Lecture 2B

RTL Design Methodology

Transition from Pseudocode & Interface to a Corresponding Block Diagram
Structure of a Typical Digital System

Datapath (Execution Unit)

Data Inputs

Control & Status Inputs

Datapath (Execution Unit)

Data Outputs

Control & Status Outputs

Control Signals

Status Signals

Controller (Control Unit)
Hardware Design with RTL VHDL

Pseudocode

Datapath
Block diagram
VHDL code

Interface

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 the 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 2
CIPHER
RC6

RC6 (Rivest Cipher 6) - symmetric key block cipher

Designers: Ron Rivest, Matt Robshaw, Ray Sidney, and Yiqun Lisa Yin

One of five finalists of the contest for the Advanced Encryption Standard (AES) 1997-2000

Candidate algorithm in the following other contests:
NIST Report: Security

Security Margin

High

Adequate

Complexity

Simple

Complex

Serpent

MARS

Rijndael

Twofish

RC6
NIST Report: Software Efficiency
Encryption and Decryption Speed

32-bit processors
- RC6
- Rijndael
- Mars
- Twofish
- Low

64-bit processors
- Rijndael
- Twofish
- Mars
- RC6
- Low

DSPs
- Rijndael
- Twofish
- Mars
- RC6
- Low
Efficiency in hardware: FPGA Virtex 1000: Speed

Throughput [Mbit/s]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>George Mason University</th>
<th>University of Southern California</th>
<th>Worcester Polytechnic Institute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serpent I8</td>
<td>431</td>
<td>444</td>
<td>414</td>
</tr>
<tr>
<td>Rijndael</td>
<td>353</td>
<td>414</td>
<td>294</td>
</tr>
<tr>
<td>Twofish</td>
<td>177</td>
<td>173</td>
<td>104</td>
</tr>
<tr>
<td>Serpent I1</td>
<td>149</td>
<td>143</td>
<td>143</td>
</tr>
<tr>
<td>RC6</td>
<td>112</td>
<td>88</td>
<td>61</td>
</tr>
<tr>
<td>Mars</td>
<td>102</td>
<td>61</td>
<td>88</td>
</tr>
</tbody>
</table>
Pseudocode

Split input I into four words, I3, I2, I1, I0, of the size of w bits each

A = I3; B = I2; C = I1; D=I0
B = B + S[0]
D = D + S[1]
for i = 1 to r do
{ 
    T = (B*(2B + 1)) <<< k
    U = (D*(2D + 1)) <<< k
    A = ((A ⊕ T) <<< U) + S[2i]
    C = ((C ⊕ U) <<< T) + S[2i + 1]
    (A, B, C, D) = (B, C, D, A)
}
A = A + S[2r + 2]
C = C + S[2r + 3]
O = (A, B, C, D)
Notation

\( w: \) word size, e.g., \( w=32 \) (constant)
\( k: \) \( \log_2(w) \) (constant)

\( A, B, C, D, U, T: \) \( w \)-bit variables

\( I_3, I_2, I_1, I_0: \) Four \( w \)-bit words of the input \( I \)

\( r: \) number of rounds (constant)

\( O: \) output of the size of \( 4w \) bits

\( S[j] : \) \( 2r+4 \) round keys stored in two RAMs.

Each key is a \( w \)-bit word.

The first RAM stores values of \( S[j=2i], \) i.e., only round keys with even indices. The second memory stores values of \( S[j=2i+1], \) i.e., only round keys with odd indices.
Operations

⊕ : XOR
+ : addition modulo $2^w$
− : subtraction modulo $2^w$
* : multiplication modulo $2^w$

$X <<< Y$ : 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 $Y$

Modular Arithmetic

\[ 11 + 4 \mod 12 = 3 \]
Circuit Interface

clk
reset
I
write_I
Sj
Write_Sj
j

4w

CIPHER

4w

O
DONE

w

m
## Interface Table

<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 internal registers.</td>
</tr>
<tr>
<td>I</td>
<td>4w</td>
<td>Input block.</td>
</tr>
<tr>
<td>write_I</td>
<td>1</td>
<td>Synchronous write control signal for the input block I. After the block I is written to the CIPHER unit, the encryption of I starts automatically.</td>
</tr>
<tr>
<td>Sj</td>
<td>w</td>
<td>Round key S[j] loaded to one of the two internal memories.</td>
</tr>
<tr>
<td>write_Sj</td>
<td>1</td>
<td>Synchronous write control signal for the round key S[j].</td>
</tr>
<tr>
<td>j</td>
<td>m</td>
<td>Index of the round key S[j] loaded using input Sj.</td>
</tr>
<tr>
<td>O</td>
<td>4w</td>
<td>Output block.</td>
</tr>
<tr>
<td>DONE</td>
<td>1</td>
<td>Asserted for one clock cycle when the output O is ready.</td>
</tr>
</tbody>
</table>

**Note:**

m is a size of index j. It is a minimum integer, such that $2^m - 1 \geq 2r+3$. 
Protocol (1)

An external circuit first loads all round keys $S[0], S[1], S[2], \ldots, S[2r+2], [2r+3]$ to the two internal memories of the CIPHER unit.

