ECE 327 Midterm Solution

2013t1 (Winter)

All requests for re-marks must be submitted in writing to Mark Aagaard before 8:30am on Monday March 11.

A random collection of midterms were photocopied. Exams that are submitted for re-marking will be verified against this set.

Q1 (20 Marks) VHDL Semantics

(estimated time: 15 minutes)

For the VHDL program below, calculate the values for the signals `a1`, `b1`, `c1`, `a2`, `b2`, and `c2` at 15ns.

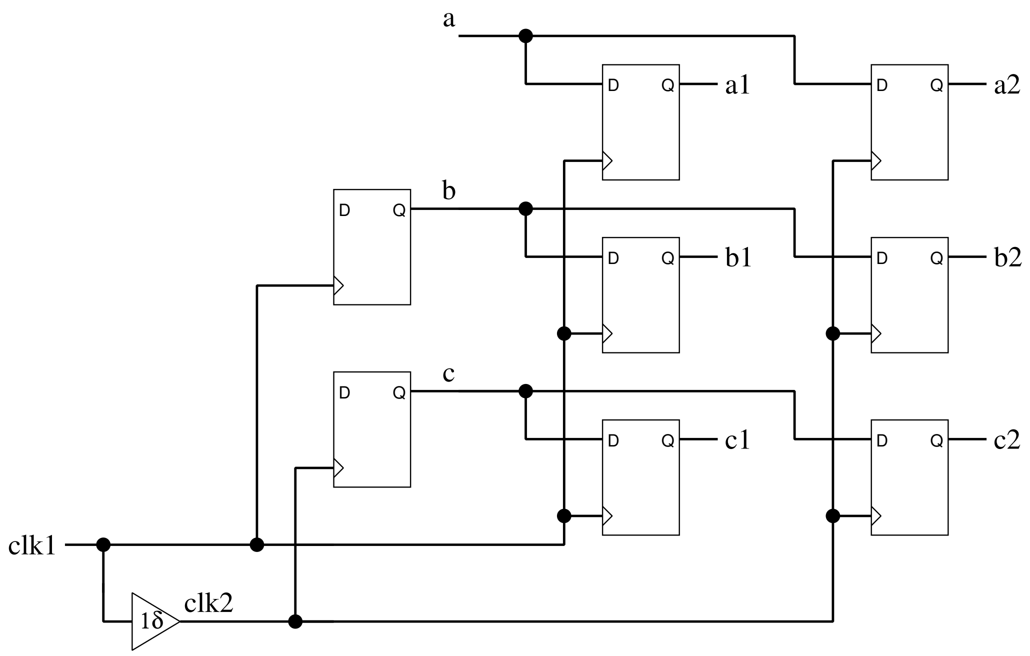
NOTES:

1. All of the processes are in the *same architecture.*
2. The signals `a1`, `b1`, `c1`, `a2`, `b2`, and `c2` are declared to be `unsigned(3 downto 0)`.
3. **For full marks you must justify your answer, using text and/or the waveform diagram. You may, but are *not* required to, show a delta-cycle simulation.**

