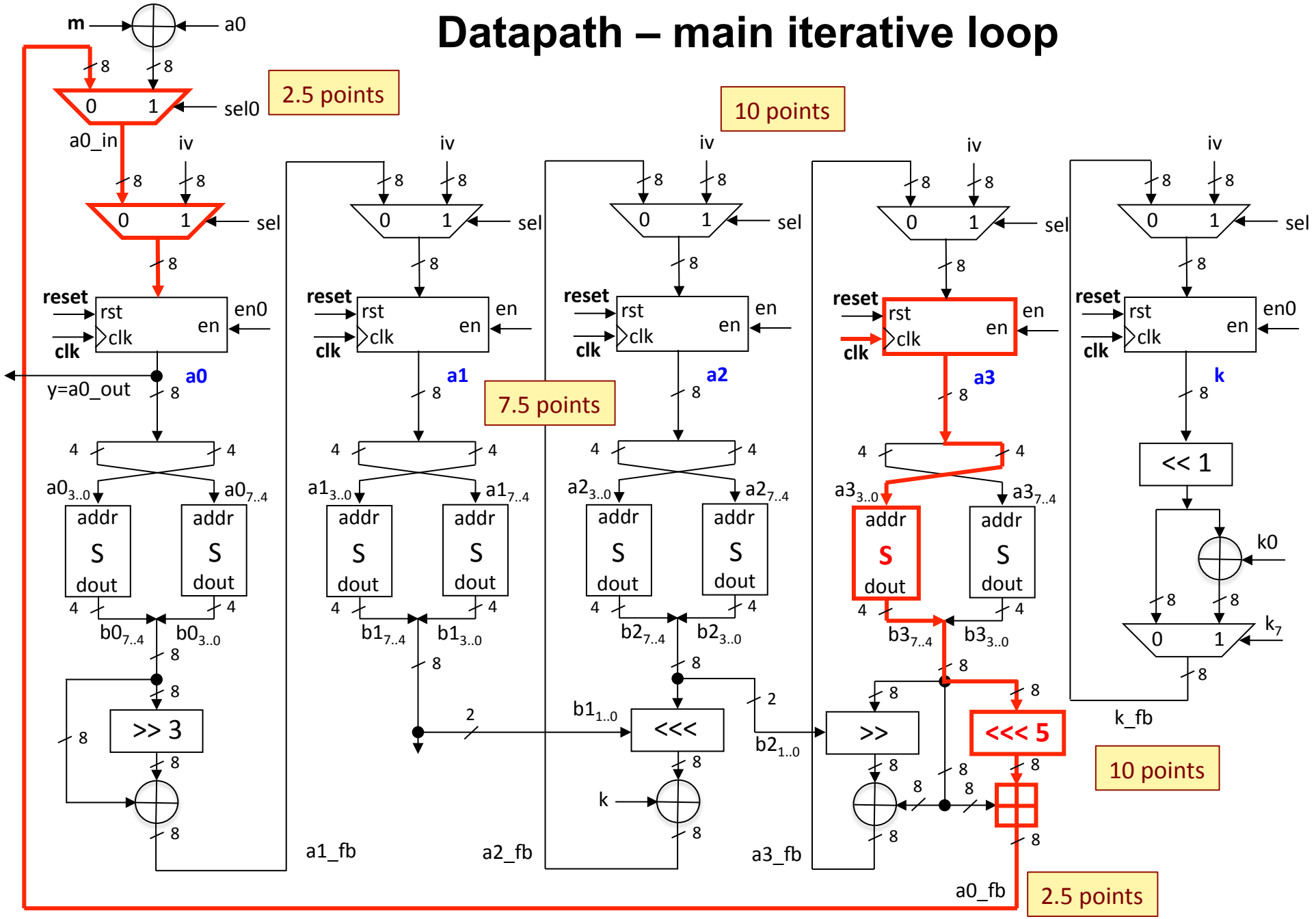
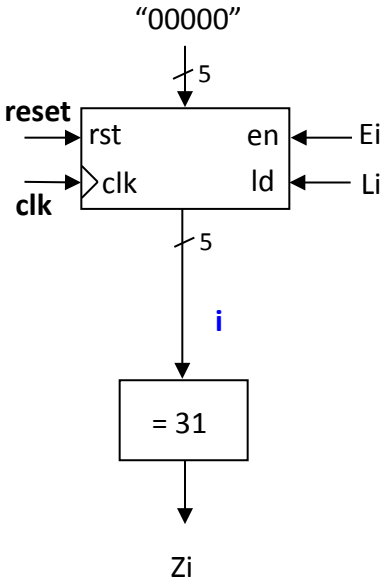
Midterms Exam – Fall 2011
Solutions

