



VHDL

System ASIC Design Lab.

:

jcho@asiclab.inchon.ac.kr



Chap. 1

Chap. 2

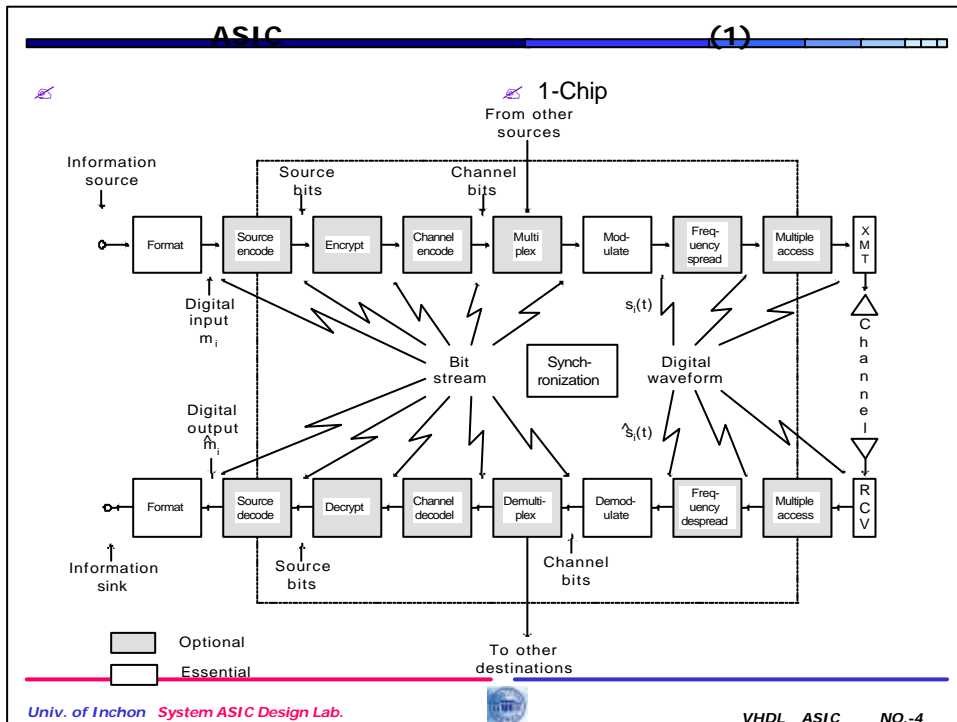
Chap. 3 Digital System VHDL Modeling

Chap. 4 Test Bench Model





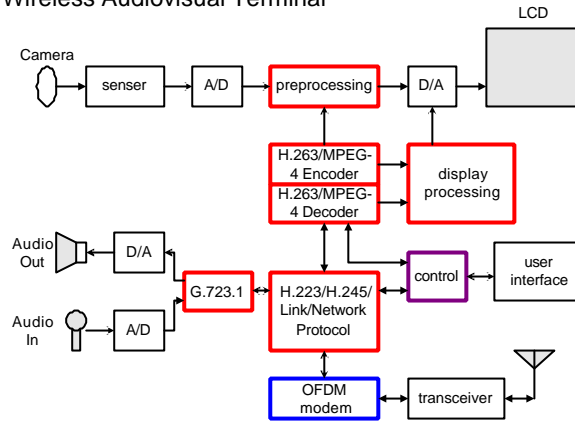
Chap. 1



ASIC

(2)

