ECE 327 Solution to Final

2017t1 (Winter)

		Total Marks	Approx. Time	Page
Q1	VHDL Semantics	5	10	2
Q2	Area Analysis	15	25	3
Q3	Performance	20	25	5
Q4	Latch Design	10	15	9
Q5	Monotone Speedup	15	15	11
Q6	Power and Delay	20	25	13
Q7	Clock Gating	15	15	15
Totals		100	130	

Q1 (5 Marks) VHDL Semantics

(*estimated time: 10 minutes*)

Is it possible for a simulation round to contain exactly one simulation cycle? **For full marks, you must justify your answer in terms of the VHDL semantics.**

Answer:

Yes, a simulation round in which the projected assignments do not change any visible values will contain exactly one simulation cycle. This can happen in two different ways:

1. *There are one or more resumed processes at the beginning of the simulation round, but when the processes execute, no projected assignments are executed (* e.g.*, because the assigments are inside an if-then statement whose condition is false).*
2. *Projected assignments are executed, but the new projected values are the same as the visible values.*

Marking:

5 marks *Correct answer with correct and detailed justification.*

3 marks *Correct answer with incomplete or partially correct justification.*

2 marks *Incorrect answer with substantial correct information in justification.*

4.5 marks *No: need at least two cycles*

3 marks *Yes: 1 cycle with change on signals*

3 marks

Q2 (15 Marks) Area Analysis

(*estimated time: 25 minutes*)

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

NOTES:

1. The code implements the equation: `z = min(a+b, b+c)`.
2. The signals `a`, `b`, `c`, `ab`, `bc`, `diff` and `z` are all declared to be `signed( 15 downto 0 )`.
3. The inputs are `a`, `b`, and `c`.
4. The output is `z`.
5. Optimizations are allowed, as long as the externally visible input-to-output behaviour of the system does not change.
6. For full marks, you must justify your answer with a drawing and/or text.

```
process begin
  wait until rising_edge(clk);
  ab        <= a + b;
  bc        <= b + c;
end process;

diff      <= ab - bc;
ab_lt_bc  <= diff(15);
z         <= ab when ab_lt_bc = '1' else bc;
```

Answer:

1. *Simplify equation:*
   ```
   z  =  min(a+b, b+c)   Original equation
   z  =  b + min( a, c)  Simplified equation
   ```
2. *Simplify VHDL code:*
   - *Use simplified equation*
   - *Use retiming to push register to* `z`*, to reduce number of flip-flops and enable use of one LUT for an add and a 2:1 mux.*

   ```
   diff <= a - c;
   a_lt_b <= diff(15);

   src2 <= a when a_lt_b = '1' else c;
   process begin
     wait until rising_edge(clk);
     z <= b + src2;
   end process;
   ```
3. *Area analysis:*

	LUTs	***Flops***
diff	*16*	*0*
src2	*0*	*0*
z	*16*	*16*
total	*32*	*16*

32 FPGA cells in total

Marking:

+4 marks	*Simplify equation*
+4 marks	*Combine add and 2:1 mux into single LUT*
+4 marks	*1 cell per bit*
+3 marks	*cells = max(LUTs, flops)*

Q3 (20 Marks) Performance

(*estimated time: 25 minutes*)

You work for Blueberry Solutions, a leading innovator in hardware system design. Your cousin just completed an IPO of her software company and used some of the money to purchase Blueberry Solutions. She's promoted you from the drudgery of optimizing Waterluvian filters to the glamourous world of edge detectors.

Two of your group members have proposed performance optimizations for your next edge detector circuit. You have the resources to implement only one of the optimizations.