Solution to Task 1

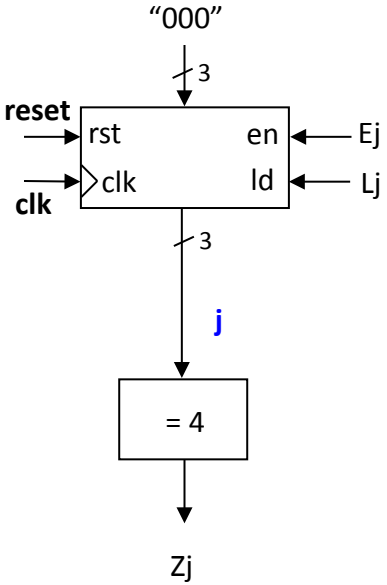
Datapath – main iterative loop



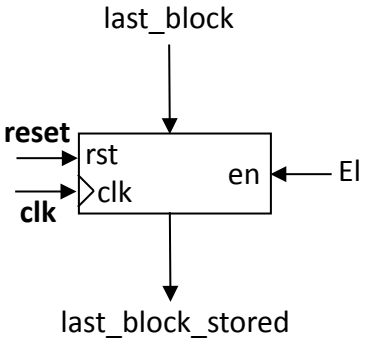
Datapath – counters & state registers



2.5 points



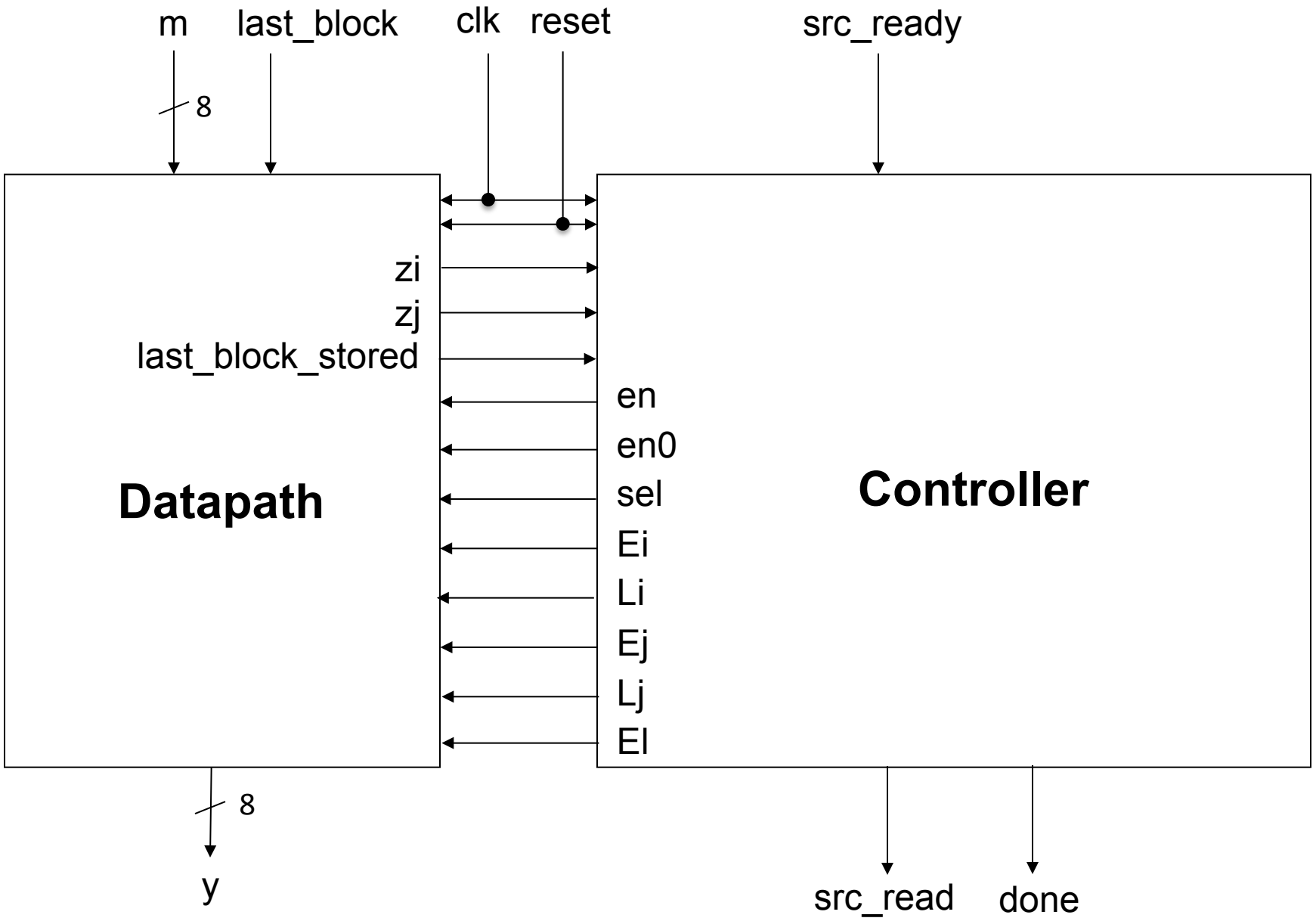
2.5 points



2.5 points

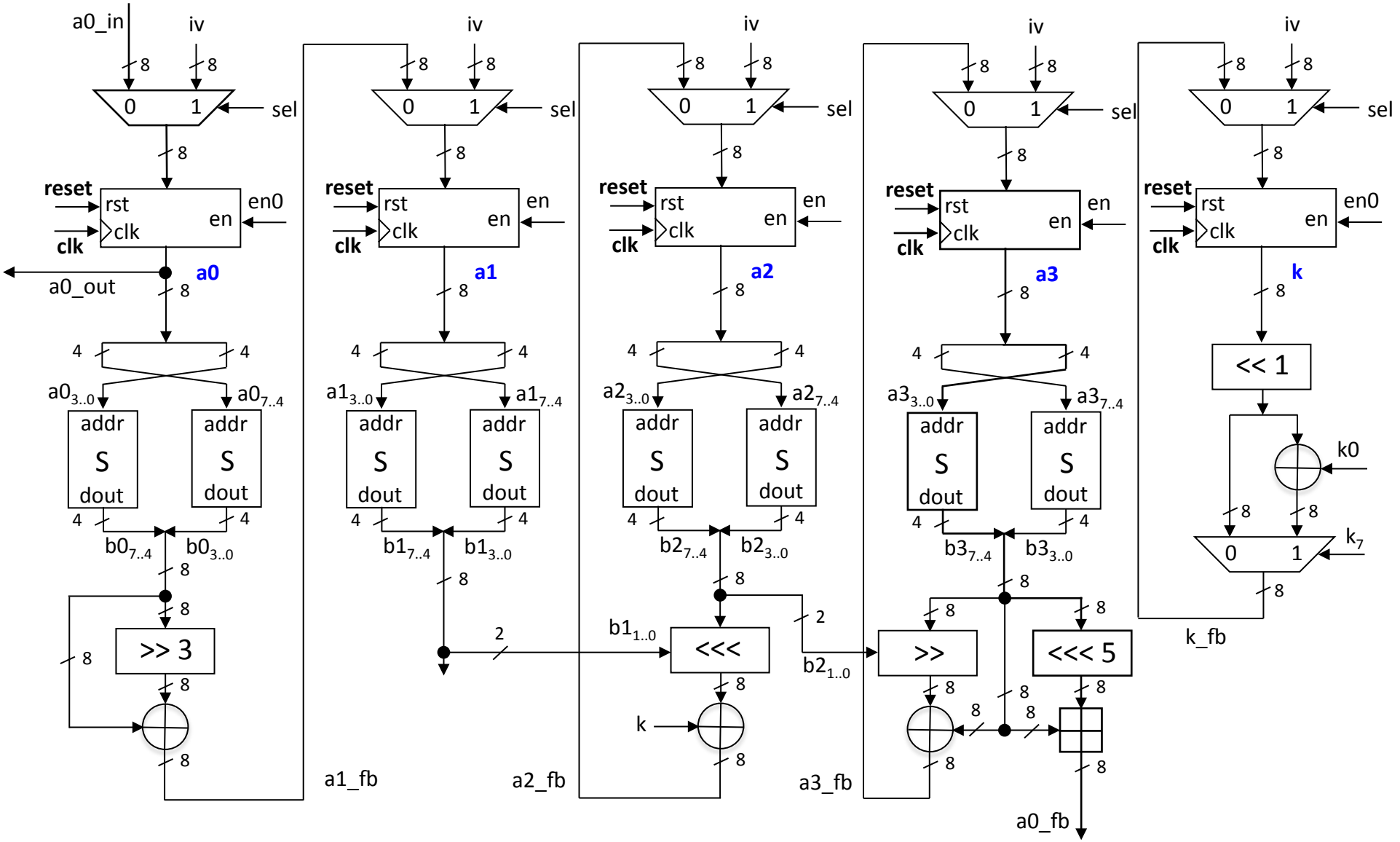
Solution to Task 2

Interface with the division into the Datapath and the Controller



Solution to Task 3

Implemented Circuit – main iterative loop



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg is
	generic (WIDTH : INTEGER := 8);
	port (
	   clk		: in STD_LOGIC;
	   reset	: in STD_LOGIC;
	   en		: in STD_LOGIC;
	   din		: in STD_LOGIC_VECTOR(WIDTH-1 downto 0);
	   dout		: out STD_LOGIC_VECTOR(WIDTH-1 downto 0)
	);
end reg;
```

```
architecture rtl of reg is
begin
	reg_gen : process(clk, reset)
	begin
		if ( reset = '1' ) then
			dout  <= (OTHERS =>'0');
		elsif rising_edge( clk ) then
			if ( en = '1' ) then
				dout <= din;
			end if;
		end if;
	end process;
end rtl;
```

3 points

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.all;

entity ROM_16x4 is
port(
      addr            : in std_logic_vector(3 downto 0);
      dout            : out std_logic_vector(3 downto 0));
end ROM_16x4;