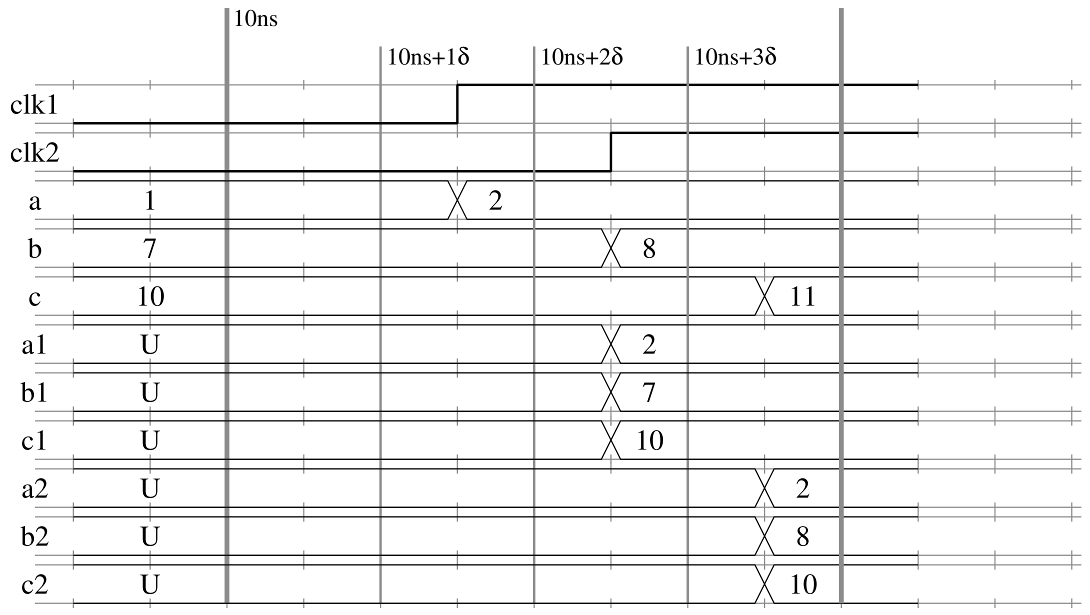
```
process begin
  clk1 <= '0';
  wait for 10 ns;
  clk1 <= '1';
  wait for 10 ns;
end process;

process begin
  a <= to_unsigned(  1, 4 );
  wait for 10 ns;
  a <= to_unsigned(  2, 4 );
  wait for 20 ns;
  a <= to_unsigned(  3, 4 );
  wait for 20 ns;
end process;

process begin
  b <= to_unsigned(  7, 4 );
  wait until rising_edge(clk1);
  b <= to_unsigned(  8, 4 );
  wait until rising_edge(clk1);
  b <= to_unsigned(  9, 4 );
  wait until rising_edge(clk1);
end process;
```

```
process (clk1) begin
  clk2 <= clk1;
end process;

process begin
  c <= to_unsigned( 10, 4 );
  wait until rising_edge(clk2);
  c <= to_unsigned( 11, 4 );
  wait until rising_edge(clk2);
  c <= to_unsigned( 12, 4 );
  wait until rising_edge(clk2);
end process;

process begin
  wait until rising_edge(clk1);
  a1 <= a;
  b1 <= b;
  c1 <= c;
end process;

process begin
  wait until rising_edge(clk2);
  a2 <= a;
  b2 <= b;
  c2 <= c;
end process;
```

Answer:



- *The signal* `a` *changes in the* **same simulation cycle** *as* `clk1`
- *The signal* `clk2` *changes* **one simulation cycle later** *than* `clk1`
- *The signals triggered by* `clk2` *change* **one simulation cycle later** *than the signals triggered by* `clk1`
- *Both of the clocked processes see the* **new** *value of* `a`*. In the simulation cycle where* `clk1='1'`*,* `a` *has its new value; the same argument holds for* `clk2`*.*
- *The process triggered by* `clk1` *sees the* **old** *value of* `b`*, because* `b` *is triggered by* `clk1`*.*
- *The process triggered by* `clk2` *sees the* **new** *value of* `b`*, because* `b` *changes in the same simulation cycle as the rising edge happens on* `clk2`*.*
- *Both of the clocked processes see the* **old** *value of* `c`*, because* `c` *is triggered by* `clk2`*.*



No changes from end of 10ns + 3δ simulation cycle to 15 ns

Marking:

Example answers and marks

a1	*2*	*2*	*2*	*2*	*2*
b1	*7*	*7*	*7*	*7*	*8*
c1	*10*	*10*	*10*	*10*	*11*
a2	*2*	*2*	*2*	*2*	*2*
b2	*8*	*8*	*8*	*8*	*8*
c2	*10*	*10*	*10*	*10*	*11*
Mark	**20**	**19**	**18**	**15**	**12**

Other answers earned fewer marks.

Major criteria

- *Basic flop-to-flop connection is done correctly*
- *clk2 changes one simulation cycle after clk1*
- *Input changing at same time as clock is handled correctly*
- *clk1 to clk2 flops handled correctly*
- *clk2 to clk1 flops handled correctly*

Q2 (20 Marks) Dataflow Diagram and Control Table

(estimated time: 15 minutes)

In this question you will do register allocation and draw the control-table for the dataflow diagram shown below.