Wireless Audiovisual Terminal

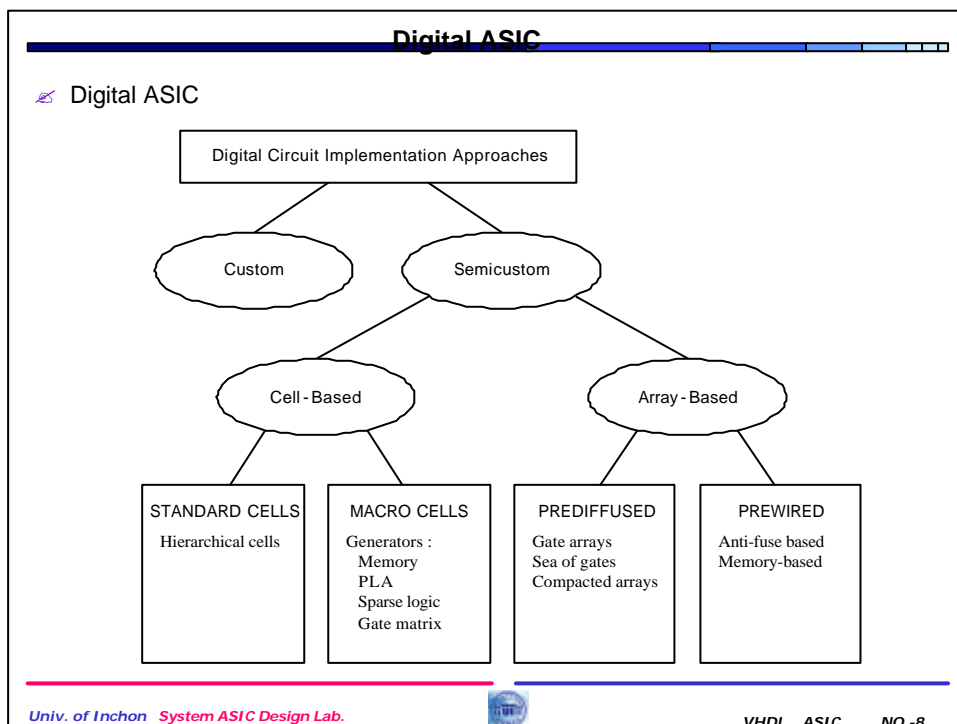
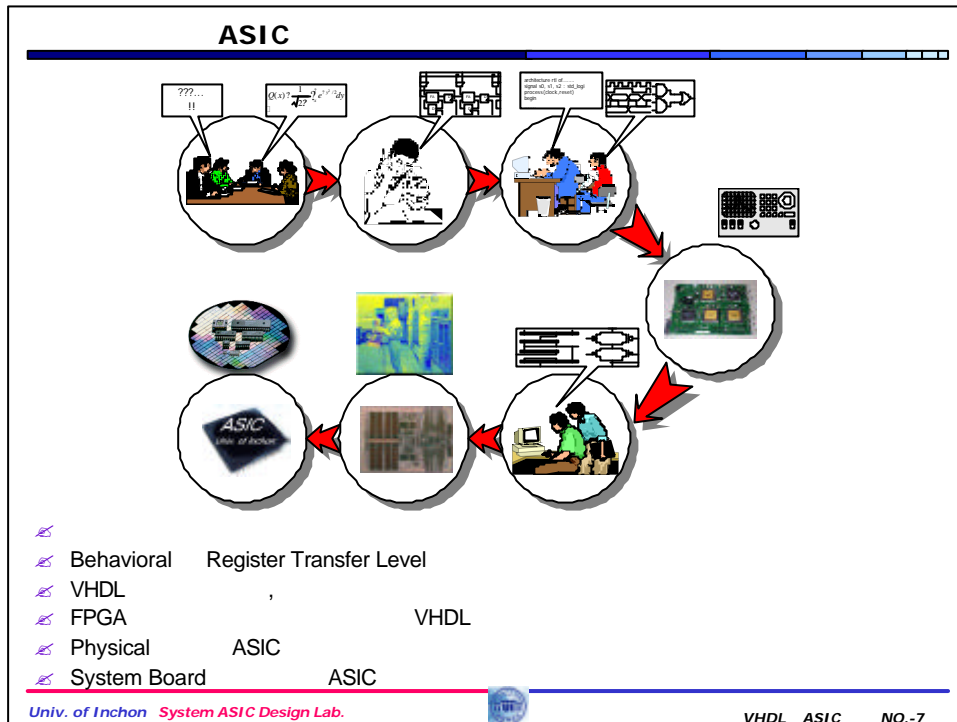


- ✍ H.324 CODEC : Video Audio Data
- ✍ Digital Communication Module : Data , Error
- ✍ Micro-Controller : 16-Bit Processor Camera, LCD Interface Chips
- ✍ Analog Front End Module : A/D, D/A

- ✍ System Level
- ✍ Behavioral Level
- ✍ RTL Level
- ✍ Logic Level
- ✍ Circuit Level
- ✍ Layout Level

Level	Graphical	Representation
System Partitioning, Pipelining		HDL, Graphical specifications
Behavioral Scheduling, Allocation		HDL, Bubble diagrams
RTL State Machines, Logic and RTL Multi-level, 2-level Optimization		Logic equations, HDL, Gates
Logic Technology mapping/translation		Layout CIF/GDSII
Implementation Place and Route		

