ECE 327 Solution to Final

2018t1 (Winter)

Q1 (6 Marks) Metrics

(*estimated time: 15 minutes*)

NOTES:

1. You do *not* need to justify your answers for this question.

Q1a (2 Marks) Optimizing Studying

You are trying to apply the principles you learned in ECE-327 to optimize your study plan for finals. You know that after some amount of studying, you begin to hit diminishing returns, in that you can put in much more time studying but earn only a few more marks. You want to maximize the efficiency of your studying while still earning a reasonable mark on the exam. Which one of the optimality measures below best captures your goal?

			$\frac{M}{t}$	$\frac{M^2}{t}$	$\frac{1}{(100\%-M)\cdot t}$	$\frac{M}{(100\%-M)\cdot t}$
M	Mark you earn					
t	Time you spend studying					
		Best:		$\surd$		

Answer:

An explanation, not part of the exam answer:

$\frac{M}{t}$ measures the efficiency of studying. $\frac{M^2}{t}$ also takes into the account the mark that is earned. The terms with $100\%-M$ in the denominator go to ∞ as M goes 100%.

Q1b (2 Marks) Battery Life

You are comparing several Waterluvian filters and want to choose the one that will process the most pixels on one battery charge. All of the filters use the same type and size of battery. Which one of the metrics below is best for this?

			$\frac{E}{pix}$	$\frac{P}{pix}$	$\frac{E\cdot F}{pix}$	$\frac{P\cdot T}{pix}$
E	Energy					
P	Power					
T	Throughput					
F	Clock frequency					
pix	Pixel	**Best:**	$\surd$			

Answer:

Explanation, not part of the exam answer:

Batteries store energy. The number of pixels processed on one battery charge is the energy in the battery divided by the energy consumed to process one pixel.

Q1c (2 Marks) Chip Temperature

You are comparing several Waterluvian filters and want to choose the one that will have the lowest temperature on the chip (*e.g.*, so that you can use a smaller and cheaper fan to cool it). Each filter is on a chip by itself: the only circuitry on the chip is the Waterluvian filter and the entire Waterluvian filter fits on one chip. Which one of the metrics below is best for this?

E	Energy				
P	Power	$\frac{E \cdot T}{A}$	$\frac{P}{A}$	$\frac{P \cdot T \cdot F}{A}$	$\frac{P \cdot T \cdot A}{F}$
T	Throughput				
F	Clock frequency				
A	Area of chip	**Best:**	$\surd$		

Answer:

Explanation, not part of the exam answer:

Electrical energy is converted into thermal energy. Power is the rate of energy consumption. Thermal energy is dissipated over an area. Power divided by area is an estimate of the temperature.

Marking:

2 marks *Each correct answer*

Q2 (20 Marks) The Spring, the Chirping Birds, the Snow

(*estimated time: 20 minutes*)

For each of the code fragments Q2a–Q2d:

1. Answer whether the code is *legal*
2. If the code is *illegal*: explain why, and proceed to the next code fragment.
3. Answer whether the code is *synthesizable*.
4. If the code is *unsynthesizable*: explain why, and proceed to the next code fragment.
5. Answer whether the code adheres to good coding practices, according to the guidelines for ECE 327.
6. If the code does *not follow good coding practices*: explain why.

NOTES:

1. All signals are `std_logic`.

Q2a

```
  process (a, b)
  begin
    if e = '1' then
      d <= a;
    else
      d <= b;
    end if;
  end process;

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

Answer:

BAD: latch, missing e in sensitivity list

Q2b

```
process begin
  wait until rising_edge( clk );
  if reset = '1' then
    d <= '0';
  end if;
end process;

process begin
  wait until rising_edge( clk );
  if reset /= '1' then
    d <= a;
    e <= d xor e;
  end if;
end process;
```

Answer:

BAD: multiple drivers on d, because d is the target signal in two processes.

Q2c