NOTES:

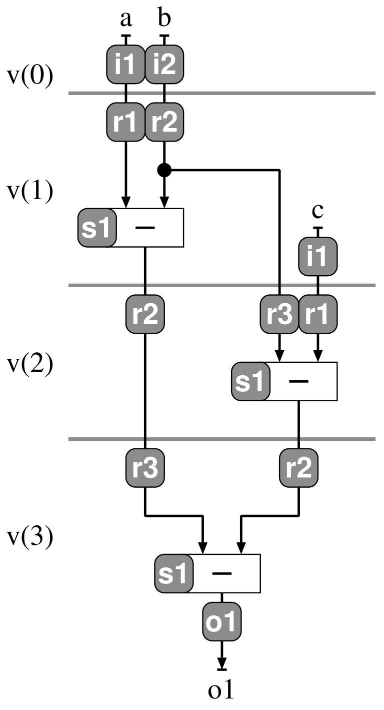
1. The optimization goal is to *minimize the total number of multiplexers and chip-enables*. More precisely, minimize the sum of the number of flip-flops that need a chip enable, and the number of 2:1 multiplexers for both registers and the subtracter. You may combine 2:1 multiplexers to create larger multiplexers. (*e.g.*, two 2:1 muxes can be combined to create a 3:1 multiplexer.)
2. You shall *not* change the allocations that have been done.

Q2a (9 Marks) Register Allocation

Allocate registers for the dataflow diagram.

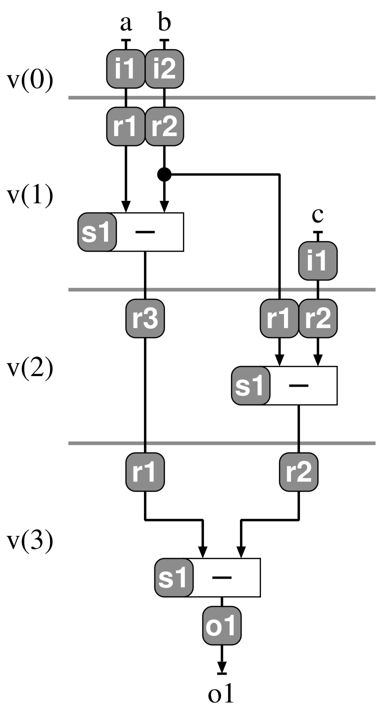
Answer:

Three sample answers.



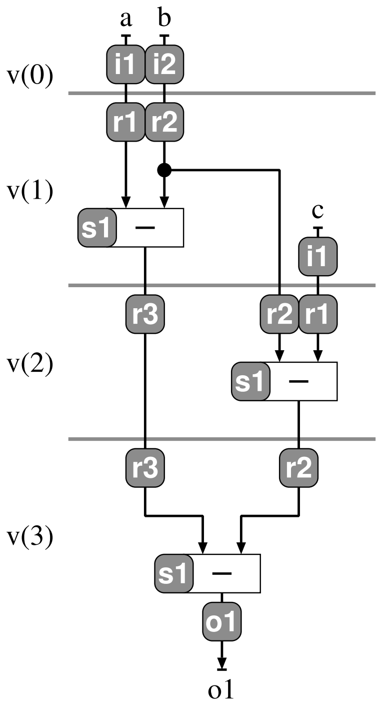
	r1		r2		r3		s1		
	ce	d	ce	d	ce	d	src1	src2	o1
idle	—	—	—	—	—	—	—	—	—
v(0)	1	i1	1	i2	—	—	×	×	×
v(1)	1	i1	1	s1	1	r2	r1	r2	—
v(2)	—	—	1	s1	1	r2	r3	r1	—
v(3)	×	×	×	×	×	×	r3	r2	s1
need ce	0		0		0				
2:1 muxes		0		1		0	1	1	0

Total cost (muxes + chip-enables) = 3



	r1		r2		r3		s1		
	ce	d	ce	d	ce	d	src1	src2	o1
idle	—	—	—	—	—	—	—	—	—
v(0)	1	i1	1	i2	—	—	×	×	×
v(1)	1	r2	1	i1	1	s1	r1	r2	—
v(2)	1	r3	1	s1	—	—	r1	r2	—
v(3)	×	×	×	×	×	×	r1	r2	s1
need ce	0		0		0				
2:1 muxes		2		2		0	0	0	0