- ✍ Graphic Diagram
- ✍ (HDL : Hardware Description Language)



Digital ASIC

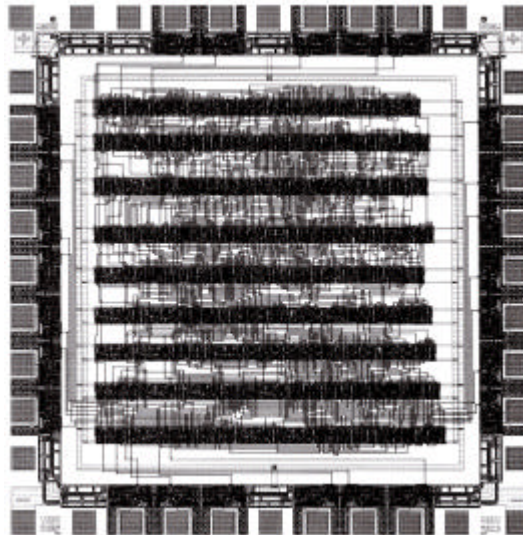
ASIC

	<i>Custom</i>	<i>Cell-based</i>	<i>Prediffused</i>	<i>Prewired</i>
Density	Very High	High	High	Medium - Low
Performance	Very High	High	High	Medium - Low
Flexibility	Very High	High	Medium	Low
Design time	Very Long	Short	Short	Very Short
Manufacturing time	Medium	Medium	Short	Very Short
Cost - low volume	Very High	High	High	Low
Cost - high volume	Low	Low	Low	High



