Introduction & tasks:

The MD-5 hash function circuit is specified below using its:
   a. pseudocode
   b. interface
   c. table of input/output ports
   d. timing requirements (in the description of Task 1).

Perform the following 5 phases of the design process for the MD-5 implementation:
   1. Block diagram of the Datapath
   2. Interface with the division into the Datapath and the Controller
   3. ASM chart of the Controller
   4. VHDL code of the Controller
   5. Analysis of the possible use of embedded resources

Pseudocode:

```
//Initialize hash result:
var int h0 := iv0
var int h1 := iv1
var int h2 := iv2
var int h3 := iv3

//Process the message in successive 512-bit blocks:
for each 512-bit block of the message
   break the block into sixteen 32-bit words w[j], 0 ≤ j ≤ 15

//Initialize the state for the block:
var int a := h0
var int b := h1
var int c := h2
var int d := h3

//Main loop:
for i from 0 to 63
   if 0 ≤ i ≤ 15 then
      f := (b and c) or ((not b) and d)
      g := i
   else if 16 ≤ i ≤ 31
      f := (d and b) or ((not d) and c)
      g := (5×i + 1) mod 16
   else if 32 ≤ i ≤ 47
      f := b xor c xor d
      g := (3×i + 5) mod 16
   else if 48 ≤ i ≤ 63
      f := c xor (b or (not d))
      g := (7×i) mod 16
   end if
   temp := d
   d := c
   c := b
   b := b + leftrotate((a + f + k[i] + w[g]), r[i])
   a := temp
end for
```
/\text{Add the state to the hash result so far:}
\begin{align*}
h_0 & := h_0 + a \\
h_1 & := h_1 + b \\
h_2 & := h_2 + c \\
h_3 & := h_3 + d \\
\end{align*}
end for

digest := h_0 || h_1 || h_2 || h_3

\textbf{Notation:}

\textit{All variables, except i, g, r[i], and digest, represent 32-bit words.}

\textit{iv0..iv3} : initialization vector
\textit{h0..h3} : intermediate hash result
\textit{digest} : hash value
\textit{k[i]} : round constant, specific to each round. All round constants are precomputed and stored in a Block RAM operating in a single-port ROM mode.
\textit{w[j]} : message block words
\textit{not} : one’s complement of a 32-bit word.
\textit{xor, and, or} : Boolean operations on 32-bit words.
\textit{leftrotate(a, r)} : rotation of the variable a by r positions to the left
\textit{r[i]}: rotation amount specific to a given round. Rotation amounts are specified below, and are assumed to be stored in a distributed ROM.
\textit{r[0..15]}:={7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22}
\textit{r[16..31]}:={5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20}
\textit{r[32..47]}:={4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23}
\textit{r[48..63]}:={6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21}
\textit{a || b} : a concatenated with b
Interface:

Assume the following interface to your circuit:

Table of input/output ports:

<table>
<thead>
<tr>
<th>Port</th>
<th>Width</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>1</td>
<td>System clock.</td>
</tr>
<tr>
<td>reset</td>
<td>1</td>
<td>System reset – clears all internal register in the datapath and resets control unit.</td>
</tr>
<tr>
<td>mi</td>
<td>32</td>
<td>Message input. Message blocks are read from the source one word at a time.</td>
</tr>
<tr>
<td>hn</td>
<td>32</td>
<td>Hash value output. The hash value is written to the destination one word at a time.</td>
</tr>
<tr>
<td>src_ready</td>
<td>1</td>
<td>Input control signal indicating that the source of data is ready with the next message word.</td>
</tr>
<tr>
<td>src_read</td>
<td>1</td>
<td>Output control signal indicating reading the next message word from the source of data.</td>
</tr>
<tr>
<td>dst_ready</td>
<td>1</td>
<td>Input control signal indicating that the destination of data is ready to receive the next output word.</td>
</tr>
<tr>
<td>dst_write</td>
<td>1</td>
<td>Output control signal indicating writing the next output word to the destination of data.</td>
</tr>
<tr>
<td>last_block</td>
<td>1</td>
<td>Input control signal indicating that the last word of the message is being read.</td>
</tr>
<tr>
<td>done</td>
<td>1</td>
<td>Set to ‘1’ when the entire hash value was written to the destination.</td>
</tr>
</tbody>
</table>
Tasks & Assumptions:

Task 1 [30 points]: Draw block diagram of the Datapath of the circuit

Assume that

- the target FPGA device is Spartan 3
- round constants are kept in a single-port embedded ROM
- rotation constants are kept in a distributed ROM
- one clock cycle is used for the once-per-message initialization:
  \[ h_0 = iv_0; \quad h_1 = iv_1; \quad h_2 = iv_2; \quad h_3 = iv_3; \]
- one clock cycle is used for the once-per-block initialization:
  \[ a = h_0; \quad b = h_1; \quad c = h_2; \quad d = h_3; \]
- one round of the main for loop of the pseudocode executes in one clock cycle;
  there are a total of 64 rounds.
- one clock cycle is used for the once-per-block finalization:
  \[ h_0 = h_0 + a; \quad h_1 = h_1 + b; \quad h_2 = h_2 + c; \quad h_3 = h_3 + d; \]

As a result, hashing of the message \( M \), consisting of \( N \) 512-bit blocks (each block=16 32-bit words) should last \( 1+(1+64+1) \times N \) clock cycles. The hash value is then written to the destination circuit in 4 additional clock cycles.

Please clearly mark the widths and directions of all buses in your block diagram.

Task 2 [10 points]: Draw an interface between the Control Unit and the Datapath

Show the names and widths of all signals forming this interface.

Task 3 [30 points]: Draw an ASM chart of a controller capable of performing computations described in the pseudocode. Translate each action and condition in your ASM chart to the values of the corresponding control and status signals.

Task 4 [20 points]: Write synthesizable VHDL code of the controller defined in Task 3. You can write architecture only, no need for entity declaration.

Task 5 [10 points]:

Assuming the maximum use of embedded resources available in Spartan 3 FPGAs, how many embedded resources of each type can be used to implement MD 5 in Spartan 3?
In case of block memories, please specify the configuration (address size and output size of a Block RAM) and the actual size of the memory portion used to store data.

How would you go about specifying the use of these resources in your VHDL code?
- Please choose and shortly describe one method.
- Please provide a short justification (a list of advantages) of the selected method.