architecture rtl of ROM_16x4 is
      type mem is array (0 to 15) of std_logic_vector(3 downto 0);
      constant my_rom : mem :=
      (  x"E", x"D", x"B", x"0", x"2", x"1", x"4", x"F”,
        x"7", x"A", x"8", x"5", x"9", x"C", x"3", x"6”
      );
begin
      dout <= my_rom(to_integer(unsigned(addr)));
end rtl;
```

5 points

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity main_iterative_loop is
    port
        (
          clk : in  STD_LOGIC;  -- System clock
          reset : in  STD_LOGIC; -- Asynchronous system reset
          sel : in  STD_LOGIC; -- select signal to choose a0, a1, a2 and a3
          en : in  STD_LOGIC; -- enable signal for registers for a1, a2, a3, and k
          en0 : in  STD_LOGIC; -- enable signal for registers for a0
          a0_in : in  STD_LOGIC_VECTOR(7 downto 0);  -- a0 input
          a0_out : out STD_LOGIC_VECTOR(7 downto 0); -- a0 output
          a0_fb : out STD_LOGIC_VECTOR(7 downto 0) -- a0 feedback
        );
end main_iterative_loop;
```

```
architecture rtl of main_iterative_loop is
    -- types
    type array_type_1  is array (0 to 4) of STD_LOGIC_VECTOR(7 downto 0);
    type array_type_2  is array (0 to 3) of STD_LOGIC_VECTOR(7 downto 0);