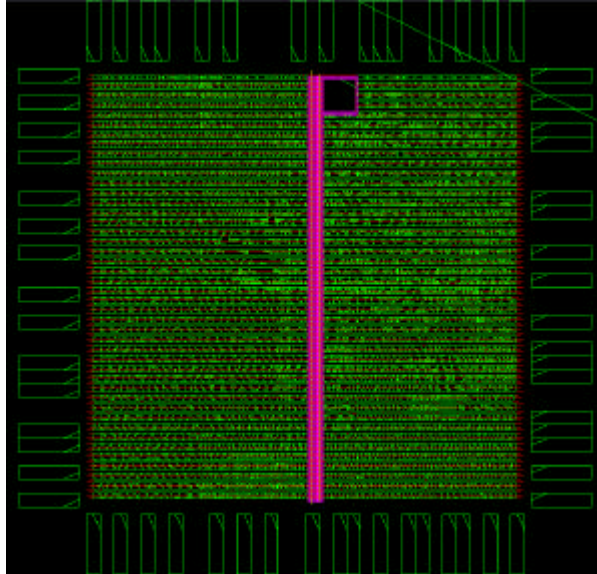
Standard-Cell Design

Boding Diagram



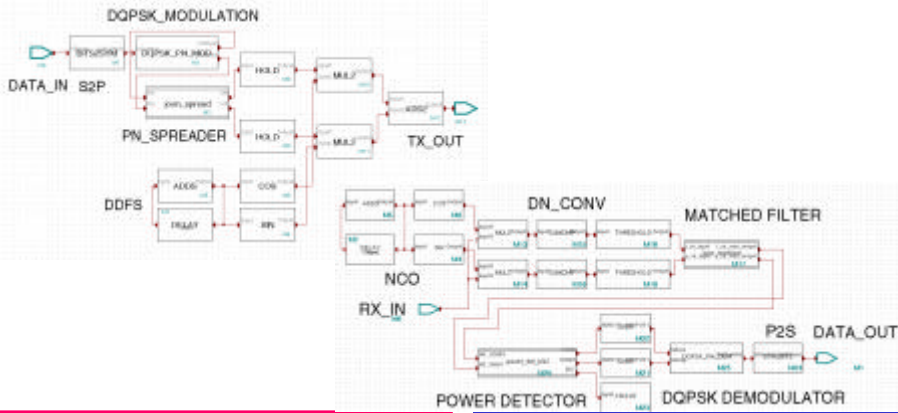
Sea-of-Gate Design

✍ Boding Diagram : FLEX Decoder Chip



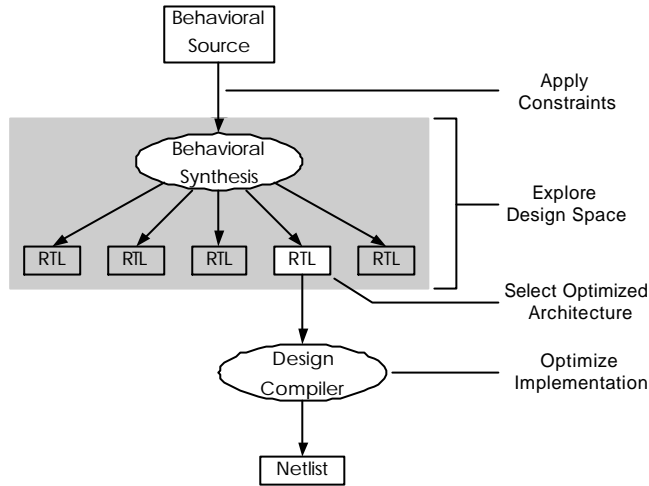
-1 : System(Algorithm) Level

- ✍ C, MATLAB, SPW COSSAP System Level
- ✍ library sub-block
- ✍ system (, BER)
- ✍ hardware complexity performance data-bus bit size
- ✍ System Chip
- ✍ COSSAP DS/SS Base Band MODEM Modeling



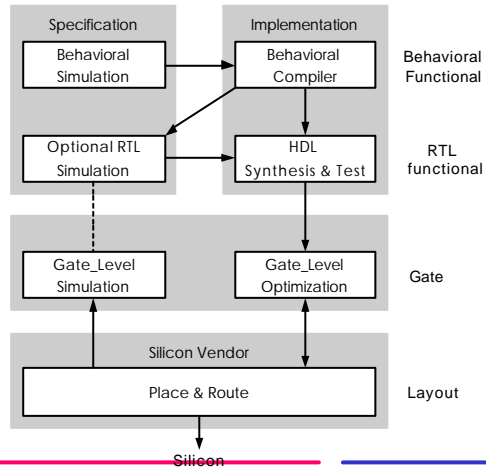
-2 : Behavioral Compiler

- System Level
- Behavioral Compiler
- 가 RTL
- Option
- RTL
- RTL



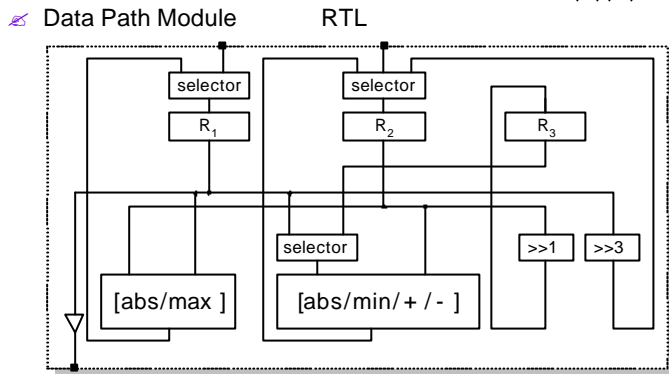
-2 : Behavioral Compiler Design Flow

- Behavioral Compiler
- Design Flow
- Operator to State Binding
- Variable to Register Binding
- Function Unit Sharing
- Control Logic
- FSM



-2 : Behavioral Compiler RTL

- ✂
- ✂
- ✂ 가 $\sqrt{a^2 + b^2} + \max((0.875x + 0.5y), x)$
 $x + \max(|a|, |b|)$
 $y + \min(|a|, |b|)$