```
process ( i_valid ) begin
  if rising_edge( i_valid ) then
    b <= i_data;
  end if;
end process;

process begin
  wait until rising_edge( clk );
  c <= b xor c;
end process;
```

Answer:

BAD: using a data signal, i_valid, as a clock

Q2d

```
process begin
  wait until rising_edge( clk );
  if a = '1' then
    d <= b;
    e <= c;
  else
    d <= d xor e;
  end if;
end process;

f <=   e when (d xor e) = '1'
  else a xor b;
```

Answer:

GOOD

Marking:

Q2a–Q2c

1 mark *Each for Legal, Synth, Good*

2 marks *Explanation*

Q2d

5 marks *Correct answer*

1 mark *Each for Legal and Synth*

2 marks *Explanation*

Q3 (16 Marks) Area Analysis

(*estimated time: 20 minutes*)

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

NOTES:

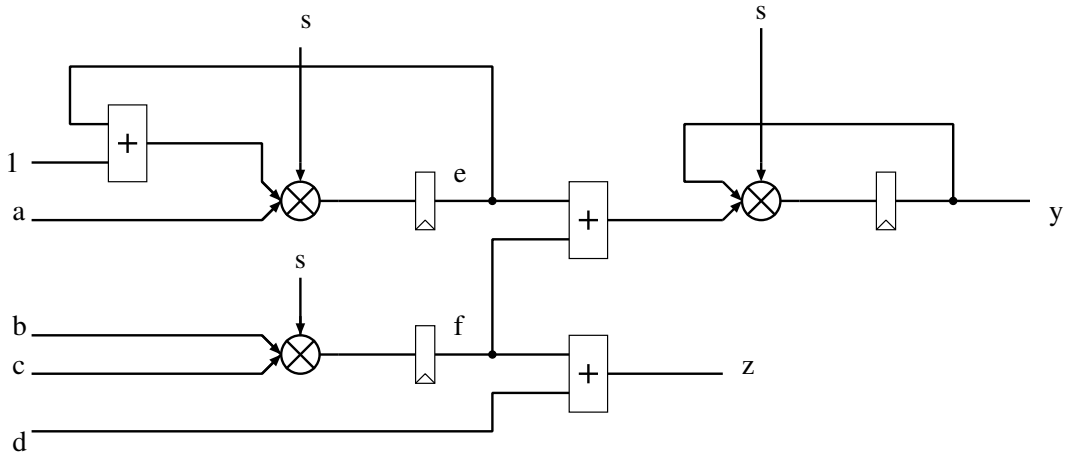
1. The signals `a`, `b`, `c`, `d`, `e`, `f`, `g`, `y` and `z` are all declared to be `signed( 15 downto 0 )`.
2. The signal `s` is declared to be `std_logic`.
3. The inputs are `a`, `b`, `c`, `d`, and `s`.
4. The outputs are `y` and `z`.
5. Optimizations are allowed, as long as the externally visible 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);
  if s = '1' then
    e <= e + 1;
    f <= b;
    g <= e + f;
  else
    e <= a;
    f <= c;
    g <= g;
  end if;
end process;

y <= g;
z <= f + d;
```

Answer:

1. Schematic



In class, we taught that a 2:1 mux on the input to an adder can be combined with the adder into 1 LUT per bit. Some students reversed the situation and combined a 2:1 mux with an adder on its input into 1 LUT per bit. It turns out the synthesis tools now also support this (this is a new feature in the tools). To be fair to students who might have discovered this behaviour during the project, this optimization was marked as correct. The solution below has two parts, first using the assumption, second using only the optimization taught in class.