    type array_type_3  is array (0 to 1) of STD_LOGIC_VECTOR(7 downto 0);

    -- signals
    signal a_in, a: array_type_1;
    signal b, a_fb : array_type_2;
    signal mux_rot, mux_shift : array_type_3;
    signal ens : STD_LOGIC_VECTOR (4 downto 0);

    signal k_in, k, k_fb : STD_LOGIC_VECTOR (7 downto 0);

    -- Initialize constants
    constant iv : STD_LOGIC_VECTOR (7 downto 0) := x"C0" ;
    constant k0 : STD_LOGIC_VECTOR (7 downto 0) := x"1B" ;
```

```
begin
    -- 5 Muxes for once-per-message initialization:
    -- a0 = iv; a1 = iv; a2 = iv; a3 = iv; k = iv;
    a_in(0)  <= iv when sel ='1' else a0_in;
    a_in(1)  <= iv when sel ='1' else a_fb(1);
    a_in(2)  <= iv when sel ='1' else a_fb(2);
    a_in(3)  <= iv when sel ='1' else a_fb(3);
    k_in     <= iv when sel ='1' else k_fb;

    -- 5 Registers for ( a0, a1, a2, a3, k)
    a_in(4) <= k_in;
    ens <= (0=> en0, OTHERS => en);
    reg_gen : for i in 1 to 4 generate
        reg_0 : entity work.reg(rtl)
            port map ( clk => clk,
                       reset => reset,
                       en  => ens(i),
                       din => a_in(i),
                       dout => a(i) );
    end generate;
    k <= a(4);
```

3 points

