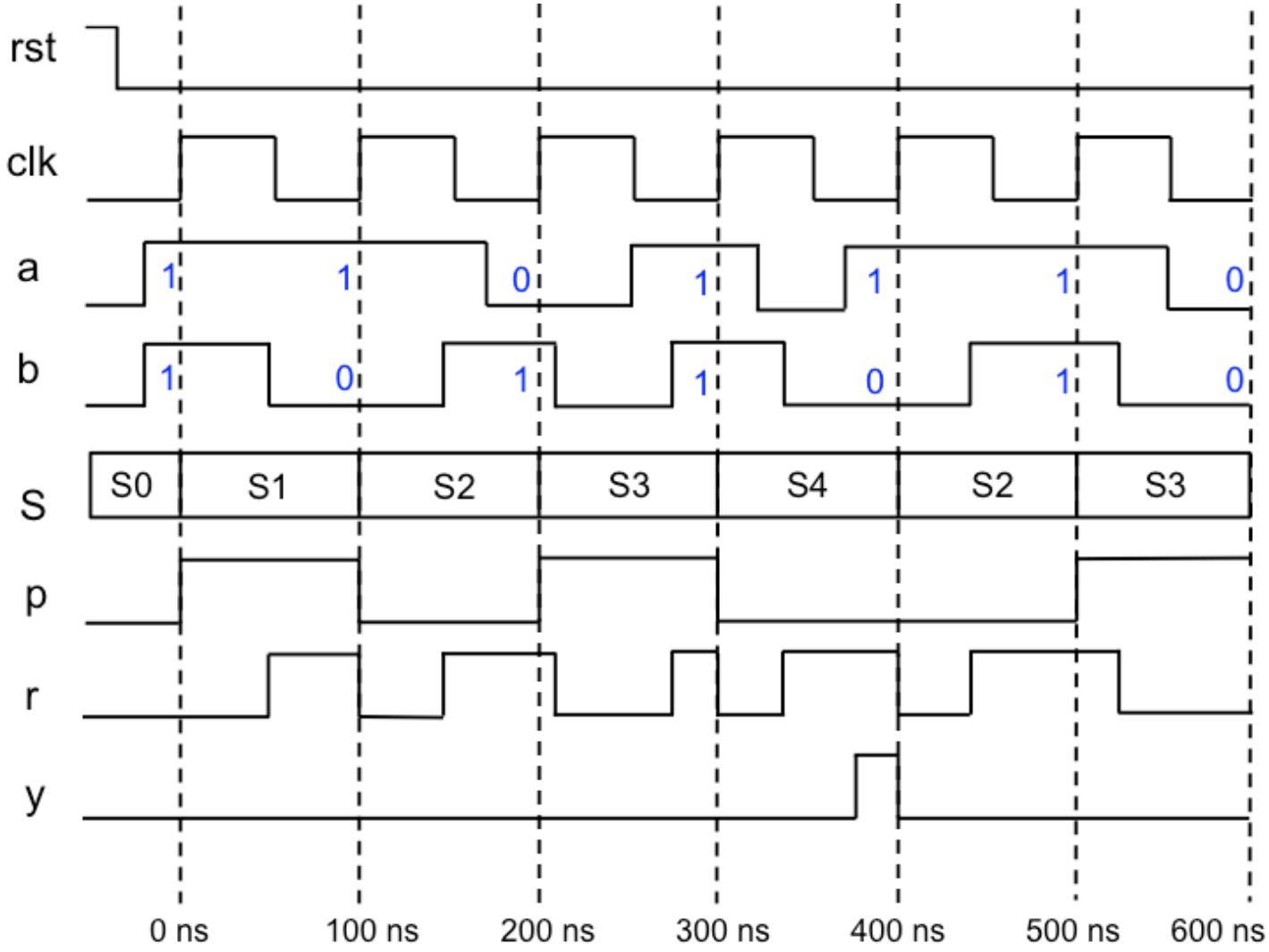
ECE 448

Midterm Exam

Solutions

March 4, 2013

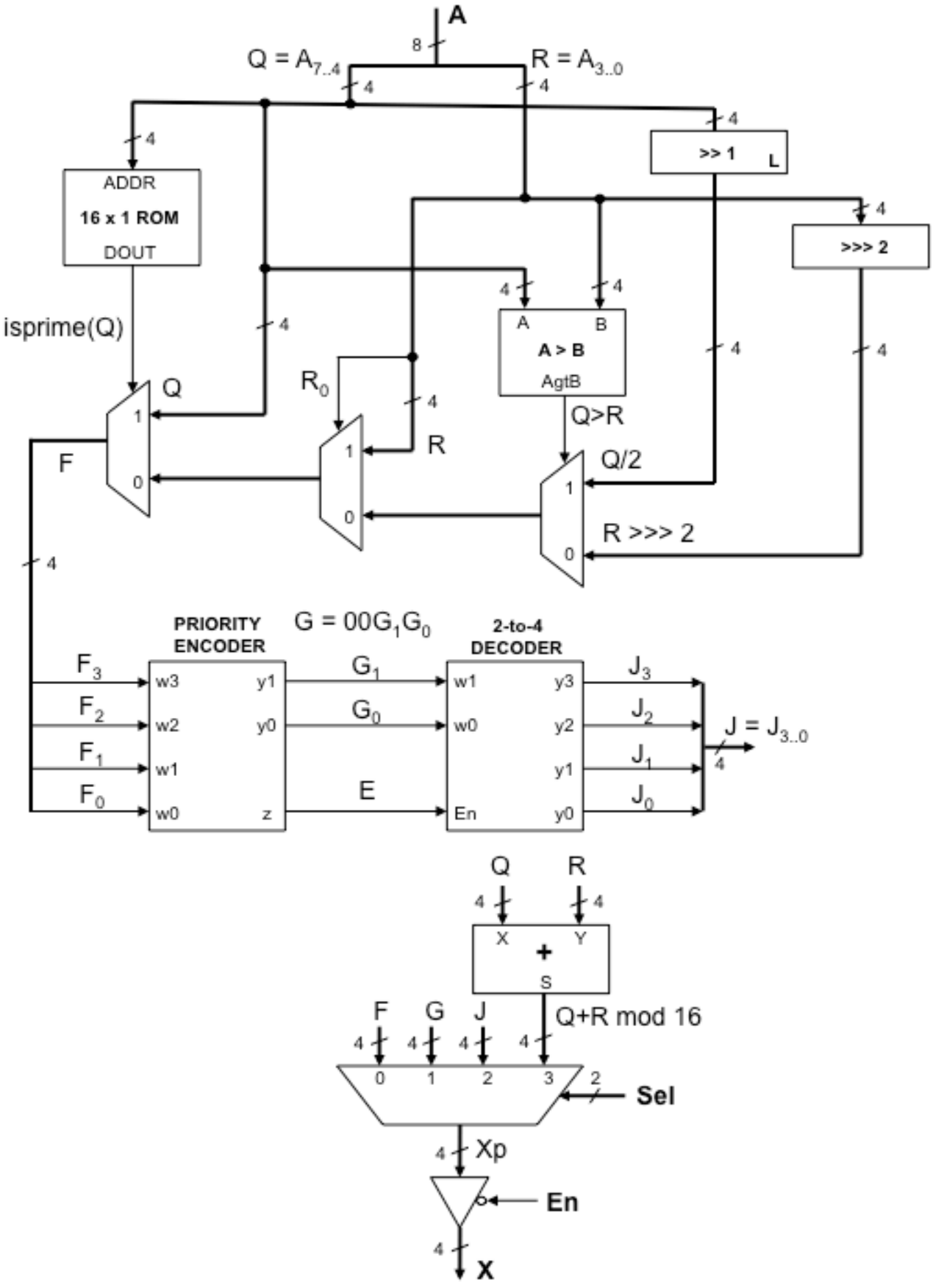
Problem 1 - ASM chart



Dataflow VHDL code for the output function computing p, r, and y, as a function of the state S, and the inputs a and b:

```
p <= '1' when (S = S1) or (S = S3) else '0';

r <= '1' when ((S = S1) and (b = '0')) or ((S = S2) and (b = '1')) or ((S = S3) and (b = '1')) or
                ((S = S4) and (b = '0')) else '0';

y <= '1' when (S = S4) and (b = '0') and (a = '1') else '0';
```