Total cost (muxes + chip-enables) = 4



	r1		r2		r3		s1		
	ce	d	ce	d	ce	d	src1	src2	o1
idle	—	—	—	—	—	—	—	—	—
v(0)	1	i1	1	i2	—	—	×	×	×
v(1)	1	i1	0	—	1	s1	r1	r2	—
v(2)	—	—	1	s1	0	—	r2	r1	—
v(3)	×	×	×	×	×	×	r3	r2	s1
need ce	0		1		1				
2:1 muxes		0		1		0	2	1	0

Total cost (muxes + chip-enables) = 6

Marking:

9 marks *Register allocation*

Total cost	Marks
3	*9*
4	*8*
5	*7*
6	*6*

-1 mark *each missing register*

9 marks *Control table*

-2 marks *chose ASAP parcels, not indeterminate number of bubbles*

1 mark *includes output port*

1 mark unused *"×" is correct*

- **1 mark** don't-care *"–" is correct*
- **1 mark** *chip-enable is correct*
- **2 marks** *inputs to registers, subtracter, and output are correct*
- **2 mark** *state encoding (row labels) are correct*
- **1 mark** *don't do commutivity on subtraction*
- **-1 mark** *extra row*
- **-1 mark** *missing idle row*

2 marks *Analysis*

- **1 mark** *chip-enables*
- **1 mark** *multiplexers*

Q2b (9 Marks) Control Table

Draw the control table for the dataflow diagram.

NOTES:

1. Label each row and column of the table clearly.
2. The system shall support an *indeterminate number of bubbles* between valid parcels. If you do not know how to support an indeterminate number of bubbles with this system, you may earn *part marks* by writing a $\surd$ in the box at at the end of this note and answering the question for parcels that *arrive as soon as possible (ASAP)*.

   I am answering for *ASAP parcels*, not an indeterminate number of bubbles. ☐
3. Do *not* do don't-care instantiation.

Q2c (2 Marks) Analysis

For each register that you used and the subtracter, give the number of 2:1 multiplexers that it needs.

For each register that you used, use a $\surd$ in the box for "Yes" or "No" to answer whether it needs a chip-enable signal.

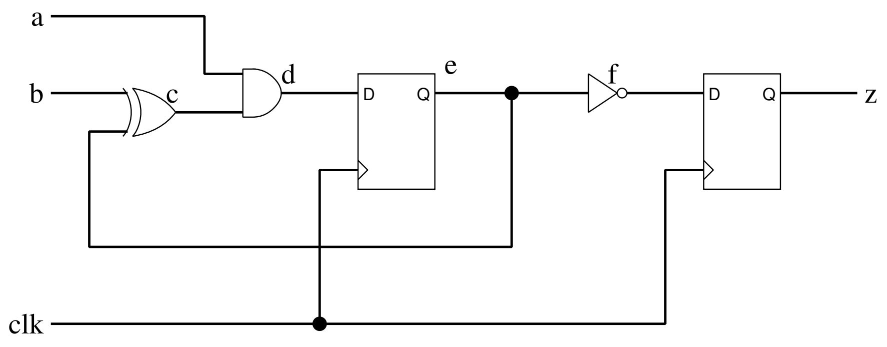
Q3 (20 Marks) The Sleet, the Snow, and the Slush

(estimated time: 15 minutes)

Each of the code fragments in Q3a–Q3c is intended to implement the schematic below, such that the behaviour of `z` with respect to the input signals `a`, `b`, and `clk` is the same in both the VHDL code and the schematic. The behaviour of internal signals does not matter.

