ECE 327 Solution to Midterm

2014t1 (Winter)

All requests for re-marks must be submitted in writing to Mark Aagaard before 8:30am on Friday March 7.

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

Q1 (20 Marks) Simulation

(*estimated time: 15 minutes*)

For the code fragment on the next page, *briefly* describe what happens in each *simulation cycle*, beginning at 3 ns.

NOTES:

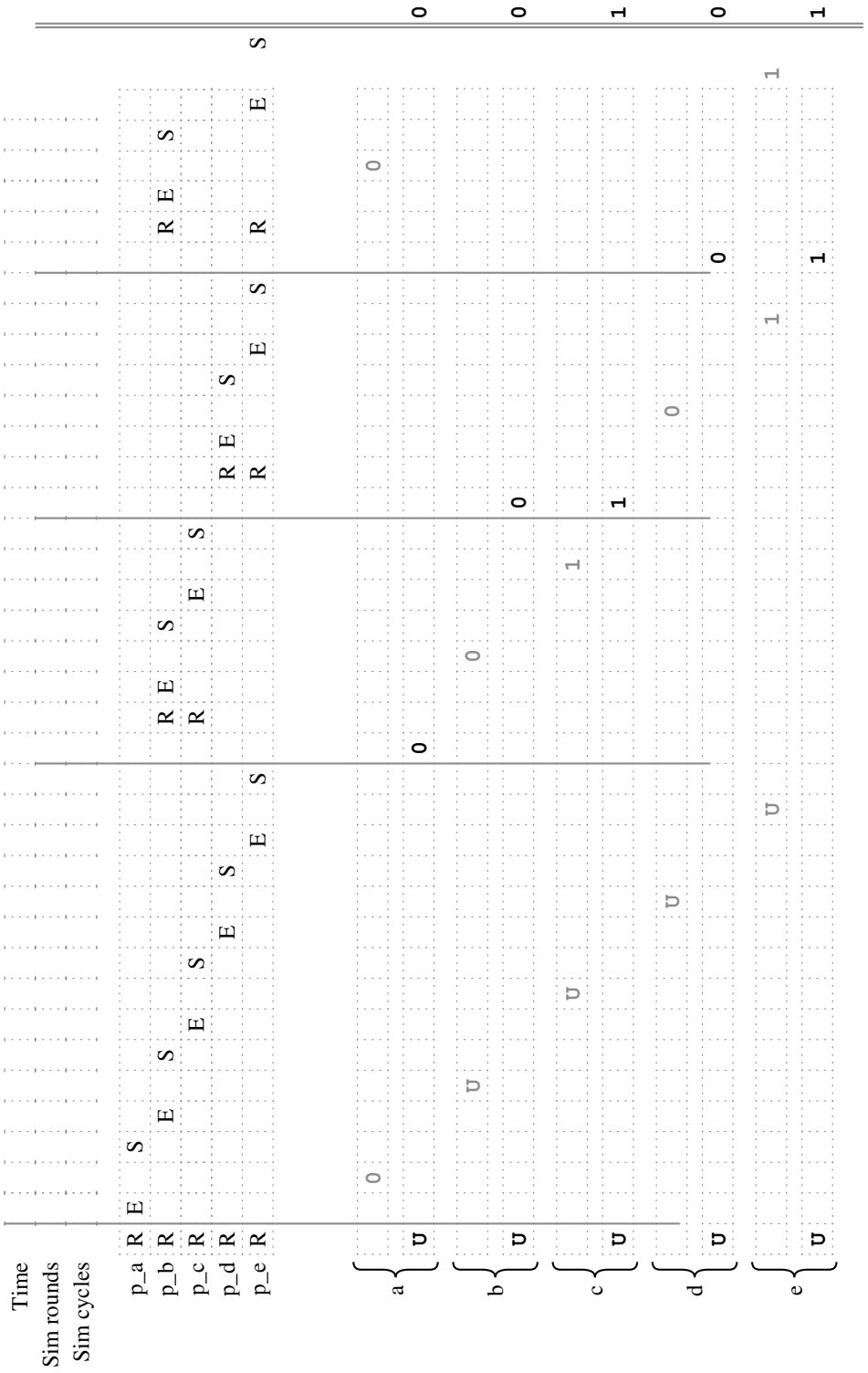
1. End your simulation after seven simulation cycles or just before 10 ns, whichever occurs first.
2. If you need less than seven simulation cycles, then write "N/A" in each simulation cycle that you do not need.
3. If, after seven simulation cycles, the simulation has not reached 10 ns, then write "$\surd$" in the box for *ran out of simulation cycles*.
4. All signals are `std_logic`.
5. Just before 3 ns, the signals have the values shown in the waveform for scratch work on the next page.

```
p_a : process begin
  a <= '0';
  wait for 3 ns;
  a <= '1';
  wait for 7 ns;
  a <= '0';
  wait;
end process;

p_b : process (a, e) begin
  b <= a and e;
end process;

p_c : process (a) begin
  c <= not a;
end process;

p_d : process (c) begin
  d <= not c;
end process;

p_e : process (b, d) begin
  e <= not( b ) or d;
end process;
```

