Lecture 5B

Block Diagrams

HASH Example
Structure of a Typical Digital System

Datapath (Execution Unit)

Data Inputs

Control Signals

Status Signals

Data Outputs

Controller (Control Unit)

Control & Status Inputs

Control & Status Outputs
Hardware Design with RTL VHDL

- Pseudocode
- Interface
- Datapath
  - Block diagram
  - VHDL code
- Controller
  - ASM chart
  - VHDL code
Steps of the Design Process
Introduced in Class Today

1. Text description
2. Interface
3. Pseudocode
4. **Block diagram of the Datapath**
5. **Interface divided into Datapath and Controller**
6. ASM chart of the Controller
7. RTL VHDL code of the Datapath, Controller, and Top-level Unit
8. Testbench for the Datapath, Controller, and Top-Level Unit
9. Functional simulation and debugging
10. Synthesis and post-synthesis simulation
11. Implementation and timing simulation
12. Experimental testing using FPGA board
Class Exercise 3
HASH FUNCTION
Hash function

arbitrary length

m

message

hash function

h

h(m)

hash value

fixed length
Authentication
Message Authentication Code - MAC

Bob

Alice

$K_{AB}$ - Secret key of Alice and Bob
IV – Initialization Vector
HMAC

\[
\text{KEY} \oplus \text{opad} = \text{KEY'}
\]

\[
\text{KEY'} \concat \text{message } m \rightarrow \text{h}
\]

- American standard FIPS 198
- Arbitrary hash function and key size
Pseudocode (1)

\[ h_0 := \text{iv0} \]
\[ h_1 := \text{iv1} \]
\[ h_2 := \text{iv2} \]
\[ h_3 := \text{iv3} \]
\[ \text{last_block_stored} := 0 \]

while (last_block_stored != 1)
do
    wait until src_ready
    \[ r := m \]
    \[ \text{last_block_stored} := \text{last_block} \]
    \[ a := h_0 \]
    \[ b := h_1 \]
    \[ c := h_2 \]
    \[ d := h_3 \]
Pseudocode (2)

for i = 0 to 63 do
    ki := k[i mod 16]
    ri := r[i mod 4]
    f := (17*ri) mod 2^8
    e := (2d+1)*f mod 2^8
    d := (4c+2)*c mod 2^8
    c := b <<< 3
    b := a >>> i
    a := e ⊕ ki
end for

h0 := (h0 + a) mod 2^8
h1 := (h1 + b) mod 2^8
h2 := (h2 + c) mod 2^8
h3 := (h3 + d) mod 2^8
end while

y := h0 || h1 || h2 || h3
done := 1
**Notation**

\[ m: \text{32-bit message block (input)} \]
\[ r: \text{32-bit register} \]
\[ y: \text{32-bit circuit output (output)} \]
\[ \text{iv0..iv3 : 8-bit initialization vectors (constants)} \]
\[ \text{a..f, h0..h3, ki, ri : 8-bit intermediate values, treated as 8-bit unsigned integers} \]
\[ k[i]: \text{8-bit round constants: k[0]..k[15], stored in ROM} \]
\[ r[i]: \text{bytes of the 32-bit register r, where} \]
\[ \quad r[0] \text{ represents the least significant byte of r, and} \]
\[ \quad r[3] \text{ represents the most significant byte of r.} \]
Operations

⊕ : XOR

X <<< Y : rotation of X to the left by the number of positions given in Y

X >>> Y : rotation of X to the right by the number of positions given in Y

X || Y: X concatenated with Y
# Interface Table

<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>m</td>
<td>Input</td>
<td>32</td>
<td>32-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>y</td>
<td>Output</td>
<td>32</td>
<td>Output y = 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 64 rounds.
- one clock cycle is used for the once-per-block finalization:
  \[ h_0 = h_0 + a; \ h_1 = h_1 + b; \ h_2 = h_2 + c; \ h_3 = h_3 + d; \]
Tasks

Draw a block diagram of the Datapath of the HASH circuit, using medium complexity components corresponding to the operations used in the pseudocode.

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.

Mark the most likely critical path in your circuit.
HASH: Solutions
Interface with the Division into the Datapath and Controller