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

Loading round keys is performed using inputs: $S_j, j, \text{write}_Sj, \text{clk}$.

Then, the external circuits, loads an input block $I$ to the CIPHER unit, using inputs: $I, \text{write}_I, \text{clk}$.

After the input block $I$ is loaded to the CIPHER unit, the encryption starts automatically.
Protocol (2)

When the encryption is completed, signal DONE becomes active, and the output O changes to the new value of the ciphertext.

The output O 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

• 2r+4 clock cycles are used to load round keys to internal RAMs
• 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 encryption of a single input block I should last r+2 clock cycles.
CIPHER: Solutions
Addition mod $2^w$

\[
\begin{array}{c}
X_{w-1} \quad X_0 \\
+ \\
Y_{w-1} \quad Y_0 \\
\hline
S_w \quad S_{w-1} \quad \ldots \quad S_0 \\
\end{array}
\]

\[
2^w \quad 2^{w-1} \quad 2^0
\]

\[
S = S_w \cdot 2^w + S_{w-1} \ldots 0 \mod 2^w = S_{w-1} \ldots 0
\]

XOR

\[
\begin{array}{c}
X_{w-1} \quad X_0 \\
\oplus \\
Y_{w-1} \quad Y_0 \\
\hline
Z_{w-1} \quad Z_0
\end{array}
\]

\[
Z_i = X_i \oplus Y_i \quad i = 0..w-1
\]
Multiplication mod $2^w$

\[ P = P_H \cdot 2^w + P_L \]

\[ P \mod 2^w = P_L \]

\[ P_H = P_{2^{w-1}...0} \]

\[ P_L = P_{w-1...0} \]

\[ Z = X \cdot Y \mod 2^w \]
Variable Rotation $X<<<Y$

\[
X<<<Y = X <<< Y_{k=1...0} \\
\text{where } k = \log_2 W
\]

\[
Y = Y_0 + Y_1 \cdot 2 + Y_2 \cdot 4 + Y_3 \cdot 8 + Y_4 \cdot 16 + Y_5 \cdot 32 + \ldots
\]

\[
\ldots Y_{15} \cdot 2^{15}
\]

\[
X <<< Y
\]

\[
2^{15} \quad 64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1
\]

\[
Y = Y_0 \ldots Y_{15}
\]

\[
\text{where } W = 16
\]

\[
4 = \log_2 16
\]
Datapath – Initialization & Main Loop
Datapath – PostProcessing & Loop Counter
Memories of Round Keys – Contents
Case of m=3 and r=2

\[ m = 3 \Rightarrow r = 2 \]

\[
\begin{array}{c|c|c}
000 & S[0] & 001 \\
010 & S[2] & 011 \\
100 & S[4] & 101 \\
110 & S[6] & 111 \\
\end{array}
\]

\[
\begin{array}{c}
\text{S[1]} \\
\text{S[3]} \\
\text{S[5]} \\
\text{S[7]} \\
\end{array}
\]
Datapath – Memories of Round Keys

[Diagram of a Datapath circuit with memories of round keys, showing signals and components such as Write Sj, DIN, ADDR, RAM, and DOUT.]
Interface with the Division into the Datapath and Controller
Timing Analysis – Key Setup & Encryption

Notation:

\( T_{\text{CLK}} \) – clock period in ns

\( N_M \) – number of encrypted message blocks

\[ T_{\text{KeySetup}}[\text{cycles}] = 2r+4 \]

\[ T_{\text{KeySetup}}[\text{ns}] = (2r+4) \cdot T_{\text{CLK}} \]

\[ T_{\text{Encryption}}(N_M)[\text{cycles}] = (r+2) \cdot N_M \]

\[ T_{\text{Encryption}}(N_M)[\text{ns}] = (r+2) \cdot N_M \cdot T_{\text{CLK}} \]
Timing Analysis - Throughput

Notation:

\( T_{CLK} \) – clock period in ns
\( N_M \) – number of encrypted message blocks
\( w \) – word size

\[
\text{Thr}_{\text{Encryption}}(N_M)[\text{Gbit/s}] = \frac{4 \cdot w \cdot N_M}{(r+2) \cdot N_M \cdot T_{CLK}} = \frac{4 \cdot w}{(r+2) \cdot T_{CLK}}
\]

\( r \) & \( w \) constants, determined by the security analysis of the cipher
For the security equivalent to AES-128, \( r=20 \) & \( w=32 \)

\( T_{CLK} \) dependent on \( w \), returned by the FPGA tools based on the static timing analysis of the critical path
Timing Analysis – Critical Path

\[ T_{CLK-min} = d_R + d_{logic-max} + t_{setup} \]

Critical Path – a path from an output of a register to an input of a register with the maximum value of the delay (denoted as \( d_{logic-max} \))

d_R – delay of a register

t_{setup} – setup time of a register
Setup & Hold Time of a Register

Setup time - the amount of time the data at the synchronous input must be stable before the next active edge of clock.

Hold time - the amount of time the data at the synchronous input must be stable after the last active edge of clock.
Datapath – Critical Path

Critical Path marked in red