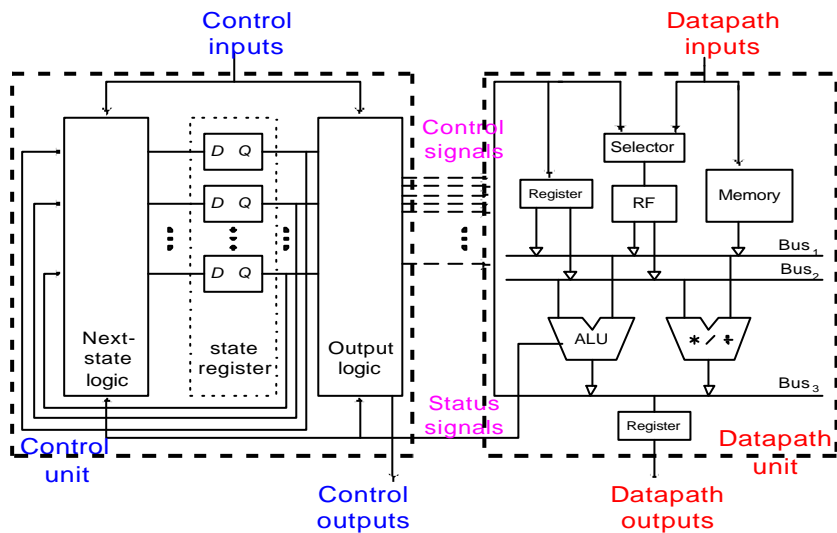


- ✂ Controller Module RTL
- ✂ FSM Table Model



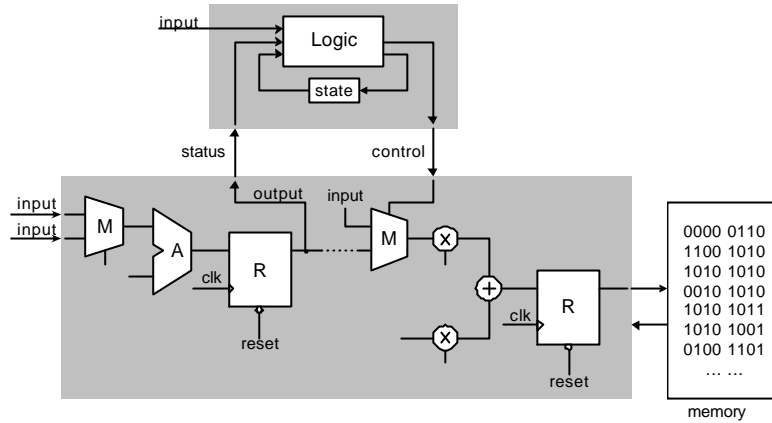
-3 : Register Transfer Level

- ✂ RTL Target Architecture



- 3 : Register-Transfer Level

☞ Synthesis 가 VHDL Hardware



☞ Target Architecture

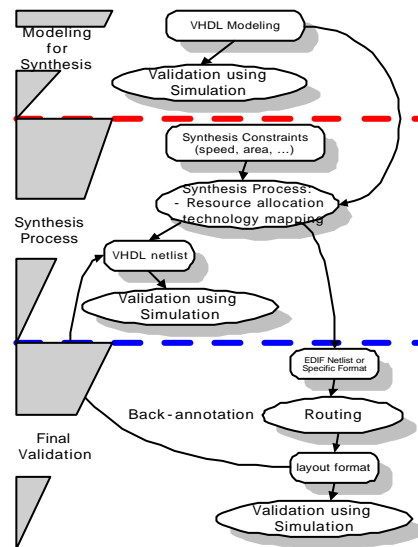
- ☞ : Basic Gates, Modular Units (mux, decoder, arithmetic units)
- ☞ : F/Fs, Registers, Counters, FSM(Finite State Machine)
- ☞ : ROM, RAM
- ☞ Bus : 3-Buffer

3 : Design Compiler

☞ Design Flow 3

- ☞ RTL
 - ☞ VHDL
 - ☞ Simulation
 - ☞ Gate Level
 - ☞ , ,Power
 - ☞ Design Library : FPGA, ASIC
 - ☞ Component Delay
 - ☞ Timing Simulation
 - ☞ Physical Design EDIF
- ☞ FPGA Physical Design
 - ☞ FPGA Tool Compile
 - ☞ Device Option
 - ☞ Layout Data Simulation
 - ☞ Target Board Emulation
- ☞ ASIC Physical Design
 - ☞ Design House EDIF
 - ☞ Wire Delay Simulation
 - ☞ Sign-off ASIC
 - ☞ Test Board 가

☞ Design Flow



HDL VHDL

✂ HDL VHDL
 ✂ VHDL : , synthesis sub-set
 VHDL modeling guide가 .
 ✂ Verilog : VHDL , gate
 simulation design kit golden simulator

✂ VHDL

✂ Value

✂ 1987	STD-1076 Version	
✂ STD library	2-Value	{'0', '1'}
✂ 1990	Value	
✂ IEEE library	9-value	{'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-'}

Univ. of Incheon System ASIC Design Lab. VHDL ASIC NO.-19

Design

✂ Design Entity

✂ Hardware

✂ :
 NOT gate, ALU, 8051 Micro-Controller, DSP Board, Digital TV System

✂ VHDL Design Entity

✂ : entity declaration design unit
 ✂ : architecture body design unit
 ✂ entity unit architecture unit : configuration design unit