NOTES:

1. For each of the code fragments Q3a–Q3c:
   (a) Answer whether the code is *legal*
   (b) If the code is *illegal*: explain why, and proceed to the next code fragment.
   (c) Answer whether the output `z` is *correct* with respect to the schematic.
   (d) If the code is *incorrect*: explain why and continue by answering whether the code is synthesizable.
   (e) Answer whether the code is *synthesizable*.
   (f) If the code is *unsynthesizable*: explain why, and proceed to the next code fragment.
   (g) List the signals in the code that are flip-flops.
   (h) Answer whether the code adheres to good coding practices, according to the guidelines for ECE 327.
   (i) If the code does *not follow good coding practices*: explain why.
2. All signals are `std_logic`.
3. When listing the flip-flops, list the signals in *alphabetical order*.



Specification schematic

Q3a

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

process (c, d ) begin
  if d = '0' then
    e <= '0';
  else
    e <= c;
  end if;
end process;

d <= b xor e;

process begin
  wait until rising_edge(clk);
  z <= not e;
end process;
```

Answer:
Legal, incorrect, synth, flops=[c,z], bad (combinational loop)

Q3b

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

e <= d xor g;
f <= c and e;

process begin
  wait until rising_edge(clk);
  g <= f;
end process;

z <= not g;
```

Answer:
Legal, correct, synth, flops=[c,d,g], good

Q3c

```
process (clk) begin
  if rising_edge(clk) then
    if a = '0' then
      e <= '0';
    else
      e <= a xor e;
    end if;
  end if;
  z <= not e;
end process;
```

Answer:
legal, incorrect (z is a latch), unsynth (registered and latched/combinational assignment in same process)

Marking:

Solution and maximum possible marks

	a		**b**		*c*			
Legal	*Y*	*1.5*	*Y*	*1.5*	*Y*	*1.5*	*Y*	*1.5*
Correct	*N*	*1.5*	*Y*	*1.5*	*N*	*1.5*	*N*	*1.5*
Synth	*Y*	*1.5*	*Y*	*1.5*	*N*	*1.5*	*Y*	*0.0*
Flops		*1.5*		*1.5*		*0.5*		*0.5*
Good	*N*	*1.5*	*Y*	*1.5*			*N*	*0.5*
Total		*7.5*		*7.5*		*5.0*		*5.0*

Yes/No and marks for explanation

		Student answer	
		Yes	*No*
Sol	*Yes*	*1.5*	*0.0 + 0.5*
	No	*0.0*	*0.5 + 1.0*

When the student answers "No", an explanation is required. The marking rules for the Yes/No part and the explanation are written as: $x+y$ where x is the mark for "Yes/No" and y is the maximum possible mark for the explanation.

Q4 (20 Marks) State Machines

(estimated time: 15 minutes)

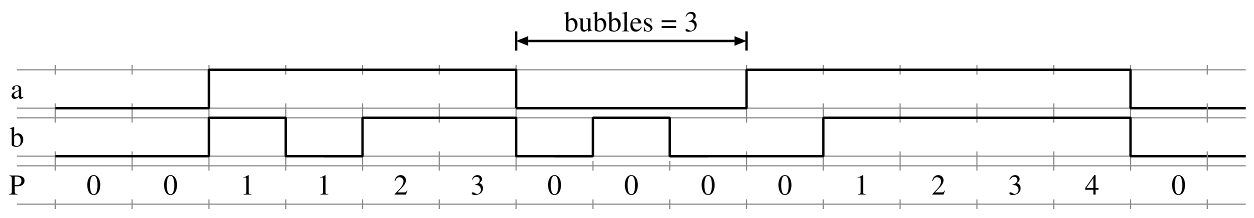
Each of the state machines in Q4a–Q4c is intended to count the number of times that `b='1'` while `a='1'`. The behaviour is described precisely in the table below, which defines the action to perform on `P` in terms of conditions on the inputs `a` and `b`:

Input conditions		
`a`	`b`	**Action**
'0'	–	set `P` to 0
'1'	'0'	no change to `P`
'1'	'1'	increment `P`