2. *Area analysis (if assume that can combine a 2:1 mux with an add on one input into a single LUT, the new feature in the tools):*

	LUTs	**Flops**
`e`	16	16
`f`	16	16
`y`	16	16
`z`	16	0
total	64	48

cells = max(LUTS, Flops)

64 FPGA cells in total

3. *If assume that to combine a mux and an add into a single LUT, must have 2:1 mux on input to adder, then continue.*

4. *Initial area analysis:*

	LUTs	**Flops**
`e`	32	16
`f`	16	16
`y`	32	16
`z`	16	0
total	96	48

cells = max(LUTS, Flops)

96 FPGA cells in total

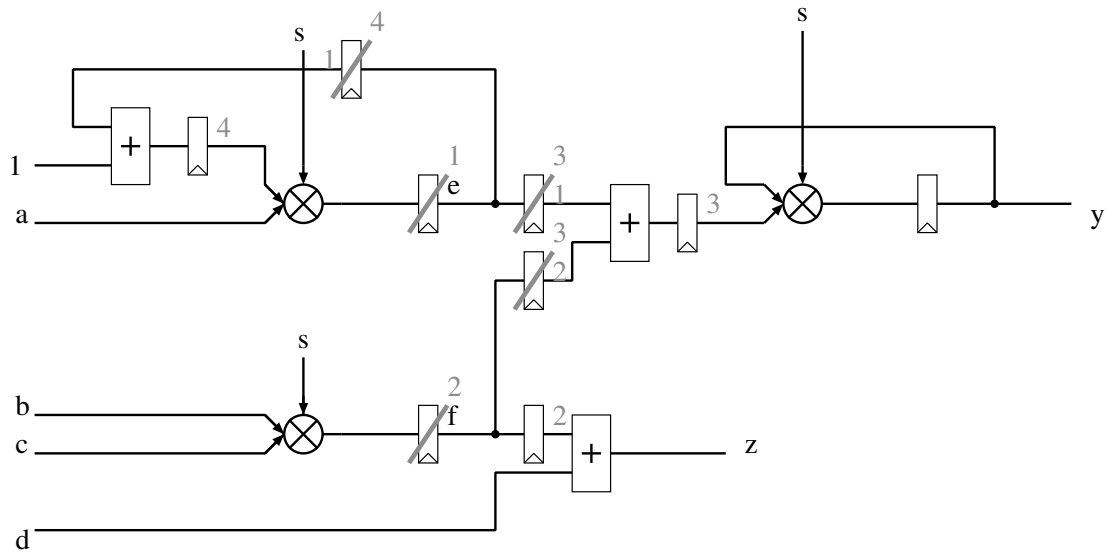
5. *Use chip-enable instead of mux on* `y`

	LUTs	**Flops**
`e`	32	16
`f`	16	16
`y`	16	16
`z`	16	0
total	80	48

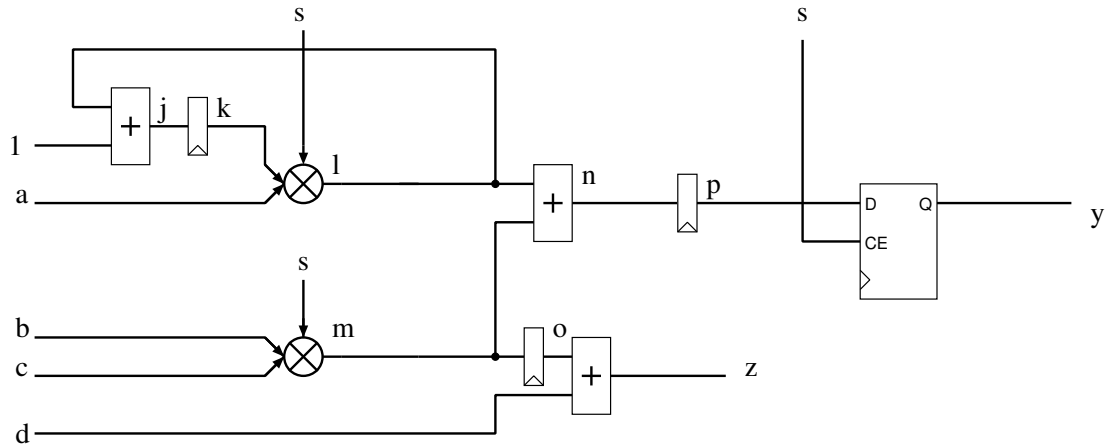
cells = max(LUTS, Flops)