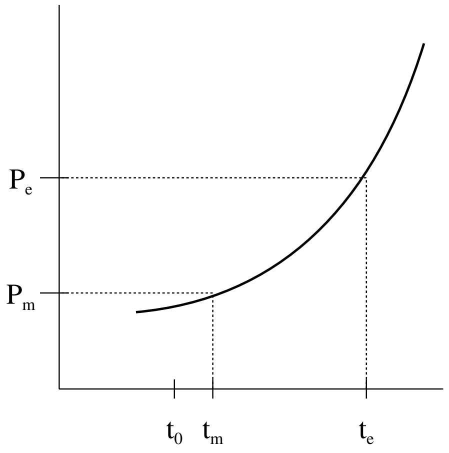
NOTES:

1. Performance is measured in MPPS: Mega-Pixels Per Second.
2. Eric Edge has proposed an optimization that will increase MPPS by 90% for pixels that are on an edge, but will delay the project release date by 25 weeks.
3. Mark Dark has proposed an optimization that will increase MPPS by 30% for pixels that are *not* on an edge, but will delay the project release date by 5 weeks.
4. Your current design has the same performance for all pixels, regardless of whether they are on an edge.
5. The average MPPS of edge detectors increases by 2% each week.

What percentage of pixels must be on an edge for Eric's optimization to provide the same performance *relative to the average edge-detector when the product will be released* as Mark's optimization will provide *relative to the average edge-detector when the product will be released*. **For full marks, you must justify your answer.**

Answer:

$$
\begin{aligned}
E &= \textit{Percentage of pixels on an edge}\\
P_e &= \textit{Performance with Eric Edge's optimization}\\
P_m &= \textit{Performance with Mark Dark's optimization}\\
P_{avg}(t) &= \textit{Performance of average edge-detector at time } t\\
&= P_{avg}(t_0) \bullet 1.02^t\\
t_e &= 25 \quad \textit{Eric's delay}\\
t_m &= 5 \quad \textit{Mark's delay}\\
T_o &= \textit{Original time to process a pixel}\\
T_e &= \textit{Eric's time to process a pixel}\\
&= E \bullet T_o/1.90 + (1-E) \bullet T_o\\
T_m &= \textit{Mark's optimized time to process a pixel}\\
&= E \bullet T_o + (1-E) \bullet T_o/1.30
\end{aligned}
$$



$$
\begin{aligned}
P_{avg}(t_e) &= \textit{Avg performance at Eric's completion time}\\
&= P_{avg}(t_0) \bullet 1.02^{25}\\
&= 1.64
\end{aligned}
$$

$$
\begin{aligned}
P_{avg}(t_e) &= \textit{Avg performance at Mark's completion time}\\
&= P_{avg}(t_0) \bullet 1.02^{5}\\
&= 1.10
\end{aligned}
$$

Find ratio between P_e and P_m for equal relative performance

$$\frac{P_e}{P_{avg}(t_e)} = \frac{P_m}{P_{avg}(t_m)}$$

Solve for P_e/P_m

$$\begin{aligned}
\frac{P_e}{P_m} &= 1.02^{t_e - t_m} \\
&= 1.49 \\
\frac{T_m}{T_e} &= 1.49 \\
T_m &= 1.49 \bullet T_e
\end{aligned}$$

Solve for E

$$\begin{aligned}
E \bullet T_o + (1-E) \bullet \frac{T_o}{1.30} &= 1.49 \left(E \bullet \frac{T_o}{1.90} + (1-E) \bullet T_o \right) \\
E\left(1 - \frac{1}{1.30} - \frac{1.49}{1.90} + 1.49\right) &= 1.49 - \frac{1}{1.30} \\
E &= \frac{0.72}{0.93} \\
&= 77\%
\end{aligned}$$

77% of the pixels need to be on an edge for the two optimizations to provide equal performance relative to the average performance at the time that they are finished.

Incorrect approach: Use weighted average of pixels/time, not time/pixel

$$
\begin{aligned}
P_e &= \textit{Eric's optimized performance} \\
&= E \bullet P_o \bullet 1.90 + (1 - E) \bullet P_o \\
P_m &= \textit{Mark's optimized performance} \\
&= E \bullet P_o + (1 - E) \bullet P_o \bullet 1.30
\end{aligned}
$$

$$
\begin{aligned}
P_e &= 1.49 \bullet P_m \\
E \bullet P_o \bullet 1.90 + (1 - E) \bullet P_o &= 1.49(E \bullet P_o + (1 - E) \bullet P_o \bullet 1.30) \\
E \bullet 1.90 + 1 - E &= 1.49 \bullet E + 1.49 \bullet 1.30 - E \bullet 1.49 \bullet 1.30 \\
0.90 \bullet E + 1 &= 1.49 \bullet E + 1.937 - 1.937 \bullet E \\
1.347 \bullet E &= 0.937 \\
E &= 0.70
\end{aligned}
$$

