Lecture 3
RTL Design Methodology
Transition from Pseudocode & Interface to a Corresponding Block Diagram

Structure of a Typical Digital System
Data Inputs
Datapath (Execution Unit)
Controller (Control Unit)
Data Outputs
Control & Status Outputs

Hardware Design with RTL VHDL
Pseudocode
Datapath
Controller
Block diagram
VHDL code
Interface
Block diagram
VHDL code
State diagram or ASM chart
VHDL code

Steps of the Design Process
1. Text description
2. Interface
3. Pseudocode
4. Block diagram of the Datapath
5. Interface with the division into the Datapath and the Controller
6. ASM chart of the Controller
7. RTL VHDL code of the Datapath, the Controller, and the Top Unit
8. Testbench of the Datapath, the Controller, and the Top Unit
9. Functional simulation and debugging
10. Synthesis and post-synthesis simulation
11. Implementation and timing simulation
12. Experimental testing

Steps of the Design Process Practiced in Class Today
1. Text description
2. Interface
3. Pseudocode
4. Block diagram of the Datapath
5. Interface with the division into the Datapath and the Controller
6. ASM chart of the Controller
7. RTL VHDL code of the Datapath, the Controller, and the Top Unit
8. Testbench of the Datapath, the Controller, and the Top Unit
9. Functional simulation and debugging
10. Synthesis and post-synthesis simulation
11. Implementation and timing simulation
12. Experimental testing

Statistics example
Pseudocode

\[ \text{no}_1 = \text{no}_2 = \text{no}_3 = \text{sum} = 0 \]

for \( i=0 \) to \( k-1 \) do

\[
\text{sum} = \text{sum} + \text{din}
\]

if \( \text{din} > \text{no}_1 \) then

\[
\text{no}_3 = \text{no}_2 \\
\text{no}_2 = \text{no}_1 \\
\text{no}_1 = \text{din}
\]

elseif \( \text{din} > \text{no}_2 \) then

\[
\text{no}_3 = \text{no}_2 \\
\text{no}_2 = \text{din}
\]

elseif \( \text{din} > \text{no}_3 \) then

\[
\text{no}_3 = \text{din}
\]

end if

end for

\[
\text{avr} = \text{sum} / k
\]
One of the fastest ciphers

Basic operations:

- Rotation by a variable number of bits:
  \[ C = A <<< B \]

- Addition modulo \(2^w\):
  \[ C = A + B \mod 2^w \]

**RC5**

- **w/r/b**
  - \(w = 16, 32, 64\)
  - input/output block = 2 words = \(2w\) bits
  - Typical value: \(w = 32\) \(\Rightarrow\) 64-bit input/output block
  - \(r\) - number of rounds
  - \(0 \leq b \leq 255\)
  - key size in bits = \(8\cdot b\) bits
  - Recommended version: **RC5 32/12/16**
  - 64 bit block
  - 12 rounds
  - 128 bit key

**Pseudocode**

Split \(M\) into two halves \(A\) and \(B\), \(w\) bits each

\[
\begin{align*}
A &= A + S[0] \\
B &= B + S[1] \\
\text{for } j = 1 \text{ to } r \text{ do } \\
&\quad \{ \\
&\quad \quad A' = ((A \oplus B) <<< B) + S[2j] \\
&\quad \quad B' = ((B \oplus A') <<< A') + S[2j+1] \\
&\quad \quad A = A' \\
&\quad \quad B = B' \\
&\quad \} \\
C &= A || B
\end{align*}
\]

**Notation**

- \(A, B, A', B' = w\)-bit variables
- \(S[2j], S[2j+1] = \) a pair of round keys, each round key is a \(w\)-bit variable
- \(\oplus = \) an XOR of two \(w\)-bit words
- \(+ = \) unsigned addition mod \(2^w\)
- \(A <<< B = \) rotation of the variable \(A\) by a number of positions given by the current value of the variable \(B\)
- \(A || B = \) concatenation of \(A\) and \(B\)

The algorithms has two parameters:
- \(r = \) number of rounds (e.g., 3)
- \(w = \) word size (always a power of 2, e.g., \(w = 2^4 = 16\))

**Circuit Interface**
Circuit Interface

<table>
<thead>
<tr>
<th>Part</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 - clear internal registers</td>
</tr>
<tr>
<td>M</td>
<td>2w</td>
<td>Message block</td>
</tr>
<tr>
<td>write_M</td>
<td>1</td>
<td>Synchronous write control signal for the message block M. After the block M is written to the RC5 unit, the encryption of M starts automatically.</td>
</tr>
<tr>
<td>Si</td>
<td>w</td>
<td>Round key $S_i$ loaded to one of the two internal memories. The first memory stores values of $S_i=2j$, i.e., only round keys with even indices. The second memory stores values of $S_i=2j+1$, i.e., only round keys with odd indices.</td>
</tr>
<tr>
<td>write_Si</td>
<td>1</td>
<td>Synchronous write control signal for the round key $S_i$.</td>
</tr>
<tr>
<td>i</td>
<td>w</td>
<td>Index of the round key $S_i$ loaded using input $S_i$.</td>
</tr>
<tr>
<td>C</td>
<td>2w</td>
<td>Ciphertext block – Encrypted block M.</td>
</tr>
<tr>
<td>DONE</td>
<td>1</td>
<td>Asserted when ciphertext is ready and available at the output.</td>
</tr>
</tbody>
</table>

Note: $w$ is a size of index $i$. It is a maximum integer, such that $2w = 2v+2$.

Protocol (1)

An external circuit first loads all round keys $S[0], S[1], S[2], ..., S[2r], [2r+1]$ to the two internal memories of the RC5 unit.

The first memory stores values of $S_i=2j$, i.e., only round keys with even indices. The second memory stores values of $S_i=2j+1$, i.e., only round keys with odd indices.

Loading round keys is performed using inputs: $S_i, i, write_Si, clk$.

Then, the external circuits loads a message block $M$ to the RC5 unit, using inputs: $M, write_M, clk$.

After the message block $M$ is loaded to the RC5 unit, the encryption starts automatically.

Protocol (2)

When the encryption is completed, signal Done becomes active, and the output $C$ changes to the new value of the ciphertext.

The output $C$ keeps the last value of the ciphertext at the output, until the next encryption is completed. Before the first encryption is completed, this output should be equal to zero.

Assumptions

- one round of the main for loop of the pseudocode executes in one clock cycle
- you can access only one position of each internal memory of round keys per clock cycle

As a result, the entire encryption of a single message block $M$ should last $r+1$ clock cycles.