Entity declaration
+ associated architectures

architecture 1

architecture 2


Univ. of Incheon System ASIC Design Lab. VHDL ASIC NO.-20

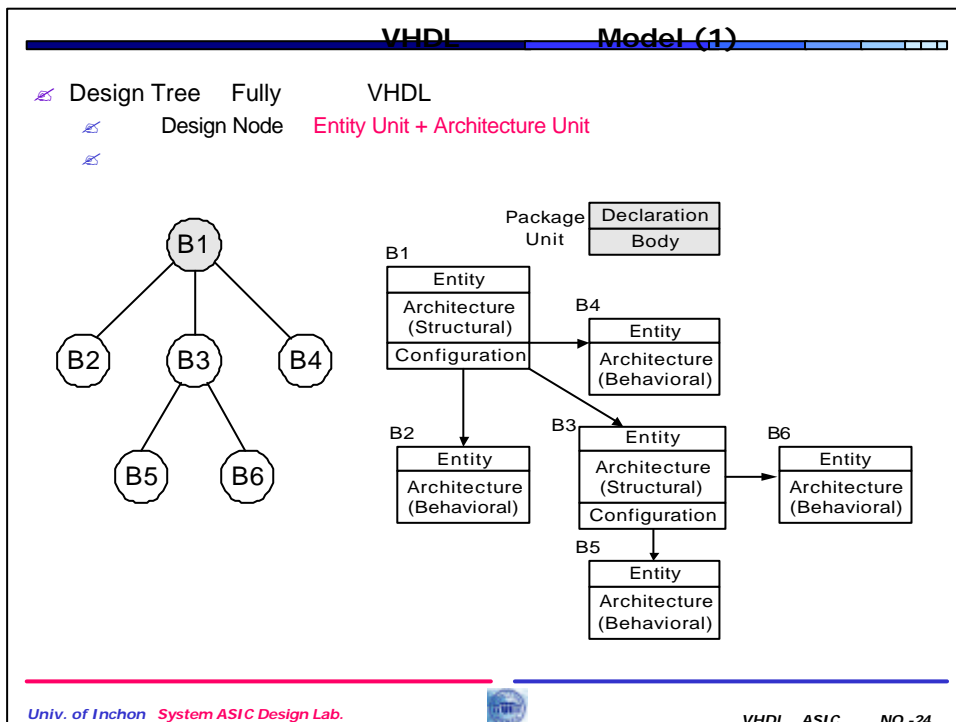
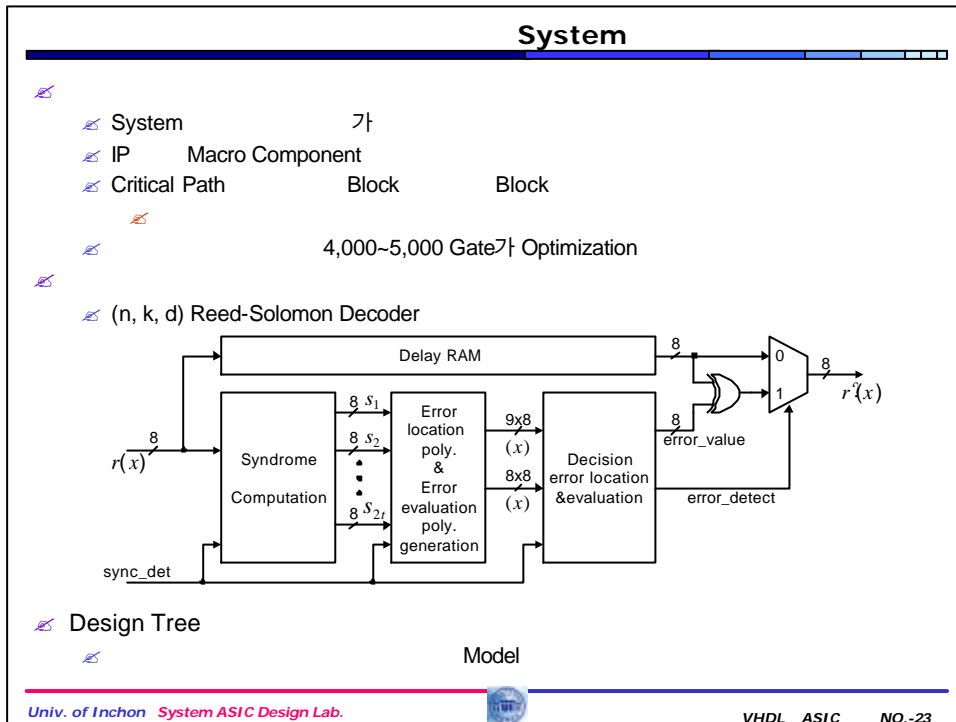
Design Entity
 Subprogram, , Component
 VHDL
 Interface
 Package Declaration Design Unit
 Sub-Program Algorithm
 Package Body Design Unit
 Package declaration + associated package
 package declaration
 package body
 Subprogram headers (declarations)
 algorithms of the subprograms

