Midterm Exam

ECE 448
Spring 2011

Wednesday Section

(15 points)

Instructions:

Please read this entire document carefully before beginning!

Zip all your deliverables into an archive <last_name>.zip and submit it through Blackboard no later than Wednesday, March 9, 10:15 PM EST.
Your task is to describe in VHDL, debug, and implement a simple hash function.

Introduction:

The AV-4 hash function circuit is specified below using its:
   a. Pseudocode,
   b. Block Diagram,
   c. Interface,
   d. Table of input/output ports,
   e. Timing requirements.

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 64-bit blocks:
for each 64-bit block of the message
   break the block into four 16-bit words w[i], 0 ≤ i ≤ 3

   //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 3
      f := (b and c) or ((not b) and d)
      wi := w[i]
      temp := d
      d := c
      c := b
      b := b + leftrotate((a + f + ki + wi), 12)
      a := temp
   end for

   //Add the state to the hash result so far:
   h0 := h0 + a
   h1 := h1 + b
   h2 := h2 + c
   h3 := h3 + d
end for

digest := h0 || h1 || h2 || h3
Notation:

All variables, except $i$ and digest, represent 16-bit words.

- $iv0..iv3$: initialization vector
- $h0..h3$: intermediate hash result
- $digest$: output value
- $ki$: round constant
- $w[i]$: message block words, $i=0..3$ for a single message block, $w[0]$ represents the least significant word of the message block.
- $not$: one’s complement of a 16-bit word.
- $xor$, $and$, $or$: Boolean operations on 16-bit words.

$\text{leftrotate}(a, r)$: rotation of the variable $a$ by $r$ positions to the left

$a || b$: a concatenated with $b$.

Block Diagram:

All registers have clock, reset, and enable even if these inputs are not shown in the diagram. Enable signals are specified in the table on the next page. $a$, $b$, $c$, $d$ signals mean output of the respective register.
**Interface:**

Assume the following interface to your circuit.

![Diagram of AV-4 interface](image)

**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 – clears all internal register.</td>
</tr>
<tr>
<td>enh</td>
<td>Input</td>
<td>1</td>
<td>Common enable for the registers h0, h1, h2 and h3.</td>
</tr>
<tr>
<td>ena</td>
<td>Input</td>
<td>1</td>
<td>Common enable for registers a, b, c and d.</td>
</tr>
<tr>
<td>selh</td>
<td>Input</td>
<td>1</td>
<td>Select signal to choose between initialization and loading in new value to registers h0, h1, h2, and h3.</td>
</tr>
<tr>
<td>sela</td>
<td>Input</td>
<td>1</td>
<td>Select signal to choose between initialization and loading in new value to registers a, b, c and d.</td>
</tr>
<tr>
<td>wi</td>
<td>Input</td>
<td>16</td>
<td>A subsequent word of a message block, starting from w[0] and ending with w[3].</td>
</tr>
<tr>
<td>digest</td>
<td>Output</td>
<td>64</td>
<td>digest = h0</td>
</tr>
</tbody>
</table>
Timing Requirements:

Assume that
- one clock cycle is used for the once-per-message initialization:
  \[ h_0 = iv_0; h_1 = iv_1; h_2 = iv_2; h_3 = iv_3; \]
- one clock cycle is used for the once-per-block initialization:
  \[ a = h_0; b = h_1; c = h_2; d = h_3; \]
- one round of the main for loop of the pseudocode executes in one clock cycle; there are a total of 4 rounds.
- one clock cycle is used for the once-per-block finalization:
  \[ h_0 = h_0 + a; h_1 = h_1 + b; h_2 = h_2 + c; h_3 = h_3 + d; \]

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

Constants:

\[ iv_0 = 0xefcd \]
\[ iv_1 = 0x98ba \]
\[ iv_2 = 0x1032 \]
\[ iv_3 = 0xdcfe \]
\[ ki = 0xab89 \]

Inputs, Outputs and Intermediate Values:

Input Message Words:

Block 1:
\[ w[0] = 0x9e10 \]
\[ w[1] = 0x7d9d \]
\[ w[2] = 0x372b \]
\[ w[3] = 0xb682 \]

Block 2:
\[ w[0] = 0x1d35 \]
\[ w[1] = 0x42a4 \]
\[ w[2] = 0x19d6 \]
\[ w[3] = 0xb682 \]

Output:

\[ \text{digest} = 0xe36b86735ac67b8c \]
Intermediate values of the variables a-d, and h0-h3 during the circuit operation:
All values are represented in hexadecimal notation.