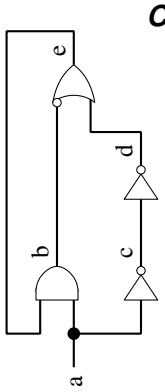
Answer:

Beginning of simulation

This is not part of the required answer. It is included just for completeness.

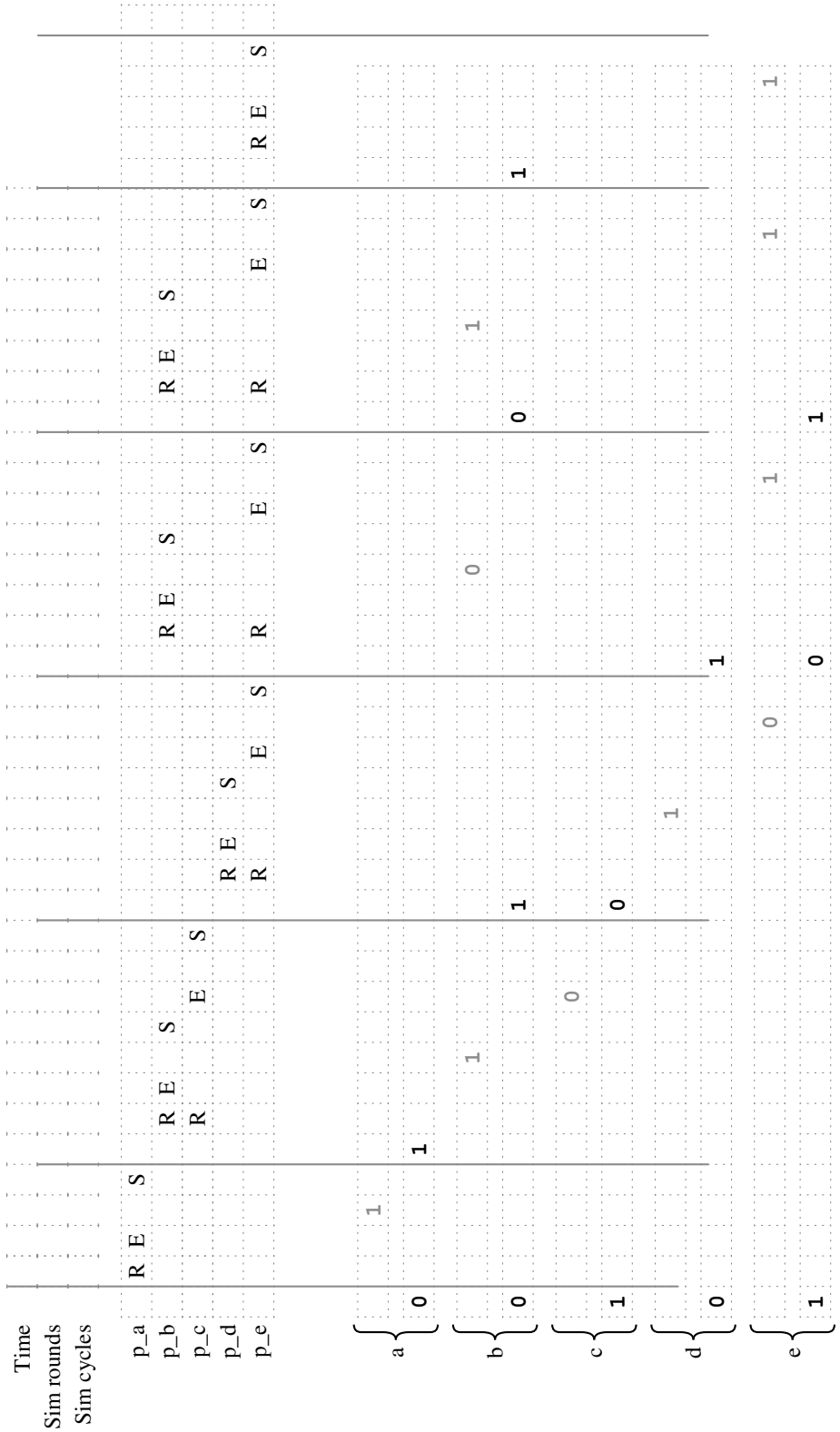


Circuit



Simulation of 3ns

The waveform is not part of the required answer. It is included just for completeness.



	visible values	*resumed processes*	*projected values*
1		*p_a*	*a=1*
2	*a=1*	*p_b, p_c*	*b=1, c=0*
3	*b=1, c=0*	*p_d, p_e*	*d=1, e=0*
4	*d=1, e=0*	*p_b, p_e*	*b=0, e=1*
5	*b=0, e=1*	*p_b, p_a*	*b=1, e=1*
6	*b=1*	*p_e*	*e=1*

Marking:

+3 marks *functional corrrectness*
+3 marks *multiple processes resume/execute per sim cycle*
+3 marks *p_a resumes correctly on time change*
+3 marks *processes resume correctly on signal change*
+3 marks *projected assignment visible in next sim cycle*
+3 marks *sim round ends when no more changes*
+2 marks *if no change on value, then no visible assignment*
-2 marks *simulation cycle for time advancing*
-2 marks *time increments to 7ns, rather than 10ns*

Q2 (17 Marks) The Gold, the Silver, and the Bronze

(*estimated time: 10 minutes*)

For each of the code fragments Q2a–Q2b:

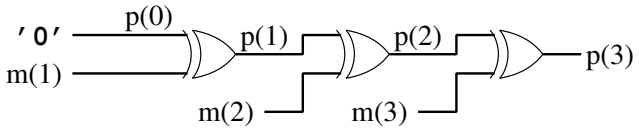
1. Answer whether the code is *legal*
2. If the code is *illegal*: explain why, and proceed to the next code fragment.
3. Answer whether the code is *synthesizable*.
4. If the code is *unsynthesizable*: explain why, and proceed to the next code fragment.
5. Answer whether the code adheres to good coding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

NOTES:

1. If the code is synthesizable: draw the circuit that would most likely result from synthesizing the code.
2. If the VHDL code includes an implicit state machine: draw the gates, wires, and flops for the datapath. All of the arithmetic and logical operators in the VHDL code (*e.g.*, "+", "−", "<", and "`xor`") are considered part of the datapath.
3. You may draw the control portion of the circuit as a cloud or black-box that drives the appropriate signals in the datapath.

Answer:

Legal, synth, good.



