.1\,ClockSpeed}} \\ VoltSup' &= 0.95\,VoltSup \end{aligned}$$

We need to decrease the supply voltage to be 95.3% of its original value.

9.6.2 Supply Voltage

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

Answer:

Decreasing the supply voltage will bring it closer to the threshold voltage. As the difference between the supply and threshold voltage decreases, it will limit the maximum frequency that the circuit can run at.

This then leads to decreasing the threshold voltage, which will then increase the leakage current, and raise the static power dissipation:

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

Answer:

Scaling the supply voltage (V) reduces all components of the power equation. Switching power is proportional to the square of supply voltage, so switching power reduces the most.

9.7.2 Transistor Sizing

The transistors were made larger.

Answer:

Resizing transistor to increase the width to length ratio decreases the resistance of the transistor, which makes it faster. This means that the supply voltage can be reduced to save power while maintaining performance.

However, increasing the width to length ratio increases the capacitance. After a certain point, the capacitance increase becomes more significant than the reduction in supply voltage, causing power to increase.

Therefore, resizing is adjusting supply voltage and load capacitance to minimize their product in the switching power component.

9.7.3 Adding Registers to Inputs

All inputs to functional units are registered

Answer:

When inputs are registered, the activity factor is decreased, which decreases the dynamic power. The activity factor decreases because combinational gates might have glitches, which causes the activity factor to be greater than 1. In contrast, a flip-flop will have at most one transition per clock cycle, and so its activity factor will at most 1.

A counter argument is that the additional circuitry for the flops will consume more power. For reasonable size functional units, adding flops would be only a minor increase to the area, and hence the power increase from adding the flops would likely be less than the savings from decreasing the activity factor.

9.7.4 Gray Coding

Gray coding of signals is used for address signals.

Answer:

Gray coding reduces the activity factor on signals that typically change by 1 or a small amount. Address signals have this behaviour, in contrast to data signals, where consecutive values are often completely different.

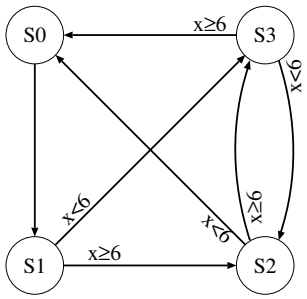
Reducing the activity factor will reduce the dynamic power.

However, there are complications in using gray coding. For example, the compiler would need to use gray coding when calculating addresses, and the computer's datapath would need a gray-code adder to compute address values.

Overall, the added design complexity of gray coding is probably too great a cost to pay except in situations where the encoding is entirely internal to the system being designed.

9.8 State Encoding

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

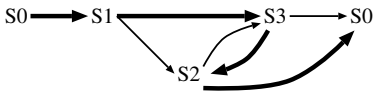


NOTES:

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.

Answer:

1. *To find the best encoding (minimum power consumption), we would need to evaluate* all *possible encodings. Instead, we will optimize for the most common transitions and consider three different encodings: binary, Gray, one-hot. If the best of these encodings has a high activity factor, then we will consider a custom encoding.*
2. *We unroll the state machine to a sequence of states, so as to better understand its overall behaviour. The most common transitions* ($x < 6$) *are drawn with bold lines.*



   The most common sequence is: S0, S1, S3, S2, S0. We will optimize for it. This sequence visits each state once, so we will treat it as if it was a counter.
3. *Overall design: We don't have a simple counter, because some states have two possible next-states. As inputs to the state machine, we will need the state bits and one-bit for* $x \geq 6$.
4. *One-hot vs Binary: Both one-hot and binary encodings have two transistions per clock cycle. Because some states have two outgoing edges, a one-hot encoding will require lookup tables to make decisions. The capacitance of the lookup tables removes the advantage of a one-hot encoding over a binary encoding. We eliminate the one-hot encoding from consideration.*
5. *Binary vs Gray: Both the binary and Gray encodings can be implemented with 2 bits of state. Gray has a lower activity factor, and so we choose Gray.*
6. *Assign encoding based on most common sequence, then reorder for S0, S1, S2, S3.*

Most common sequence		***Normal order***	
S0	*00*	*S0*	*00*
S1	*01*	*S1*	*01*
S3	*11*	*S2*	*10*
S2	*10*	*S3*	*11*

7. *We achieved an activity factor of just one transition per common transition and a minimum size design of two bits, so we are happy with the Gray code and do not pursue a custom encoding.*
8. *Ironically, the final encoding is the same as a binary encoding. This is just a curious coincidence, and not a sign that we should have jumped straight to a binary encoding.*

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.

Answer:
Leakage current — because same design, but different fabrication process resulted in power change.

9.9.2 Experiment

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

Answer:
Measure power consumption of circuit with clock turned off, if higher than expected, then leakage current is the problem.

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.

Answer:

Reduce the clock frequency.