80 FPGA cells in total

6. *Retime to get 2:1 mux as input to adder. In step 4 we can push the register through the adder, because `'1'` is a constant.*



7. *Change y from reg with feedback mux to reg with chip-enable. Add signal names to make area-analysis table easier to understand.*



8. *Area analysis:*
   The 2:1 mux for `l` is replicated and combined with the adder for `n` and with the adder for `j`. The combined mux and adder for `l` and `j` requires 16 LUTs. Similarly, `l` and `n` require 16 LUTs.

	LUTs	***Flops***
`l` and `j`	*16*	*0*
`k`	*0*	*16*
`l` and `n`	*16*	*0*
`m`	*16*	*0*
`o`	*0*	*16*
`p`	*0*	*16*
`y`	*0*	*16*
`z`	*16*	*0*
total	*64*	*64*

cells = max(LUTS, Flops)
64 FPGA cells in total

Marking:

+4 marks *Schematics and optimizations*

+3 marks *Analysis of adders*

+3 marks *Analysis of muxes (including combining 2:1 mux and adder)*

- **3 marks** *Correct analysis*
- **1 mark** *1 mistake, such as forgetting a mux*
- **0 marks** *Did not demonstrate that took into account muxes*

+2 marks *Analysis of registers*

+2 marks *1 cell per bit*

+2 marks *cells = max(LUTs, flops)*

Q4 (20 Marks) DFD

(*estimated time: 30 minutes*)

Design and analyze a dataflow diagram that implements the pseudocode below.

```
for idx in 0 to 255 {
  z = z - a + b + c + 123;
}
```

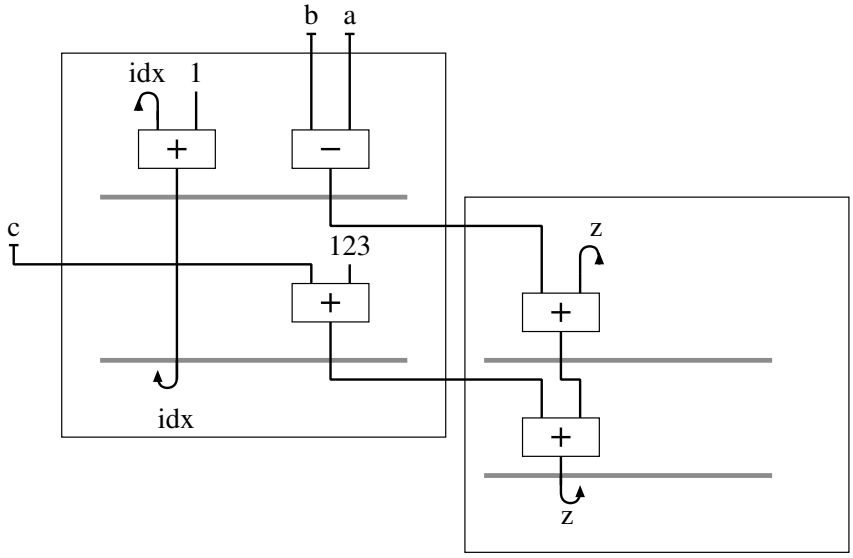
NOTES:

1. Read the expression for `z` carefully. It is *not*: `z - (a + b + c + 123)`.
2. `a`, `b`, and `c` are inputs.
3. `idx` is an interparcel varaible.
4. `z` is an interparcel variable and an output.
5. Optimization goals, in order of decreasing importance
   - Minimize clock period
   - Maximize throughput, up to a maximum of 1/2 parcels/clock-cycle. (There is no benefit, and no additional marks will be given, for achieving a throughput that is greater than 1/2.)
   - Minimize number of input ports
   - Minimize total number of datapath components (sum of number of adders and number of subtracters)
   - Minimize number of registers
   - Minimize latency
6. Your DFD *shall* show the increment of `idx` and the computation of `z`.
7. Your DFD shall *not* show the initialization of `idx` or the comparison of `idx` to detect the end of 256 iterations of the loop.
8. Your DFD shall *not* show the initialization of `z`.
9. You may ignore the area consumed by multiplexers.
10. You may schedule the input values to arrive in any order within an iteration.
11. You do *not* need to do any allocation.
12. Algebraic optimizations are allowed, so long as the externally visible functionality remains unchanged.