Marking:

+4 marks	*Exponential growth in performance*
+4 marks	*Mark and Eric's performance relative to avg performance*
+4 marks	*Weighted avg of edge and non-edge pixels*
+4 marks	*Weighted avg uses time/pixel not pixel/time*
+4 marks	*Ratio between Eric and Mark's performance*
+4 marks	*Strategy and clarity*

Q4 (10 Marks) Latch Design

(*estimated time: 15 minutes*)

The circuit below is a correct active-high latch.

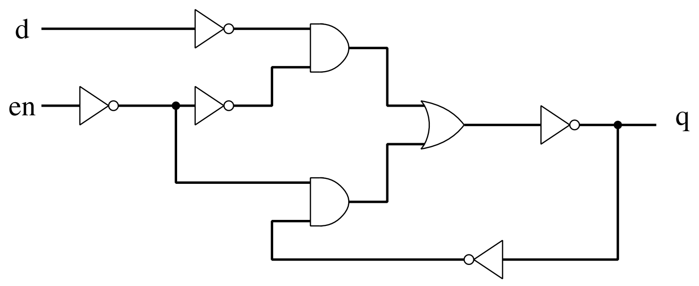
Is it possible to modify the circuit to decrease $T_{su} - T_{ho}$, that is, reduce the difference between the setup time and the hold time?

If **yes**, then draw the modified latch and calculate the new setup and hold times.

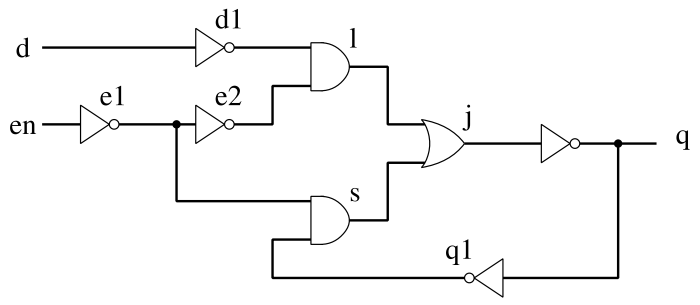
If **no**, the justify why you cannot reduce $T_{su} - T_{ho}$.

NOTES:

1. Each gate has a delay of 1 time unit.
2. If you modify the latch:
   - Your modified latch must be a correctly working active-high latch.
   - Make the minimum modifications necessary.



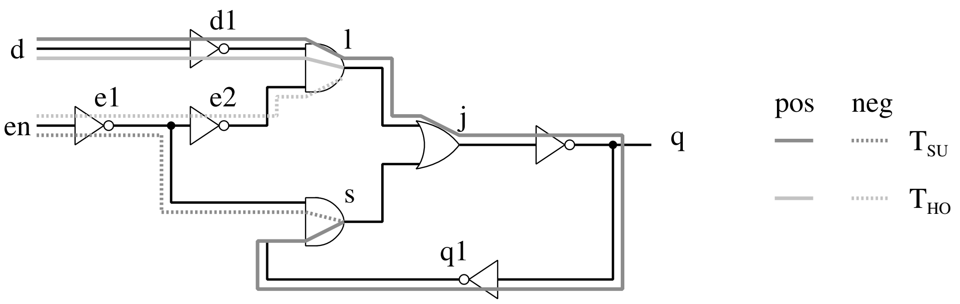
Answer:



$$T_{su} = \textit{delay}(d \rightarrow s) - \textit{delay}(en \rightarrow s)$$

$$T_{ho} = \textit{delay}(en \rightarrow l) - \textit{delay}(d \rightarrow l)$$

$$T_{su} - T_{ho} = \textit{delay}(d \rightarrow s) - \textit{delay}(en \rightarrow s) - \textit{delay}(en \rightarrow l) + \textit{delay}(d \rightarrow l)$$



The circuit is already a minimal latch, in that there aren't any gates that we can delete without introducing a bug. Therefore, we must add gates, not delete existing gates.