<table>
<thead>
<tr>
<th>Clock cycle</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>h0</th>
<th>h1</th>
<th>h2</th>
<th>h3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>efcd</td>
<td>98ba</td>
<td>1032</td>
<td>defe</td>
</tr>
<tr>
<td>2</td>
<td>efcd</td>
<td>98ba</td>
<td>1032</td>
<td>defe</td>
<td>efcd</td>
<td>98ba</td>
<td>1032</td>
<td>defe</td>
</tr>
<tr>
<td>3</td>
<td>defe</td>
<td>6197</td>
<td>98ba</td>
<td>1032</td>
<td>efcd</td>
<td>98ba</td>
<td>1032</td>
<td>defe</td>
</tr>
<tr>
<td>4</td>
<td>1032</td>
<td>c304</td>
<td>6197</td>
<td>98ba</td>
<td>efcd</td>
<td>98ba</td>
<td>1032</td>
<td>defe</td>
</tr>
<tr>
<td>5</td>
<td>98ba</td>
<td>07ce</td>
<td>c304</td>
<td>6197</td>
<td>efcd</td>
<td>98ba</td>
<td>1032</td>
<td>defe</td>
</tr>
<tr>
<td>6</td>
<td>6197</td>
<td>adab</td>
<td>07ce</td>
<td>c304</td>
<td>efcd</td>
<td>98ba</td>
<td>1032</td>
<td>defe</td>
</tr>
<tr>
<td>7</td>
<td>6197</td>
<td>adab</td>
<td>07ce</td>
<td>c304</td>
<td>5164</td>
<td>4665</td>
<td>1800</td>
<td>a002</td>
</tr>
<tr>
<td>8</td>
<td>5164</td>
<td>4665</td>
<td>1800</td>
<td>a002</td>
<td>5164</td>
<td>4665</td>
<td>1800</td>
<td>a002</td>
</tr>
<tr>
<td>9</td>
<td>a002</td>
<td>9207</td>
<td>4665</td>
<td>1800</td>
<td>5164</td>
<td>4665</td>
<td>1800</td>
<td>a002</td>
</tr>
<tr>
<td>10</td>
<td>1800</td>
<td>db8a</td>
<td>9207</td>
<td>4665</td>
<td>5164</td>
<td>4665</td>
<td>1800</td>
<td>a002</td>
</tr>
<tr>
<td>11</td>
<td>4665</td>
<td>42c6</td>
<td>db8a</td>
<td>9207</td>
<td>5164</td>
<td>4665</td>
<td>1800</td>
<td>a002</td>
</tr>
<tr>
<td>12</td>
<td>9207</td>
<td>400e</td>
<td>42c6</td>
<td>db8a</td>
<td>5164</td>
<td>4665</td>
<td>1800</td>
<td>a002</td>
</tr>
<tr>
<td>13</td>
<td>9207</td>
<td>400e</td>
<td>42c6</td>
<td>db8a</td>
<td>e36b</td>
<td>8673</td>
<td>5ac6</td>
<td>7b8c</td>
</tr>
</tbody>
</table>

Timing Waveform:
**Design Requirements**

The combinational portion of the circuit should be described using the dataflow VHDL code, and the sequential portion of the circuit should be described using the synthesizable behavioral code. Your code should infer a circuit that requires a minimum amount of FPGA resources. The target clock frequency should be 50 MHz.

**Tasks**

Perform the following tasks:

1. Write a synthesizable VHDL code representing the described above circuit.
2. Write a testbench verifying the operation of your circuit for inputs shown given above.
3. Perform functional simulation of your circuit and use it to debug your VHDL code. Take screen shots of the waveform.
4. Synthesize your circuit.
5. Implement your circuit using
   - FPGA family: Spartan3E,
   - Device: XC3S1600E
   - Speed Grade: -4.
6. Run the static timing analysis of your circuit.
7. Based on the circuit block diagram and the implementation reports, determine the most critical path in your circuit and the circuit maximum clock frequency.
8. Based on the implementation reports and the report from the static timing analysis, determine the number of CLB slices, Logic Cells, LUTs, D flip-flops, and pins used by the circuit.

**Deliverables**

1. VHDL code of your entire circuit fulfilling the requirements specified in the Design Requirements section above.
2. VHDL code of your testbench.
3. Timing waveforms from the functional and timing simulations demonstrating the correct operation of your circuit. Take screen shots and include in the report.
4. Description of the critical path of your circuit.
5. FPGA resource utilization (as defined in Task 8 above)
6. Minimum clock period and maximum clock frequency of your circuit.