Q2a

```
signal m : std_logic( 0 to 3 );
signal p : std_logic( 1 to 3 );
...
process (m, p)
begin
  p(0) <= '0';
  for i in 1 to 3 loop
    p(i) <= p(i-1) xor m(i);
  end loop;
end process;
```

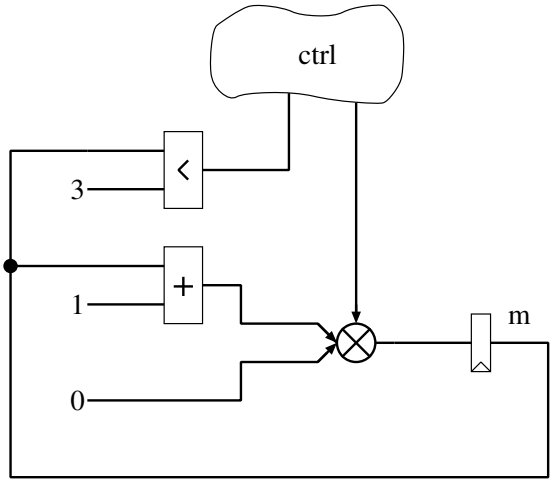
There were two typos in the signal declarations. The correct declarations are:

```
m :  std_logic_vector( 1 to 3 )
p :  std_logic_vector( 0 to 3 )
```

Taking these typos into account, full marks were given for an answer of "Illegal" that identified the problems of the incorrect type or the incorrect bounds on p.

Answer:

Legal, synth, good.



Q2b

```
signal m : unsigned( 7 downto 0 );
...
process begin
  wait until rising_edge(clk);
  m <= to_unsigned( 0, 8 );
  wait until rising_edge(clk);
  while m < 3 loop
    m <= m + 1;
    wait until rising_edge(clk);
  end loop;
end process;
```

Marking:

+1 mark *free, if all of both Q2a and Q2b are answered*
+8 marks max *Legal, synth, good*
- **+4 marks** *baseline*
- **+4 marks** *circuit*

+6 marks max *Legal, synth, bad*
- **+1 marks** *baseline*
- **+2 marks** *justification*
- **+3 marks** *circuit*

+3 marks max *Legal, unsynth*
- **+1 marks** *baseline*
- **+2 marks** *justification*

+2 marks max *Illegal*
- **+2 marks** *justification*

circuit drawing A
- **+2 marks** *3* XOR *gates*
- **+1 mark** *'0' and m() are inputs*
- **+1 mark** *chain of p() connections*
- **-1 mark** *any mistake not listed above*

circuit drawing B
- **+1 marks** *comparator, adder, mux*
- **+1 mark** *output of comparator is input ctrl*
- **+1 mark** *output of ctrl is sel of mux*
- **+1 mark** *m is a register*
- **-1 mark** *any mistake not listed above*

Q3 (18 Marks) State Machine Behaviour

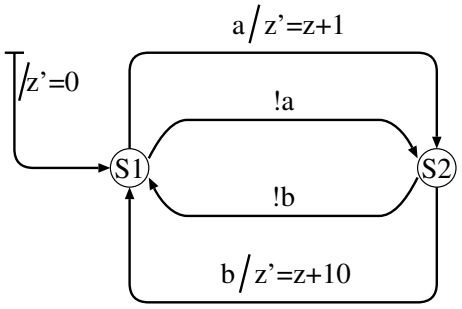
(*estimated time: 10 minutes*)

In this question, you will compare the behaviour of z in the state machines Q3a–Q3b against z in the specification machine.

NOTES:

1. A state machine is *correct* if its z has the *same sequence* of values as z in the specification machine.
2. Ignore any differences in the first few clock cycles.
3. The value on z does *not* need to appear at the same time as in the specification machine. Correctness is determined only by the sequence of values; a *time-shift* between the state machine and the specification machine is allowed. (Illustrations of "time-shift" are given below.)
4. If the state machine is *correct*, answer what the *time-shift* is from the specification machine.
5. If the state machine is *incorrect*, explain either how the machines behaviour differs from the specification machine or how the state machine could be modified to fix the incorrect behaviour.