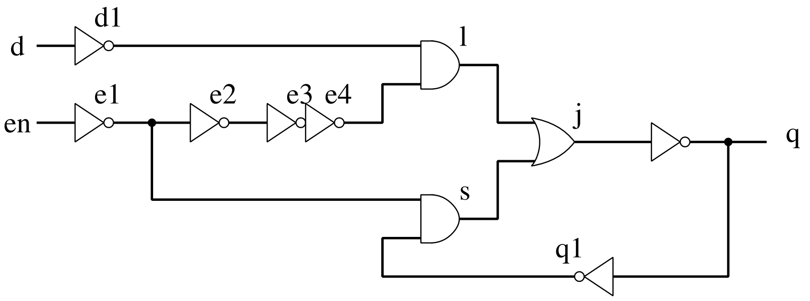
To reduce $T_{su} - T_{ho}$, we want to increase delay$(en \to s)$ or delay$(en \to l)$.

For the latch to transition correctly into store mode, we need the load path to turn off after the store loop turns on, which means:

delay$(en \to l) >$ delay$(en \to s)$

Thus, we will increase delay$(en \to l)$ and not increase delay$(en \to s)$.

To increase the delay without affecting functionality, we need to introduce a pair of inverters $(e3, e4)$.



The timing parameters are: T_{su} = 6-2 = 4, T_{ho} = 5-2 = 3

Marking:

+3 marks	*Reduced $T_{su} - T_{ho}$*
+3 marks	*Preserved functionality of active high latch*
+2 marks	*Minimal changes*
+2 marks	*Timing parameters*

If said "no"

+1.5 marks	*Tsu path*
+1.5 marks	*Tho path*
+1.5 marks	*en to load-enable must be longer than en to store-enable*
+1.5 marks	*Difference between paths, analysis*

Q5 (15 Marks) Monotone Speedup

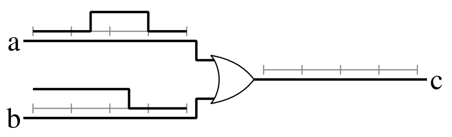
(*estimated time: 15 minutes*)

Use *one* of the excitations and circuits below to demonstrate a problem that can occur if your critical path algorithm does not satisfy the monotone speedup correctness criterion.

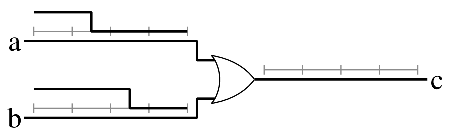
Instructions:

1. Write down any assumptions you use.
2. Choose one of the excitations and circuits (A, B, C, or D).
3. For the excitation you chose, draw the waveform for c.
4. Draw a modified excitation that illustrates the problem.
5. For your modified excitation, use the same gate as the excitation you chose (*e.g.*, OR for A or B, AND for C or D).
6. Briefly explain the problem.

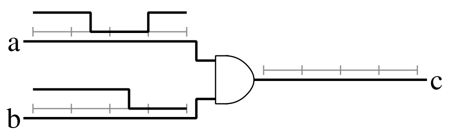
A



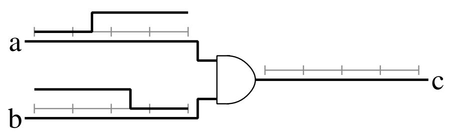
B



C



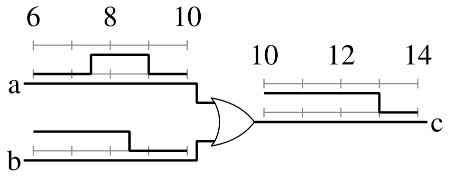
D



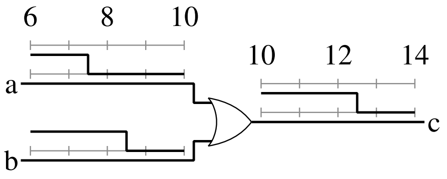
Answer:

