Problem 1 [10 points]

Show how to implement an address decoder that recognizes the following five ranges of the 16-bit address A, and generates the corresponding enable signals e0-e4, indicating that the address is in the given range:

Range 0: 0000-07FF
Range 1: 0800-0FFF
Range 2: 1000-17FF
Range 3: 1800-1FFF
Range 4: 2000-27FF

Problem 2 [15 points]

Given eight 32x1 RAMs and basic combinational logic components, show (using a block diagram) how to build a single 128x2 RAM.

Assume that all RAMs have only the following ports: ADDR (address), DIN (data in), WE (write enable), CLK (clock), and DOUT (data out).

In your block diagram, clearly specify
• names, widths and directions of all buses
• names, widths and directions of all inputs and outputs of logic components.

Problem 3 [60 points]

The EXAM function is specified below using its:
   a. Pseudocode
   b. Table of input/output ports
   c. Timing requirements.

1. Pseudocode:

begin:
   wait for s=1

   a0 := iv
   a1 := iv
   a2 := iv
   a3 := iv
   k := ivp
   j := 0
   last_block_stored := 0

   do
      if (last_block_stored /= 1) then
         wait until src_ready
         last_block_stored := last_block
         a3 := a3 ⊕ m
      end if
for i from 0 to 63 do
    b(0) := S[a0/16] || S[a0 mod 16]
    b(1) := a1 <<< S[a0 mod 16]
    b(2) := a2 * (4*a2+3) / 2^10
    b(3) := (8*a3 ⊕ 6) >> 3
    a0 := b(k mod 4)
    d0 := b(0) ⊕ b(1)
    d1 := b(2) ⊕ b(3)
    a1 := d0 · d1
    if (d0 is odd)
        a2 := d0
    else
        a2 := d1
    if (((b(3) mod 16) is prime)
        a3 := b(3)
    elsif (((b(2) mod 16) is prime)
        a3 := b(2)
    elsif (((b(1) mod 16) is prime)
        a3 := b(1)
    elsif (((b(0) mod 16) is prime)
        a3 := b(0)
    else
        a3 := ivp;
    if (k0=0)
        k := k>>1
    else
        k :=(k>>1) ⊕ (k0 << 7)
    end if
    end for

    if (last_block_stored = 1) then
        j++
    end if;

    until ((last_block_stored = 1) and (j=8))

wait for s=0
y := a0 || a1 || a2 || a3
go to begin

Notation:

\textbf{m}: 8-bit message block (input)
\textbf{y}: 32-bit circuit output (output)
\textbf{iv}, \textbf{ivp}: 8-bit initialization vectors (constants), iv='76', ivp='0B' (in hexadecimal notation)
\textbf{a0..a3, b(0)..b(3)}, \textbf{k}, \textbf{d0, d1}: 8-bit intermediate values, treated as 8-bit unsigned integers
\textbf{k0}: bit 0 of \textbf{k}

Operations:

\textbf{X ⊕ Y}: bitwise XOR
\textbf{X · Y}: bitwise AND
\textbf{X + Y}: addition
\textbf{X * Y}: multiplication
\textbf{X || Y}: \textbf{X} concatenated with \textbf{Y}
\textbf{X/16}: integer part of the result of division of \textbf{X} by 16
\[ X \ll Y : \text{rotation of } X \text{ to the left by the number of positions given in } Y \]
\[ X \gg Y : \text{rotation of } X \text{ to the right by the number of positions given in } Y \]
\[ X \ll Y : \text{logical shift of } X \text{ to the left by the number of positions given in } Y \]
\[ X >> Y : \text{logical shift of } X \text{ to the right by the number of positions given in } Y \]

\[ Y = S[X] : \text{substitution defined using the following table (all values in the hexadecimal notation):} \]

<table>
<thead>
<tr>
<th>X ( \text{hex} )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y ( \text{hex} )</td>
<td>E</td>
<td>D</td>
<td>B</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>F</td>
<td>7</td>
<td>A</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>C</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

2. Table of input/output ports:

<table>
<thead>
<tr>
<th>Port</th>
<th>Mode</th>
<th>Width</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>Input</td>
<td>1</td>
<td>System clock.</td>
</tr>
<tr>
<td>reset</td>
<td>Input</td>
<td>1</td>
<td>Asynchronous system reset.</td>
</tr>
<tr>
<td>s</td>
<td>Input</td>
<td>1</td>
<td>Operating mode: 0 = waiting for data / reading results, 1 = processing.</td>
</tr>
<tr>
<td>m</td>
<td>Input</td>
<td>8</td>
<td>8-bit message block.</td>
</tr>
<tr>
<td>src_ready</td>
<td>Input</td>
<td>1</td>
<td>Control signal indicating that the source is ready. Must remain active until source is read.</td>
</tr>
<tr>
<td>src_read</td>
<td>Output</td>
<td>1</td>
<td>Control signal confirming that the source was read. Active for one clock cycle.</td>
</tr>
<tr>
<td>last_block</td>
<td>Input</td>
<td>1</td>
<td>Control signal indicating the last block of the message.</td>
</tr>
<tr>
<td>done</td>
<td>Output</td>
<td>1</td>
<td>Control signal indicating that the output is ready.</td>
</tr>
<tr>
<td>yout</td>
<td>Output</td>
<td>32</td>
<td>32-bit output block ( y ) when s=0, high impedance otherwise.</td>
</tr>
</tbody>
</table>

3. Timing Requirements:

Assume that

- one clock cycle is used for the once-per-message initialization:
  \( a_0 := iv; a_1 := iv; a_2 := iv; a_3 := iv; k := ivp \)
- one round of the main for-loop of the pseudocode executes in one clock cycle; there are a total of 64 rounds.

Tasks:

Task 1: Block Diagram [45 points]

Draw a block diagram of the datapath of the EXAM circuit using medium complexity components corresponding to the operations used in the pseudocode.

Your Datapath should

a) be capable of executing the entire pseudocode given in point 1,

b) match interface given in point 2, and

c) meet all timing requirements specified in point 3.

Clearly specify

- names, widths and directions of all buses
- names, widths and directions of all inputs and outputs of the logic components.
Assume that one round of the main for-loop of the pseudocode executes in one clock cycle. Minimize the number of control signals to be generated by the Control Unit.

Task 2: Substitution S and the function “Is prime” [5 points]

Explain in detail your implementation of the substitution function $Y=S[X]$. Which logic component do you use to implement this function? How is this component initialized?

Show in detail your implementation of the function “Is prime”, which checks whether a given number between 0 and 15 is prime.

Assume that this function returns
1 if the input is a prime number, and returns
0 for the inputs 0 and 1, and for all composite numbers greater or equal to 2.

Task 3: Interface [10 points]

Draw an interface of the EXAM circuit, with the division into the Datapath and Controller. Show the names, widths, and directions of all signals forming this interface.

Problem 4 [15 points]

Fill in the blanks in the code of MISR (Multiple Input Signature Register), provided in the answer sheet. Do not write this code from scratch! The block diagram of MISR is shown in figure below. Assume that the circuit has an asynchronous reset.