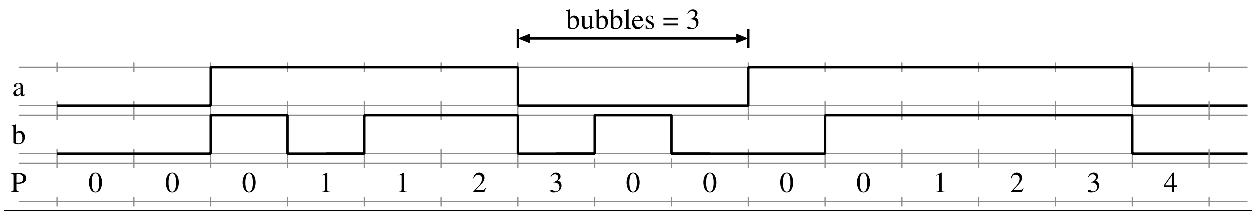
NOTES:

1. The *latency* between the inputs `a` and `b` and the output `P` may be *zero or more* clock cycles. The waveforms below illustrate the behaviour with a latency of 0 and 1.
2. When `a='0'`, the state machine may require multiple clock cycles of `a='0'` before the next clock cycle in which `a='1'`. These clock cycles of `a='0'` are called *bubbles*.
3. For each of the state machines Q4a–Q4c, answer whether the state machine is *correct* with respect to the intended behaviour.
4. If the state machine is *correct*, answer what is the *minimum number of bubbles* that the state machine requires in order to work correctly.
5. If the state machine is *incorrect*, explain either how the machine's behaviour differs from the intended behaviour or how the state machine could be modified to fix the incorrect behaviour.
6. The input `a` is guaranteed to be `'0'` for first four clock cycles.

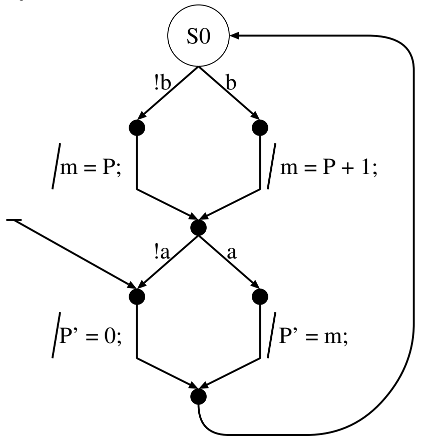
Latency = 0: increment P in same clock cycle as b='1':



Latency = 1: increment P in next clock cycle after b='1':

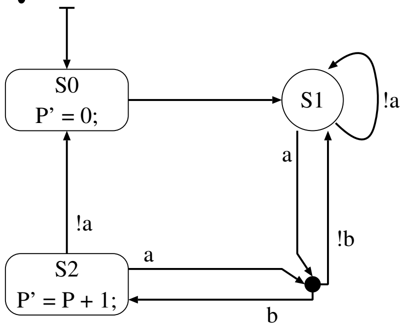


Q4a



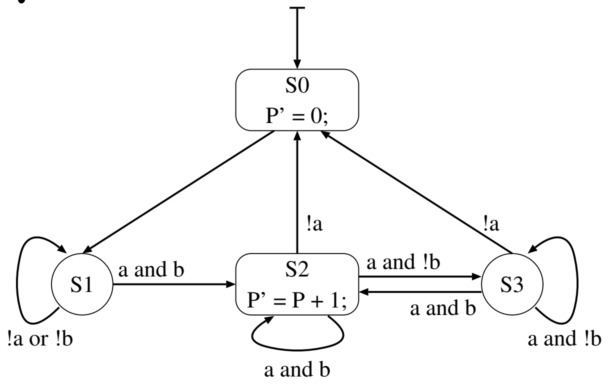
Answer:
Correct. Bubbles=1.

Q4b



Answer:
Incorrect. If a drops to '0' when the system is in S1, the system will not set P to 0. This is a bug **if-and-only-if** *the system has already incremented P. If said "correct", the answer for number of bubbles is 2.*

Q4c



Answer:
Correct. Bubbles = 2.