Original waveforms *(only **C** is needed)*

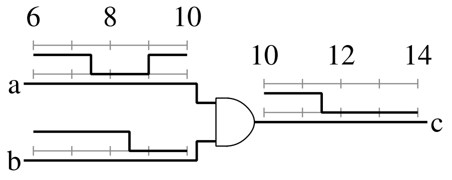
A



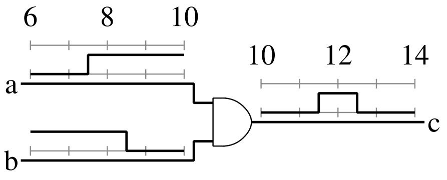
B



C

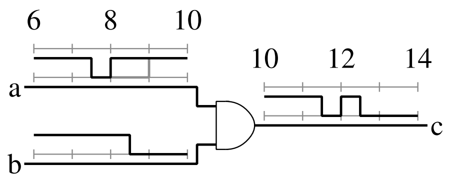


D



Assumptions *None needed.*

Excitation ***C***



Explanation *With normal timing, the last edge on* c *occurs at 11.5. If the last edge on* a *is sped up to arrive at 8 rather than at 9, then the last edge on* c *occurs at 12.5.*

The problem is that if we do our critical path analysis based only upon the worst-case delays of the gates, we would use a delay of 11.5 for the propagation delay to a*. Ignoring all other timing parameters, we could conclude that the circuit would work correctly with a clock period of 11.5. But if some of the gates driving* a *have average (or, better than worst-case) delays, then we would need a clock period of at least 11.5 for the circuit to work correctly.*

Using the worst-case delays for the gates could underestimate *the minimum clock period.*

Marking:

+2 marks *Original waveform for* c *is correct*

+2 marks *Choice of waveform*

+3 marks *Modified waveform shows one input sped up*

+2 marks *Modified waveform for* c *is correct*

+2 marks *Explanation: Go from worst-case to better-than-worst-case timing of one or more gates*

+2 marks *Explanation: A new edge, later than other edges, appears on output*

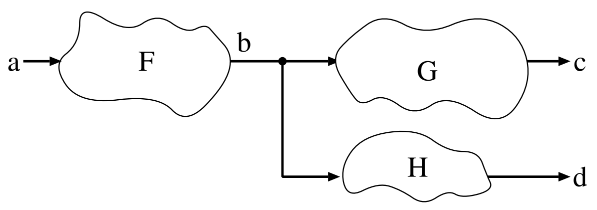
+2 marks *Explanation: Can underestimate clock period*

Q6 (20 Marks) Power and Delay

(*estimated time: 25 minutes*)

You've just returned from an extended, and well-deserved, holiday of kite-surfing and trekking in New Zealand. When you return, you discover two surprises. Spring has arrived early in Waterloo, and your employer, Blueberry Solutions, is making a bold move out of the world of FPGAs and into ASICs.

With ASICs, you have options to choose different supply voltages for different parts of your chip. Your task is to choose the supply voltages for modules F, G, and H below so as to minimize power consumption without increasing delay above the delay where all three modules use the medium supply voltage (V_m).



Voltages

V_h	1.20 V	High VDD
V_m	1.15 V	Medium VDD
V_l	1.00 V	Low VDD
V_{th}	0.70 V	Threshold voltage

	Delay with V_m	**Area**
F	150 ps	500 GE
G	250 ps	600 GE
H	100 ps	300 GE

NOTES:

1. Delay is measured in pico-seconds (ps), which is 10^{-12} seconds.
2. Area is measured in gate equivalents (GE), where 1 GE is the area of a 2-input NAND gate. The NAND gate is used because it is the smallest 2-input gate in a cell library.
3. The modules F, G, and H are purely combinational.