Univ. of Incheon System ASIC Design Lab.  VHDL ASIC NO.-21

VHDL 57 Design Unit

Hardware 37 Unit
 entity declaration unit
 interface
 architecture body unit
 configuration declaration unit
 entity unit architecture unit
 simulation
 Software 27 Unit
 package declaration unit
 design entity
 package body unit
 subprogram
 27 Unit
 entity unit
 9-Value Package Visibility
 I/O
 architecture unit
 : Behavioral View Model
 : Structural View Model

Univ. of Incheon System ASIC Design Lab.  VHDL ASIC NO.-22

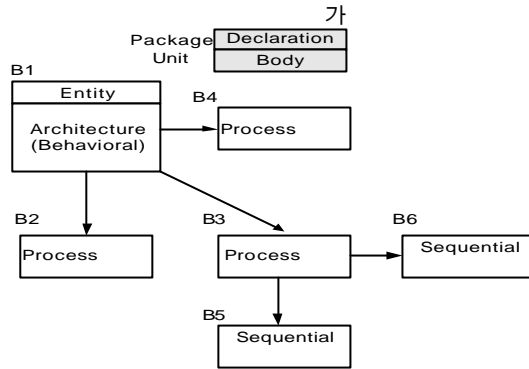
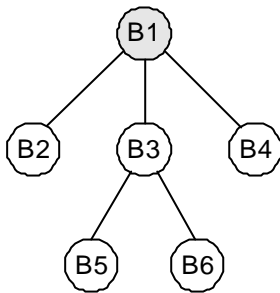


VHDL Model (2)

Design Tree 1-Level VHDL

- Root Design Node
- Root Design Node Concurrent VHDL

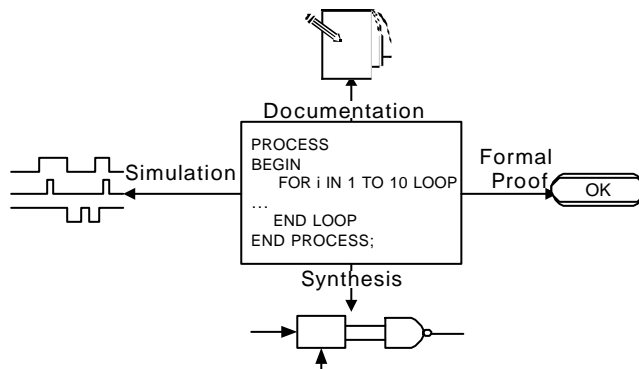
Entity Unit + Architecture Unit



- 가 : (B3 Block Critical Path Behavioral View Model)

VHDL

- System : Simulation Documentation
- Chip : Simulation, Synthesis Documentation
- Chip
 - IP(Intellectual Property) Model Re-Use 가
 - Synthesis가 가 VHDL Modeling 가



Full Adder

2가

VHDL

1-Level VHDL

```

library IEEE;
use IEEE.std_logic_1164.all;
entity FULL_ADD is
port (
  A          : in std_logic;
  B          : in std_logic;
  C_IN       : in std_logic;
  C_OUT      : out std_logic;
  SUM       : out std_logic);
end FULL_ADD;
architecture RTL1 of FULL_ADD is
  signal S0 : std_logic;
begin
  S0 <= A xor B;
  SUM <= S0 xor C_IN;
  C_OUT <= (A and B) or (S0 and C_IN);
end RTL1;
  