```
-- 8 instances of S-box module
ROM_gen : for i in 0 to 3 generate
        ROM_0 : entity work.ROM_16x4(rtl)
                port map (addr => a(i)(3 downto 0), dout => b(i)(7 downto 4) );
        ROM_1 : entity work.ROM_16x4(rtl)
                port map (addr => a(i)(7 downto 4), dout => b(i)(3 downto 0) );
end generate;

-- a0 = ((b3 <<< 5) + b3) mod 256
a_fb(0) <= (b(3)(2 downto 0) & b(3)(7 downto 3) ) + b(3);

-- a1 = (b0 >> 3) XOR b0
a_fb(1) <= ( "000" & b(0)(7 downto 3) ) XOR b(0);
```

5 points

2 points

2 points

```
        -- a2 = (b2 <<< (b1 mod 4) ) XOR k
        mux_rot(0) <= b(2) when b(1)(0)='0' else
                    ( b(2)(6 downto 0) & b(2)(7) );
        mux_rot(1) <= mux_rot(0) when b(1)(1)='0' else
                    ( mux_rot(0)(5 downto 0) & mux_rot(0)(7 downto 6) );
        a_fb(2) <= mux_rot(1) XOR k_out;

        -- a3 = (b3 >> (b2 mod 4) ) + b3) XOR b3
        mux_shift(0) <= b(3) when b(2)(0)='0' else
                     ( '0' & b(3)(7 downto 1) );
        mux_shift(1) <= mux_shift(0) when b(2)(1)='0' else
                     ( "00" & mux_shift(0)(7 downto 2) );
        a_fb(3) <= mux_shift(1) XOR b(3);

        -- if k(7)==0 then k = k<<1 else k=(k<<1) XOR k0
        k_fb <= (k(6 downto 0) & '0') when k(7) ='0' else
              ((k(6 downto 0) & '0') XOR k0);