Marking:

Q4a,c *(correct answer is "correct")*

- **student answered "correct"**
  - **6.5 marks** *bubbles correct*
  - **5 marks** *bubbles off by 1*
  - **4 marks** *bubbles off by 2*
  - **3 marks** *bubbles off by 3 or more*
  - **2 marks** *no bubbles answer*
- **student answered "buggy"**
  - **4–5 marks** *description of bug is mostly correct*
  - **2–4 marks** *some correct information*
  - **0.5 marks** *no explanation*

Q4b *(correct answer is "buggy")*

- **student answered "correct"**
  - **4 marks** *bubbles correct*
  - **3 marks** *bubbles off by 1*
  - **2 marks** *bubbles off by 2 or more*
  - **0.5 marks** *no explanation*
- **student answered "buggy"**
  - **7 marks** *correct and complete*
  - **5 marks** *mostly correct*
  - **3 marks** *some correct info*
  - **2 marks** *no explanation*

Common problems

- *confusion on comb vs reg asn*
- *confusion on transient state vs normal state*
- *confusion on when reg asn are visible*
- *forget that need to write to variable before read*

Q5 (20 Marks) Dataflow Diagram with Memory

(estimated time: 15 minutes)

Draw a dataflow diagram that implements the specification:

```
M[a] = b;
M[c] = d;
z    = M[c] + M[c+1];
```

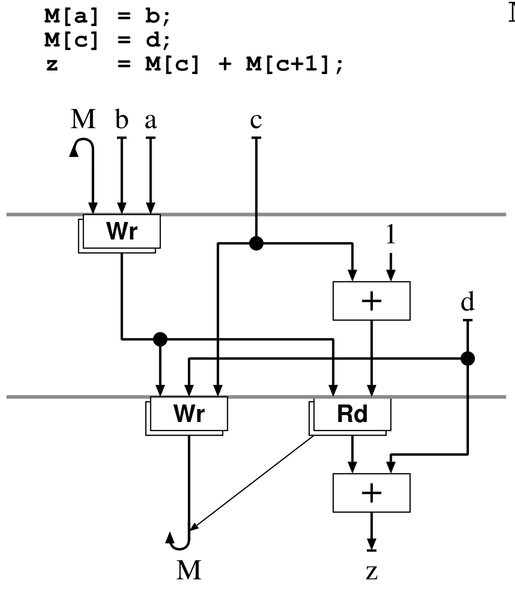
NOTES:

1. It is guaranteed that `a` $\neq$ `c`.
2. Inputs shall be *registered*
3. Outputs may be *either combinational or registered*
4. Memory has registered inputs and combinational outputs (same as in class)
5. The memory shall be *dual-ported.* If you do not know how to answer the question with dual-ported memory, you may earn *part marks* by writing a $\surd$ in the box at at the end of this note and answering the question for *single-ported memory.*

   I am answering for *single-ported memory*, not dual-ported memory. ☐
6. Optimization goals in order of decreasing importance:
   (a) minimize *latency*
   (b) minimize *area*
      i. input ports
      ii. registers (*excluding* memory)
      iii. adders
      iv. output ports
   (c) minimize *clock period*
7. Input values may be read in any clock cycle, but each input value shall be read exactly once.
8. Optimizations to the pseudocode are allowed, so long as the final values on `z` and `M` are correct.

Q5a (15 Marks) Dataflow Diagram

Answer:



Marking:

2 marks	*functional correctness*
2 marks	*DFD syntax is correct*
2 marks	*optimal latency*
1 mark	*DFD uses M*
1 mark	*Wr produces M*
1 mark	*anti-dependency arrow*
1 mark	*mem operations on clock cycle boundaries*
1 mark	*registered inputs*
1 mark	*combinational outputs*
1 mark	*1 register*
1 mark	*3 inputs*
1 mark	*1 adder unit*

Q5b (2 Marks) Bubbles

What is the minimum number of bubbles between valid parcels for your dataflow diagram?

Answer:
1 bubble

Q5c (3 Marks) Parcel Schedule

Should this system require that parcels arrive as soon as posible (ASAP parcels) or should the system support an indeterminate number of bubbles between parcels? **For full marks, you must justify your answer.**

Answer:
The system should support an indeterminate number of bubbles. `M` *is an inter-parcel variable. The best state-encoding for a system with an inter-parcel variable is valid-bits, because we need to support the initial clock cycles when the first parcel enters the empty system. Once we have valid-bits, we automatically support an indeterminate number of bubbles.*

Marking:

1 mark	*inter-parcel variable*
1 mark	*with inter-parcel variables, valid-bit is best encoding*
1 mark	*with valid-bits, support for indeterminate number of bubbles is free*