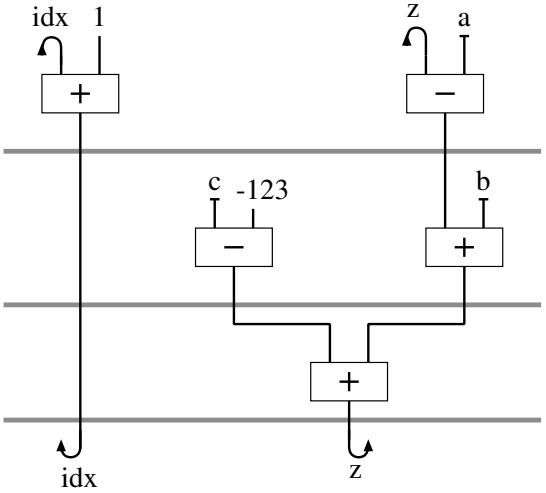
Marking:

+16 marks *DFD*

16 marks *tput=1/2, inp=2, add=2, sub=1, reg=3, lat=3*



14 marks *tput=1/3, inp=2, add=1, sub=1, reg=3, lat=3*



12 marks *tput=1/4, inp=2, add=1, sub=1, reg=3, lat=4*

-3 marks *interparcel variables not at top/bot of stage*

-3 marks *interparcel variables inconsistent with tput*

-2 marks *messy or difficult to understand*

-2 marks *suboptimal latency*

+4 marks *Analysis*

-1 mark *first mistake*

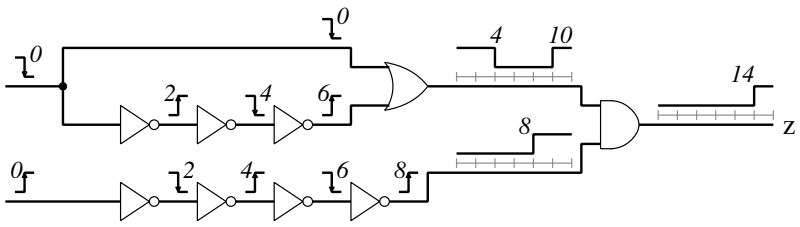
-0.5 marks *each additional mistake*

Q5 (16 Marks) Monotone Speedup

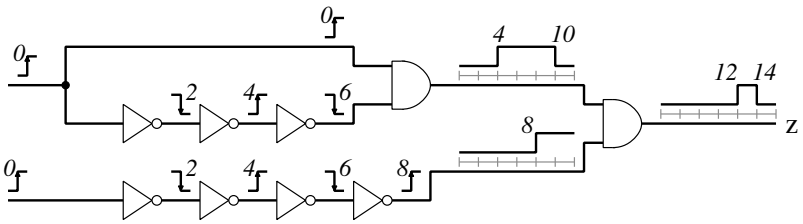
(*estimated time: 20 minutes*)

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.