Specification machine



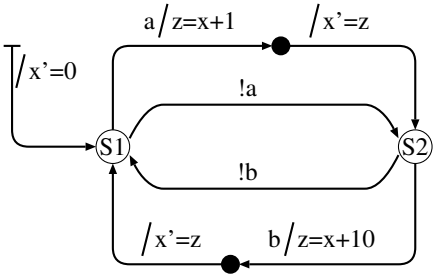
Time-shift examples

(waveform values are just examples, they do not match the actual values of z in the specification machine)

specification z	1	3	6	7	8	9
time-shift = 1	• • • •	1	3	6	7	8

specification z	1	3	6	7	8	9
time-shift = -2	6	7	8	9	• • • • • • • •	

Q3a



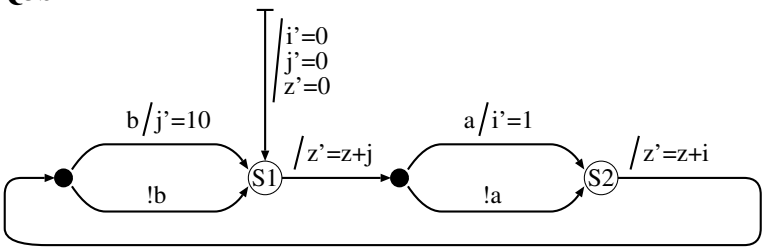
Answer:

Incorrect, z must be assigned a value in each clock cycle, because it is combinational

Marking:

9 marks *Incorrect, z must be assigned a value on the "!a" and "!b" edges*
6 marks *Incorrect, because z is combinational*
4 marks *Incorrect, most of justification is correct.*
3 marks *Incorrect, significant fraction of justification is correct.*
9 marks *Correct and time-shift=-1*
7 marks *Correct and time-shift=0*
6 marks *Correct and time-shift=1*
4 marks *Correct and time-shift=2*

Q3b



Answer:

Incorrect, should add j'=0 and i'=0 to !b and !a edges.

Marking:

9 marks *Incorrect, because do not reset j and i*

7 marks *Incorrect, identified that problem is with i and j, but did not give precise description*

6 marks *Correct and time-shift=1*

5 marks *Correct and time-shift=2*

6 marks *Incorrect, most of justification is correct*

5 marks *Incorrect, justification related to i and j*

4 marks *Incorrect, justification is technically correct, but not related to buggy behaviour*

3 marks *Incorrect, justification is mostly incorrect or difficult to understand*

Q4 (15 Marks) Allocation and Control

(*estimated time: 15 minutes*)

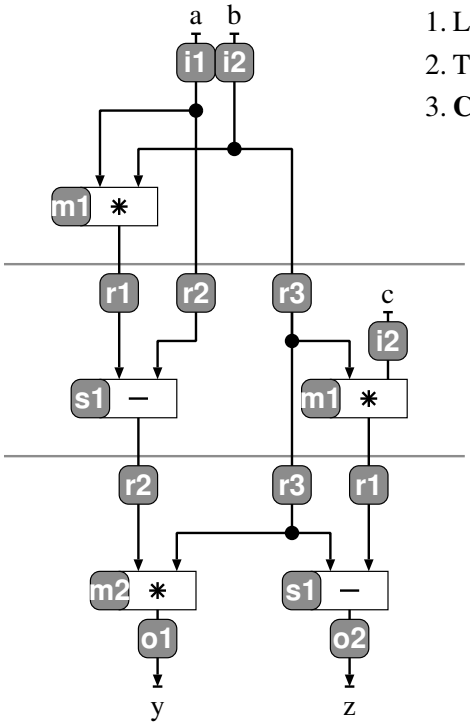
For the dataflow diagram below: perform input/output allocation, datapath allocation, and register allocation; then draw the control table.