Problem 2 – Block Diagram of Combinational Logic



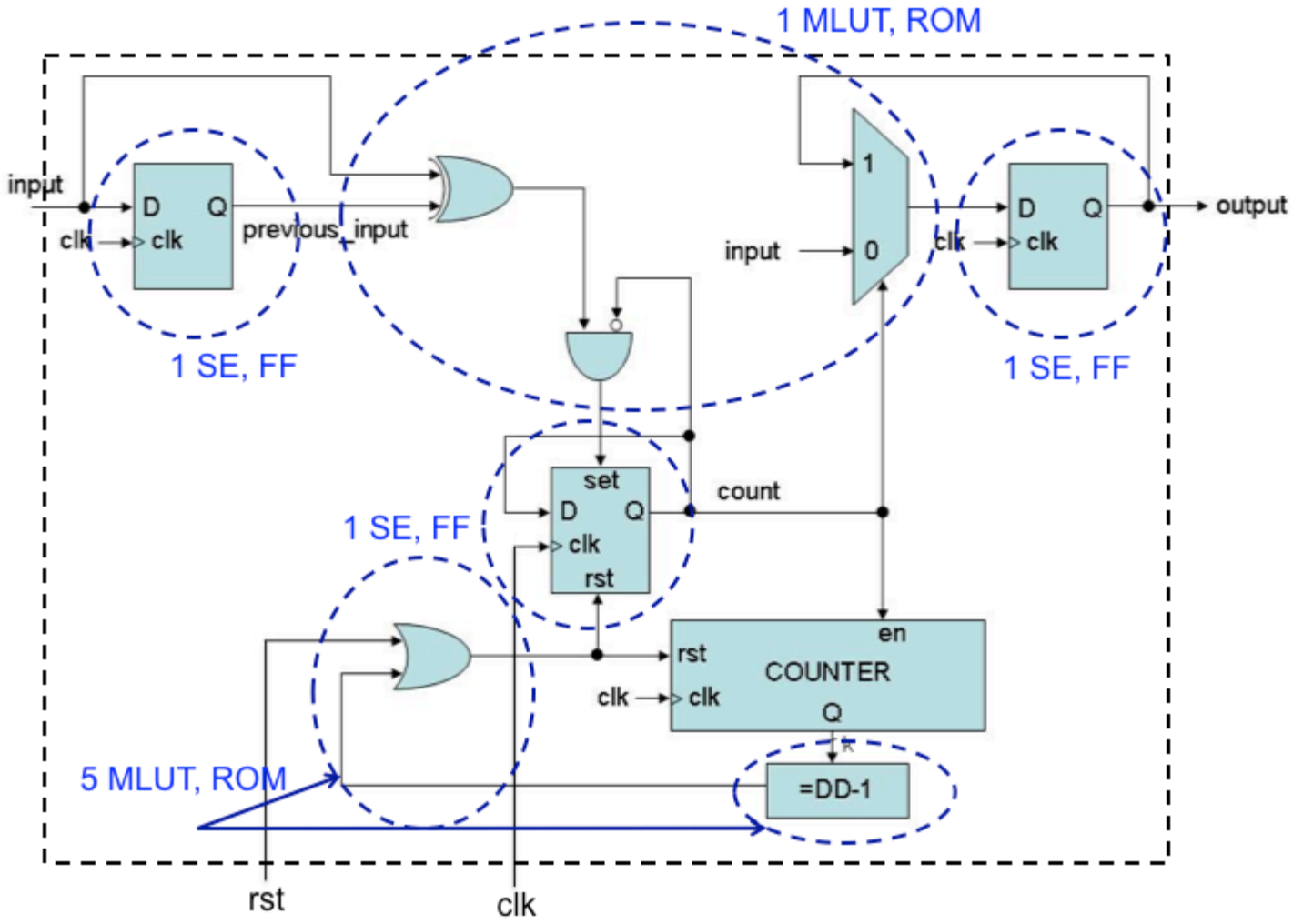
16 x 1 ROM
Memory Map

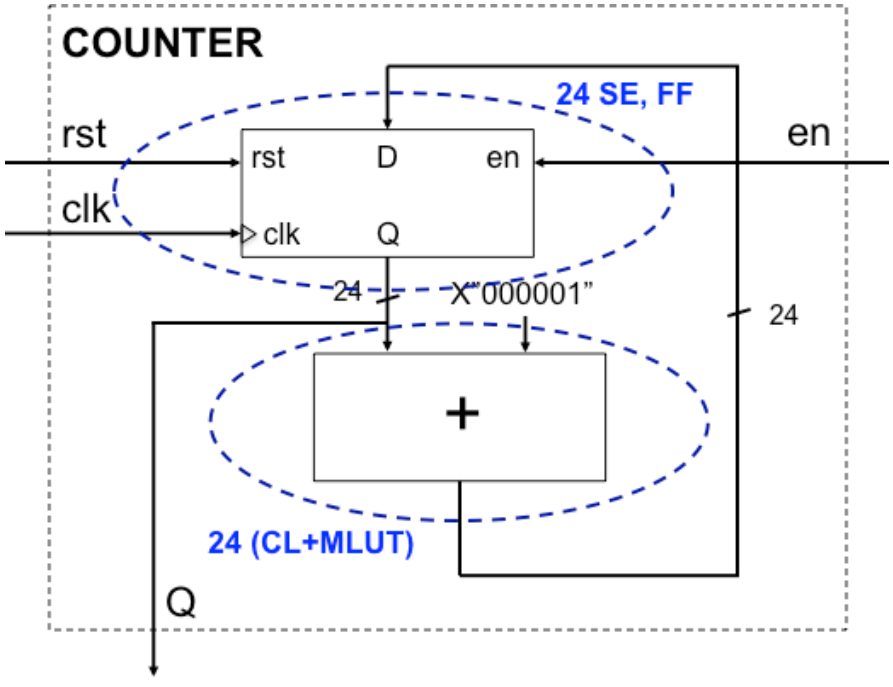
ADDR	
0	0
1	0
2	1
3	1
4	0
5	1
6	0
7	1
8	0
9	0
A	0
B	1
C	0
D	1
E	0
F	0

Problem 3 – Mixed Code of Sequential Logic

```
library ieee;
use ieee.std_logic_1164.all;

entity misr is
        generic (C : std_logic_vector(7 downto 0) );
    port (
        -- inputs
        clk     :       in      std_logic;
        rst     :       in      std_logic;
        en      :       in      std_logic;
        D       :       in      std_logic_vector (7 downto 0);

        -- outputs
        Q_out:          out     std_logic_vector (7 downto 0)
       );
end misr;

architecture mixed of misr is

-- intermediate signals
  signal  Q : std_logic_vector (7 downto 0);
  signal  Q0_replicated : std_logic_vector (7 downto 0);
  signal  d_ff_in : std_logic_vector (7 downto 0);

begin

        Q0_replicated <= (others => Q(0));
        d_ff_in <= D xor ('0' & Q(7 downto 1)) xor (C and Q0_replicated);


-- D flip flop operation
        D_FFs: process (rst, clk)
        begin
                if (rst = '1') then
                        Q <= (others => '0');
                elsif rising_edge(clk) then
                        if(en = '1') then
                                Q <= d_ff_in;
                        end if;
                end if;
        end process;

        Q_out <= Q;
end mixed;
```

Problem 4 – FPGA Resources





Problem 5 – Simple Testbench

```
library ieee;
use ieee.std_logic_1164.all;

entity debouncer_tb is
end debouncer_tb;