        a0_out <= a(0);
        a0_fb  <= a_fb(0);
end rtl;
```

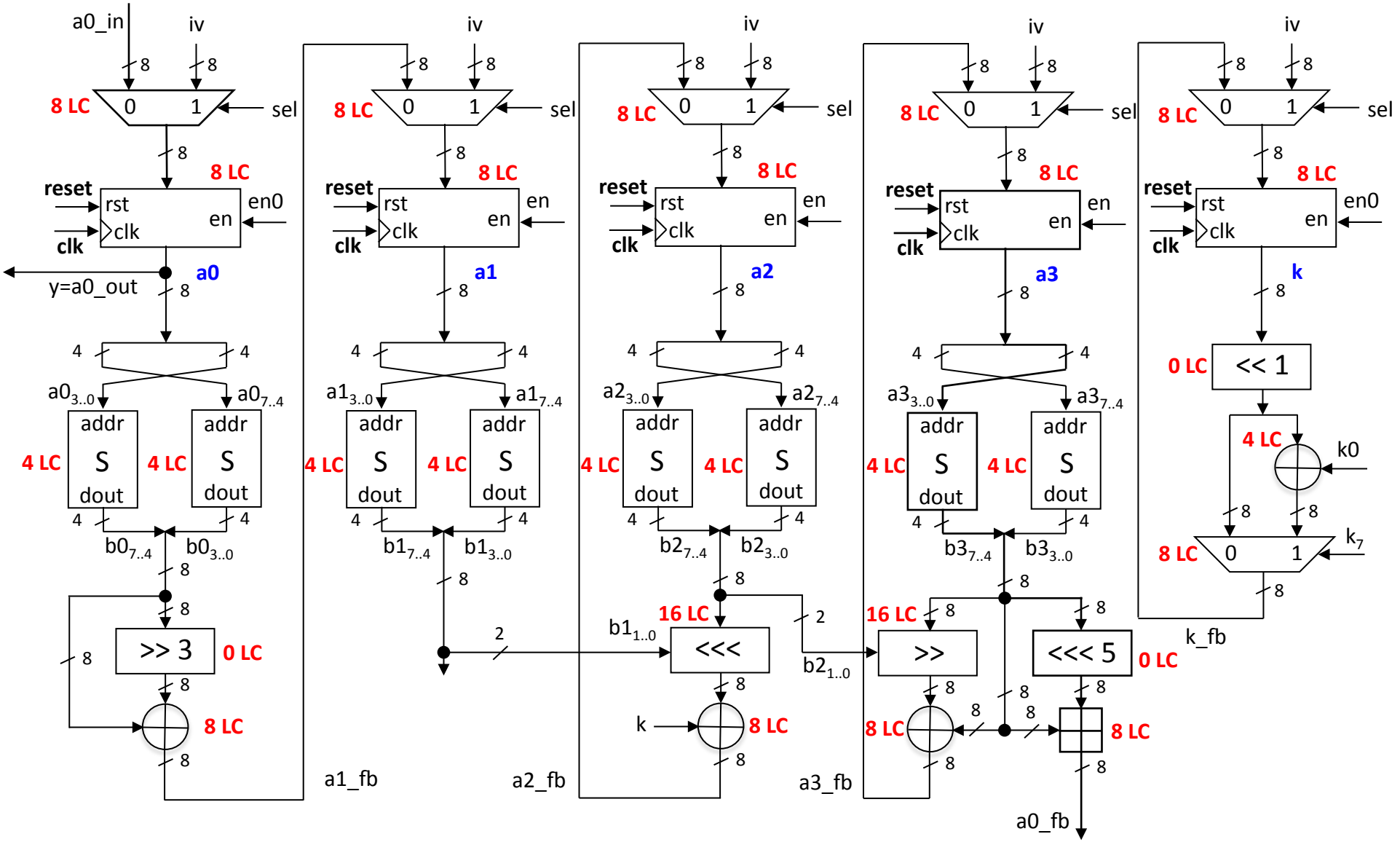
4 points

4 points

2 points

Solution to Task 4

Resource Utilization in Xilinx Spartan 3



Resource Utilization Summary

6 x 8-bit 2-to-1 MUX	=	6 x 8 MLUTs in logic mode	= 48 LC
5 x 8-bit register	=	5 x 8 Storage Elements in FF mode	= 40 LC
8 x 4-by-4 S-box	=	8 x 4 MLUTS in logic mode	= 32 LC
Variable Rotator	=	16 MLUTs in logic mode	= 16 LC
Variable Shifter	=	16 MLUTs in logic mode	= 16 LC
8-bit Adder	=	8 Carry & Control units	= 8 LC
3 x 8-bit XOR	=	24 MLUTs in logic mode	= 24 LC
4 inverters	=	4 MLUTs in logic mode	= 4 LC

Total = 188 LC

Solution to Task 5

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity exam_top_tb is
end exam_top_tb;

architecture behavior of exam_top_tb is
    -- component declaration for the unit under test (uut)
    component exam_top
    port(
         clk : in  std_logic;
         rst : in  std_logic;
         m : in  std_logic_vector(7 downto 0);
         src_ready : in  std_logic;
         src_read : out  std_logic;
         last_block : in  std_logic;
         done : out  std_logic;
         y : out  std_logic_vector(7 downto 0)
        );
    end component;
```

```
--Inputs
 signal clk : std_logic := '0';
 signal rst : std_logic;
 signal m : std_logic_vector(7 downto 0);
 signal src_ready : std_logic;
 signal last_block : std_logic;

--Outputs
 signal src_read : std_logic;
 signal done : std_logic;
 signal y : std_logic_vector(7 downto 0);

 -- Clock period definitions
 constant clk_period : time := 100 ns;
```

1 point

```
BEGIN
  -- instantiate the Unit Under Test (UUT)

  uut: exam_top PORT MAP (
        clk => clk,
        rst => rst,
        m => m,
        src_ready => src_ready,
        src_read => src_read,
        last_block => last_block,
        done => done,
        y => y
      );
```

3 points

```
-- clock generation
 clk_gen: process
 begin
       clk <= '0';
       wait for clk_period/2;
       clk <= '1';
       wait for clk_period/2;
 end process;

 --reset generation
 reset_gen: process
 begin
       rst <= '1';
       wait for clk_period;
       rst <= '0';
       wait;
 end process;
```

3 points

2 points

```
 -- Stimuli process
  stim_proc: process
  begin
        m <= x"FF";
        src_ready <= '1';
        last_block <= '0';

        for i in 0 to 2 loop
           wait until src_read='1';
           wait until src_read='0';
        end loop;

        last_block <= '1';
        wait until src_read='1';
        wait until src_read='0';

        wait until done='1';
        wait;

   end process;
end;
```

2 points

4 points