NOTES:

1. You may use 2:1 multiplexers, and may combine 2:1 multiplexers to create larger multiplexers.
2. The optimization goals, in order of highest priority to lowest, are to minimize the number of:
   (a) multipliers and subtracters
   (b) input and output ports
   (c) registers
   (d) 2:1 multiplexers on *registers* (including the 2:1 multiplexers used to build larger multiplexers)
   (e) 2:1 multiplexers on *multipliers and subtracters* (including the 2:1 multiplexers used to build larger multiplexers)
   (f) chip-enables
3. You may *not* perform scheduling optimizations on the dataflow diagram.
4. The *only* algebraic optimization that you may perform is *commutativity*.

Q4a (7 Marks) Allocation

Answer:



Q4b (8 Marks) Control Table and Multiplexer Count

Draw the control table for the dataflow diagram and calculate the number of multiplexers.

NOTES:

1. Label each row and column of the table clearly.
2. The system shall support an ASAP parcel schedule between parcels.
3. **Clearly show how you calculated the number of multiplexers.**

Marking:

+1 mark *inputs and outputs allocated*
+1 mark *registers allocated*
+1 mark *multipliers allocated*
+1 mark *subtracters allocated*
+3 marks *multiplexers*
- **3 marks** *1 reg mux + 3 datapath muxes = 4 muxes total*
- **2 marks** *2 reg muxes + 2 datapath muxes = 4 muxes total*
- **1 marks** *2 reg mux + 3 datapath muxes = 5 muxes total*

-2 marks *applied commutativity to subtraction*
-2 marks *used m1 in both cycle 0 and cycle 2*
-2 marks *more than 2 inputs, 3 registers, 2 multipliers or 1 subtracter*
-1 marks *mistake not listed above*
-2 marks *2–3 mistakes not listed above*

Answer:

Control table using clock cycles:

	r1		**r2**		**r3**		**m1**		**m2**		**s1**		**o1**	**o2**
	ce	*d*	*ce*	*d*	*ce*	*d*	*src1*	*src2*	*src1*	*src2*	*src1*	*src2*		
0	*1*	*m1*	*1*	*i1*	*1*	*i2*	*i1*	*i2*	*X*	*X*	*X*	*X*	*X*	*X*
1	*1*	*m1*	*1*	*s1*	*0*	*–*	*r3*	*i2*	*–*	*–*	*r1*	*r2*	*–*	*–*
2	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*X*	*r2*	*r3*	*r3*	*r1*	*m2*	*s1*
muxes		*0*		*1*		*0*	*1*	*0*	*0*	*0*	*1*	*1*	*0*	*0*

Because have ASAP schedule, combine first and last rows, label by states:

	r1		**r2**		**r3**		**m1**		**m2**		**s1**		**o1**	**o2**
	ce	*d*	*ce*	*d*	*ce*	*d*	*src1*	*src2*	*src1*	*src2*	*src1*	*src2*		
S0	*1*	*m1*	*1*	*i1*	*1*	*i2*	*i1*	*i2*	*r2*	*r3*	*r3*	*r1*	*m2*	*s1*
S1	*1*	*m1*	*1*	*s1*	*0*	*–*	*r3*	*i2*	*–*	*–*	*r1*	*r2*	*–*	*–*
muxes		*0*		*1*		*0*	*1*	*0*	*0*	*0*	*1*	*1*	*0*	*0*

Marking:

+2 marks *two rows in table, labeled by states*
+2 marks *don't care values*
+2 marks *chip enable*
+1 mark *included o1 and o2*
+1 mark *correct columns for regs, muls, sub*
-1 mark *included some "unused" cells*
-1 mark *1–2 mistakes not listed above*
-2 marks *more than 2 mistakes*

Q5 (10 Marks) Bubbles and Machines

(*estimated time: 7 minutes*)