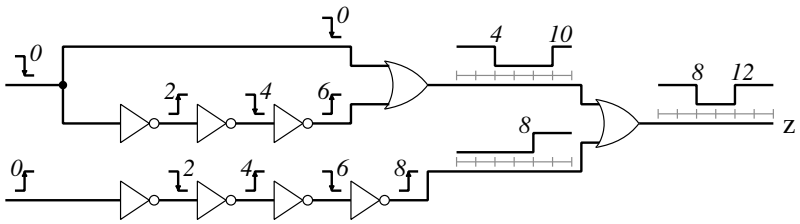
A



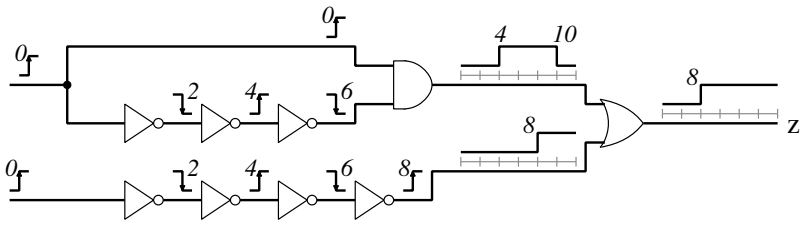
B



C



D



Answer:

NOTE: This solution is much longer and more detailed than an exam answer should be. The extra details are included to demonstrate the reasoning behing the solution.

Assumptions *None needed.*

Choose a circuit and excitation *We are trying to* reduce *the delay through some gates and* increase *the delay through the circuit. By increase the delay through the circuit, we mean that the last edge on the output* `z` *will occur later than with the original delays.*

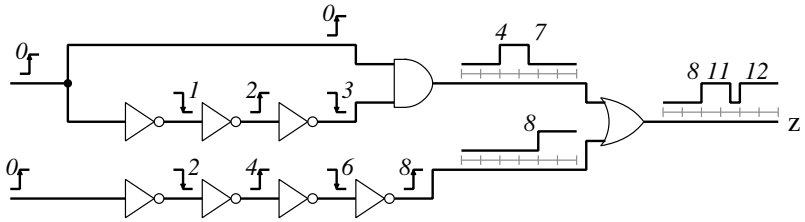
For this to happen, the last edge on `z` *in the original excitation must* not *be caused by the last edge on the input to* `z`*. Because, if the last output edge is caused by the last input edge, then we cannot create a* later *output edge.*

A, B *The last edge on* `z` *(at time 14) is already caused by the last input edge (at time 10).*
C *The last edge on* `z` *is caused by the rising edge at time 8. If we move this edge earlier than the rising edge at time 4, then the output will be a constant* `1`*.*
If we move the rising edge at time 10 earlier than the rising edge at time 8, then the rising edge on `z` *will move earler.*

D *The last edge on* `z` *is caused by the rising edge at time 4. If we move the falling edge at time 10 to be earlier than the rising edge at time 8, then* `z` *will get a glitch that ends later than 8.*

Speedup *To speedup the rising edge at 10 on the input to* `z` *to happen before 8, we need to reduce the delay through the 3 inverters by at least 2 time units. We choose to reduce the delay each inverter from 2 down to 1.*

The resulting behaviour is:



Explanation *With normal timing, the last edge on* `z` *occurs at 8. If we reduce the delay through the chain of 3 inverters, then the last edge on* `z` *occurs at 12.*

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 8 for the propagation delay of ***D****. Ignoring all other timing parameters and excitations, we could conclude that the circuit would work correctly with a clock period of 8. But by reducing some of the gate delays in the circuit to average (or, better than worst-case) delays, then we would need a clock period of at least 12 for the circuit to work correctly.*

If our critical path algorithm does not satisfy the monotone speedup correctness criteria, then using the worst-case delays for the gates could underestimate *the minimum clock period for the circuit.*

Marking:

+3 marks *Use the same input*

+3 marks *Decrease the delay through some gates*

+3 marks *Simulation is correct*

+3 marks *Last edge on output is later than with original gate delays*

+4 marks *Explanation*

- *Reduced delay of some gates*
- *Increased delay of circuit*
- *Critical path algorithm could underestimate delay of circuit*

Q6 (22 Marks) Power and Delay

(*estimated time: 30 minutes*)

You work for Pbπ, the *lead*ing developer of *pipe*lined Waterluvian filters. The CEO, Cassandra Scarlet, has promoted you to the top-secret R&D lab, codenamed "The Library". Dr. Scarlet has asked you to investigate the potential use of "approximate computing" in Waterluvian filters.