architecture behavioral of debouncer_tb is

   -- inputs
   signal clk : std_logic := '0';
   signal rst : std_logic;
   signal input: std_logic;

   -- outputs
   signal output : std_logic;

   -- constant definitions
   constant clk_period : time := 10 ns;
   constant rst_length : time := 50 ns;
   constant min_before_pulse : time := 100 ns;
   constant pulse_width :  time := 500 ns;
   constant bounce_period : time := 40 ns;

begin

        debouncer_inst: entity work.debouncer
                        generic map (
                                K => 4,
                                DD => 15)
                        port map (
                                clk => clk,
                                rst => rst,
                                input => input,
                                output => output
        );

        clk <= not clk after clk_period/2;

        rst_process: process
        begin
                        rst <= '1';
                        wait for rst_length;
                        rst <= '0';
                        wait;
                end process;
```

```
	input_process: process

	begin
			input <= '0';
			wait for min_before_pulse;
			wait until falling_edge(clk);

			for i in 0 to 2 loop
				input <= '1';
				wait for bounce_period/2;
				input <= '0';
				wait for bounce_period/2;
			end loop;

			input <= '1';
			wait for pulse_width;

			for i in 0 to 2 loop
				input <= '0';
				wait for bounce_period/2;
				input <= '1';
				wait for bounce_period/2;
			end loop;

			input <= '0';
			wait;

		end process;

end behavioral;
```