Answer:

1. *Using a supply voltage of V_m everywhere, the critical path is through F and G with a delay of 400 ps.*
2. *To minimize power, we will try to replace V_m with V_l on at least one of the modules.*
3. *Compute delay for each module with V_l and V_h*

$$\textit{delay} \propto \frac{V_{DD}}{(V_{DD}-V_{th})^2}$$

	V_l	V_m	V_h
F	293.48	150.00	126.78
G	489.13	250.00	211.30
H	195.65	100.00	84.52

4. *H is not on the critical path, so we definitely want to reduce its supply voltage. The delay through F and H becomes: 150+196=346.*
5. *If we decrease the supply voltage for G, the delay through G by itself will exceed the current delay of 400 ps.*

6. *If we decrease the supply voltage for F, even with an increased supply voltage for H, the path through F and H will exceed the current delay: 293+211=504.*
7. *Answer:*

F	V_m
G	V_m
H	V_l

Marking:

+2 marks	*Assume leakage power is negligible*
+2 marks	*Original delay is 400 ps*
+3 marks	*Idea that want to reduce voltage for H*
+3 marks	*Use of delay and VDD equation*
+3 marks	*Use of eqn Pwr* $\propto$ *Area* $\bullet V_{dd}^2$
+3 marks	*Show that new delay is less than original*
+2 marks	*Show that using* V_h *for F and* V_l *for G is too slow.*
+2 marks	*Strategy and analysis*
-1 marks	*Incorrect or irrelevant assumption*

Q7 (15 Marks) Clock Gating

(*estimated time: 15 minutes*)

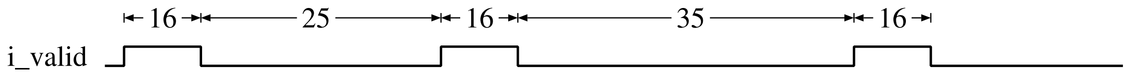
Your task is to minimize the energy per parcel for the system described below.

NOTES:

1. The system uses a "cycle count" clock gating scheme.
2. The latency is 20 clock cycles.
3. Parcels always arrive in a sequence of 1 parcel per clock cycle for 16 consecutive clock cycles, followed by some number of bubbles.
4. You have two choices for the "bubble schedule" (the number of bubbles that come after the 16 parcels):

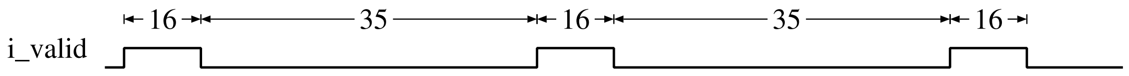
A The number of bubbles ranges from 25 to 35 with a uniform distribution.

Example waveform:



B The number of bubbles is always 35.

Example waveform:



Answer:

Assume that the clock gating schemes are 100% effective, therefore PctBusy=PctClk.

$$\mathit{Pwr} \propto \mathit{PctClk}$$

$$\begin{aligned} \mathit{J/pcl} &= \mathit{J/sec} \bullet \mathit{sec/clk} \bullet \mathit{Pcl/clk} \\ \mathit{E/pcl} &= \mathit{Pwr} \bullet \mathit{ClkPer} \bullet 1/\mathit{Tput} \\ \mathit{E/pcl} &\propto \frac{\mathit{PctClk}}{\mathit{Tput}} \end{aligned}$$

	NumPcls	*Lat*	*NumBubbles*	*LenWindow*	*LenClk*	*PctClk*	*Tput*	*E/Pcl*
Unpredictable	*16*	*20*	*30*	*46*	*36*	*0.78*	*0.35*	*2.25*
Constant	*16*	*20*	*35*	*51*	*36*	*0.71*	*0.31*	*2.25*

Both bubble schedules have the same energy per pixel. This is because both clock gating schemes are 100% effective. So, power is consumed only when parcels are in the system, and the bubble schedule does not affect the behaviour of parcels.

Marking:

+2 marks *LenWindow*

+2 marks *LenClk*

+3 marks *PctClk*

+3 marks *Tput*

+3 marks *Energy/pcl*

+2 marks *Strategy and clarity*

2 marks *Some work that is correct.*

5 marks *“B” with justification that clock gating scheme will be simpler with a constant number of bubbles.*

5 marks *“A” with justification that will have fewer wasted clock cycles because using the average number of bubbles.*

7 marks *“B” with justification that will have lower throughput and so will consume less power.*