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

Control Signals

Status Signals

Data Outputs

Control & Status Outputs

Controller (Control Unit)
Hardware Design with RTL VHDL

Pseudocode

Interface

Datapath

Block diagram

Controller

ASM chart

VHDL code

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 2
CIPHER
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., \( w=8 \) (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

\( S[j] \): 2\( r+4 \) 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 $Y$
Circuit Interface

- clk
- reset
- write_I
- Sj
- Write_Sj
- j

- I
- 4w

- O
- DONE
- 4w
- 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>S_j</td>
<td>w</td>
<td>Round key S[j] loaded to one of the two internal memories.</td>
</tr>
<tr>
<td>write_S_j</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 S_j.</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: $S_j, j, write_S_j, clk$.

Then, the external circuits, loads an input block $I$ to the CIPHER unit, using inputs: $I, write_I, clk$.

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.
Typical Internal Structure of a Secret-Key Block Cipher

- Round Key[0] → Initial Transformation
- Round Key[i] → Cipher Round
- i := 1
- i < #rounds?
- Round Key[#rounds+1] → Final Transformation

#rounds times
Basic iterative architecture
(no Initial Transformation, no Final Transformation)
Basic architecture: Timing

#rounds \cdot \text{clock\_period}
Primary parameters of hardware implementations of secret-key block ciphers

**Latency**
- Time to encrypt/decrypt a single block of data

**Throughput**
- Number of bits encrypted/decrypted in a unit of time

**Encryption/decryption**
- $M_i$
- $C_i$
Advanced Encryption Standard
AES
AES Encryption

Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
  byte state[4,Nb]

  state = in

  AddRoundKey(state, w[0, Nb-1])  // See Sec. 5.1.4

  for round = 1 step 1 to Nr-1
    SubBytes(state)  // See Sec. 5.1.1
    ShiftRows(state)  // See Sec. 5.1.2
    MixColumns(state)  // See Sec. 5.1.3
    AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
  end for

  SubBytes(state)
  ShiftRows(state)
  AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

  out = state
end
AES Decryption

EqInvCipher(byte in[4*Nb], byte out[4*Nb], word dw[Nb*(Nr+1)])
begin
    byte state[4,Nb]

    state = in

    AddRoundKey(state, dw[Nr*Nb, (Nr+1)*Nb-1])

    for round = Nr-1 step -1 downto 1
        InvSubBytes(state)
        InvShiftRows(state)
        InvMixColumns(state)
        AddRoundKey(state, dw[round*Nb, (round+1)*Nb-1])
    end for

    InvSubBytes(state)
    InvShiftRows(state)
    AddRoundKey(state, dw[0, Nb-1])

    out = state
end
Basic Iterative Architecture of AES (Encryption and Decryption)

Encryption circuit

Data input

round key

R1

SubBytes & InvSubBytes

ShiftRows

MixColumns

round key

round key

Decryption circuit

InvShiftRows

InvMixColumns

round key

Data output
Modes of Operation
Block vs. stream ciphers

**Block cipher**

- **Input**: $M_1, M_2, \ldots, M_n$
- **Output**: $C_1, C_2, \ldots, C_n$

\[ C_i = f_K(M_i) \]

Every block of ciphertext is a function of only one corresponding block of plaintext

**Stream cipher**

- **Input**: $m_1, m_2, \ldots, m_n$
- **Output**: $c_1, c_2, \ldots, c_n$

\[ c_i = f_K(m_i, IS_i) \]

\[ IS_{i+1} = g_K(m_i, IS_i) \]

Every block of ciphertext is a function of the current block of plaintext and the current internal state of the cipher
Typical stream cipher

**Sender**
- Key
- Initialization vector (seed)
- Pseudorandom Key Generator
- $k_i$
- Keystream
- $m_i$ plaintext
- $c_i$ ciphertext

**Receiver**
- Key
- Initialization vector (seed)
- Pseudorandom Key Generator
- $k_i$
- Keystream
- $c_i$ ciphertext
- $m_i$ plaintext
### Standard modes of operation of block ciphers

<table>
<thead>
<tr>
<th>Block cipher</th>
<th>Block cipher turned into a stream ciphers</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB mode</td>
<td>Counter mode</td>
</tr>
<tr>
<td></td>
<td>CFB mode</td>
</tr>
<tr>
<td></td>
<td>CBC mode</td>
</tr>
</tbody>
</table>
ECB (Electronic CodeBook) mode
Electronic CodeBook Mode – ECB Encryption

\[ C_i = E_K(M_i) \quad \text{for } i=1..N \]
Electronic CodeBook Mode – ECB
Decryption

\[ C_i = E_K(M_i) \text{ for } i=1..N \]
Counter Mode
Counter Mode - CTR Encryption

c_i = m_i \oplus k_i
k_i = E_K(IV+i-1) \quad \text{for } i=1..N
Counter Mode - CTR
Decryption

\[ \begin{align*}
\text{for } i &= 1 \ldots N \\
\text{c}_i &= \oplus \text{k}_i \\
\text{k}_i &= E_K(\text{IV} + i - 1) \\
\text{m}_i &= \text{c}_i \oplus \text{k}_i
\end{align*} \]
Counter Mode – CTR
(simplified block diagram)

IV → counter
  ↓
  IS_i

IN → E
  ↓
  c_i
  ↓
m_i

OUT → c_i

K → E
  ↓
  c_i
  ↓
m_i

IV → counter
  ↓
  IS_i

K → E
  ↓
  c_i
  ↓
m_i

IS_1 = IV

\[ c_i = E_K(IS_i) \oplus m_i \]

\[ IS_{i+1} = IS_i + 1 \]
Counter Mode – Potential for Parallel Processing

\[ C_i = M_i \oplus AES(IV+i) \quad \text{for } i=0..N \]
Increasing speed by parallel processing

Encryption/decryption unit

Encryption/decryption unit

Encryption/decryption unit

Encryption/decryption unit

Encryption/decryption unit

Encryption/decryption unit
Increasing speed using pipelining

 Cipher 1

 Cipher 2

 Throughput = \frac{\text{block size}}{\text{target_clock_period}}

 target clock period, e.g., 20 ns
CFB (Cipher FeedBack) Mode
Cipher Feedback Mode - CFB

Encryption

\[ c_i = m_i \oplus k_i \]
\[ k_i = E_K(c_{i-1}) \quad \text{for } i=1..N, \text{ and } c_0 = IV \]
Cipher Feedback Mode - CFB Decryption

\[ m_i = c_i \oplus k_i \]

\[ k_i = E_K(c_{i-1}) \quad \text{for } i=1..N, \text{ and } c_0 = IV \]
Cipher Feedback Mode – CFB
(simplified block diagram)

IV

register

IS_i

IN

E

OUT

m_i

c_i

K

E

IS_1 = IV

c_i = E_K(IS_i) \oplus m_i

IS_{i+1} = c_i

register

IV

IN

E

OUT

m_i
CBC (Cipher Block Chaining) Mode
Cipher Block Chaining Mode - CBC Encryption

\[ c_i = E_K(m_i \oplus c_{i-1}) \quad \text{for } i=1..N \quad c_0=\text{IV} \]
Cipher Block Chaining Mode - CBC Decryption

\[ m_i = D_K(c_i) \oplus c_{i-1} \text{ for } i=1..N \quad c_0=\text{IV} \]