You are clueless about approximate computing, so you do some web surfing and learn that approximate computing is an optimization that can be used to increase performance or decrease power in digital hardware systems. The disadvantage is that the system will sometimes compute an incorrect result. A typical application is image processing, where users usually will not notice if a few pixels are incorrect.

Approximate computing works by decreasing the supply voltage and/or decreasing the clock period so that the delay through the combinational logic in some parts of the system is sometimes longer than the clock period. When this happens, some flip-flops might store incorrect values.

Your task is to evaluate the effect on power and energy if approximate computing is applied to Pbπ's Waterluvian filter.

The Pbπ Waterluvian filter currently has the following parameters:

	Current Values		
Supply voltage	1.5	V	
Threshold voltage	0.7	V	
Clock period	4.0	ns	
Dynamic power	15	mW	
Static power	2	mW	
Performance	100	MPPS	(mega-pixels per second)

Dr Scarlet has asked you to analyze the effects of the following new values with the same circuit:

	New Values	
Supply voltage	1.3	V
Clock period	3.5	ns

NOTES:

1. Short circuiting power is negligible.

Answer:

Activity factor remains constant

Leakage current remains constant

Q6a (12 Marks) Power

Calculate the new total power consumption of the Waterluvian filter

Answer:

If you're clueless about Waterluvian filers, the answer is: Dr. Scarlet, in the Library, with a lead pipe.

1. *Calculate new dynamic power*
   (a) *Dynamic power is proportional to* V_{DD}^2
   (b) *Dynamic power is inversely proportional to clock period (*T*)*
   (c) *Equation*

$$\begin{aligned} P'_D &= P_D \cdot \frac{{V'_{DD}}^2}{V_{DD}^2} \cdot \frac{T}{T'} \\ &= 15 \cdot \frac{1.3^2}{1.5^2} \cdot \frac{4.0}{3.5} \\ &= 12.88\,mW \end{aligned}$$

2. *Calculate new static power*
   (a) *Leakage power is proportional to* V_{DD}
   (b) *Equation*

$$\begin{aligned} P'_S &= P_S \cdot \frac{V'_{DD}}{V_{DD}} \\ &= 2 \cdot \frac{1.3}{1.5} \\ &= 1.73\,mW \end{aligned}$$

3. *Calculate new total power*

$$\begin{aligned} P' &= P'_D + P'_S \\ &= 12.88 + 1.73 \\ &= 14.61\,mW \end{aligned}$$

Marking:

2 marks *Assumptions*
2 marks *Strategy and clarity*
2 marks *Dynamic power is proportional to clock frequency*
2 marks *Dynamic power is proportional to supply voltage squared*
2 marks *Static power is proportional to supply voltage*
2 marks *Analysis includes both dynamic and static power correctly*

Q6b (10 Marks) Energy

Calculate the new energy per pixel of the Waterluvian filter.

Answer:

1. *Top level equation (play the units game)*

$$E = \text{Pwr} \cdot \frac{1}{\text{Perf}}$$

$$\frac{J}{\text{pix}} = \frac{J}{\text{sec}} \cdot \frac{\text{sec}}{\text{pix}}$$

2. *Performance is inversely proportional to clock period*

$$\begin{aligned} \text{Perf}' &= \text{Perf} \cdot \frac{T}{T'} \\ &= 100 \cdot \frac{4.0}{3.5} \\ &= 114\,\textit{MPPS} \end{aligned}$$

3. *Use energy equation*

$$\begin{aligned} E &= \text{Pwr} \cdot \frac{1}{\text{Perf}} \\ &= 14.61 \cdot 10^{-3} \cdot \frac{1}{114 \cdot 10^{6}} \\ &= 0.128\,\textit{nJ} \end{aligned}$$

Marking:

4 marks *Strategy and clarity*
3 marks *Performance*
3 marks *Energy*