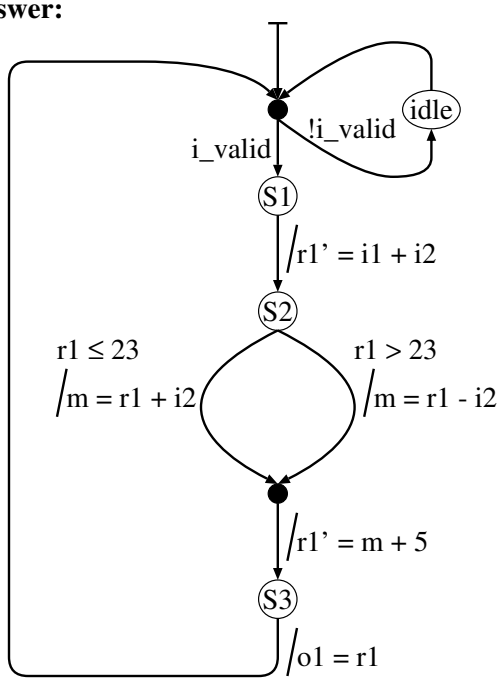
Draw the state-machine diagram for the pseudocode specification below, where the "#---" comments specify the clock-cycle boundaries.

NOTES:

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

```
#------------------
r1 = i1 + i2;
#------------------
if r1 > 23 then {
  m = r1 - i2;
} else {
  m = r1 + i2;
}
r1 <= m + 5;
#------------------
o1 <= r1;
```

Answer:



Marking:

- **+1 mark** *initial edge*
- **+2 marks** *3 main states*
- **+1 mark** *idle state with loop*
- **+1 mark** *transient state at join-point for idle-loop*
- **+1 mark** *registered assignments to r1*
- **+1 mark** *conditional test on value of r1*
- **+1 mark** *combinational assignments to m*
- **+1 mark** *transient state after assignments to m*
- **+1 mark** *loop back from bottom to transient state at idle-loop*
- **-1 mark** *1 mistake not listed above*
- **-2 marks** *2–3 mistakes not listed above*

Q6 (20 Marks) Design with Memory

(*estimated time: 15 minutes*)

Draw a dataflow diagram that implements the pseudocode specification:

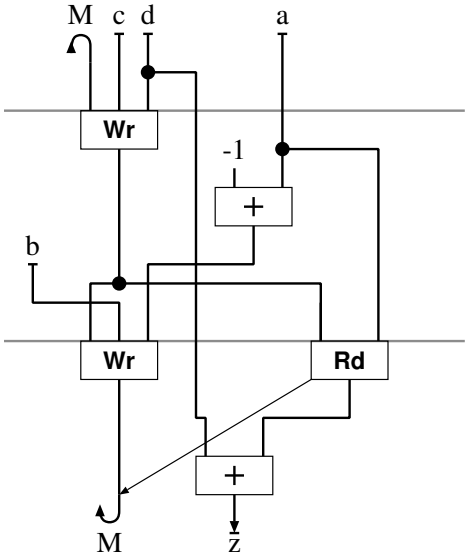
```
M[a-1] = b;
M[c]   = d;
z      = M[a] + M[c];
```

NOTES:

1. It is guaranteed that `a` $<$ `c`.
2. Inputs shall be *registered*
3. Outputs may be *either combinational or registered*
4. The system shall support an *indeterminate number of bubbles*
5. Memory has registered inputs and combinational outputs (same as in class)
6. 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 the end of this note and answering the question for *single-ported memory*.

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

Answer:



The read operation could be moved up one clock cycle without any penalty.

Marking:

+3 marks *functional correctness*
+2 marks *optimal latency*
+2 marks *DFD syntax is correct*
+2 marks *mem operations on clock cycle boundaries*
+1 mark *DFD uses M*
+1 mark *M is an inter-parcel variable*
+1 mark *use dual-port memory*
+1 mark *M has one write port and one read port*
+1 mark *Wr produces M*
+1 mark *anti-dependency arrow*
+1 mark *registered inputs*
+1 mark *combinational outputs*
+1 mark *2 registers*
+1 mark *3 inputs*
+1 mark *1 adder unit (a – 1 == a + -1)*