```



VHDL

```

library IEEE; use IEEE.std_logic_1164.all;
entity FULL_ADD is
  port ( A, B, C_IN : in std_logic;
         SUM, C_OUT : out std_logic);
end FULL_ADD;
architecture RTL2 of FULL_ADD is
  signal S0,S1,S2 : std_logic;
  for u0,u1 : HA use entity work.HA(RTL);
  for u2 : OR2 use entity work.OR2(RTL);
  component HA
    port ( A, B : in std_logic;
          SUM, CARRY : out std_logic);
  end component;
  component OR2
    port ( IN1, IN2 : in std_logic;
          OUT1 : out std_logic);
  end component;
begin
  u0 : HA port map (X, Y, S0, S1);
  u1 : HA port map (S0, C_IN, SUM, S2);
  u2 : OR2 port map (S1, S2, C_OUT);
end RTL2;
  
```

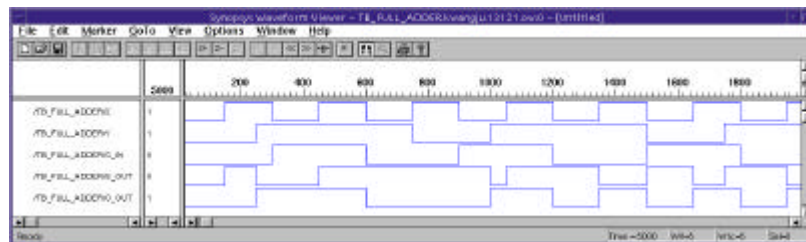


Simulation Synthesis

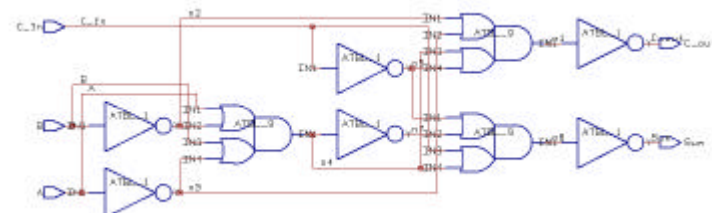
(1)

Simulation

Waveform



1-Level VHDL



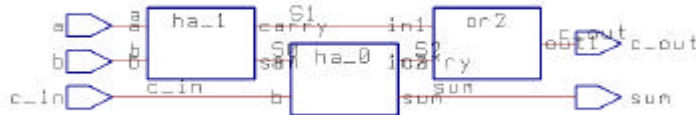
LSI-10K Library : 3.0



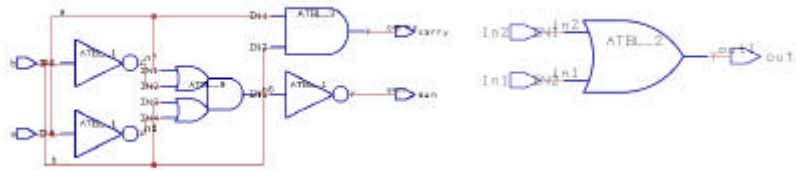
Simulation Synthesis (2)

VHDL

Structural View Node



Behavioral View Node (HA: Half Adder, OR2: 2-input OR)



LSI-10K Library : $1.5 * 2 + 0.5 = 3.5$

Optimization



VHDL

```

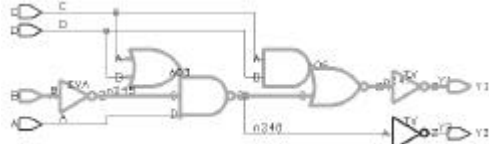
library IEEE;
use IEEE.std_logic_1164.all;
entity EX is
    port (A, B, C, D : in std_logic;
          Y1, Y2 : out std_logic);
end EX;
architecture RTL of EX is
begin
    process (A, B, C, D)
        variable TMP:
            std_logic_vector(3 downto 0);
    begin
        TMP := A & B & C & D;
        case TMP is
            when "0000" => Y1<='1'; Y2<='0';
            when "0001" => Y1<='1'; Y2<='0';
            when "0010" => Y1<='1'; Y2<='0';
            when "0011" => Y1<='1'; Y2<='0';
            when "0100" => Y1<='1'; Y2<='0';
            when "0101" => Y1<='1'; Y2<='0';
            when "0110" => Y1<='1'; Y2<='0';
            when "0111" => Y1<='1'; Y2<='0';
            when "1000" => Y1<='1'; Y2<='0';
            when "1001" => Y1<='0'; Y2<='1';
            when "1010" => Y1<='0'; Y2<='1';
            when "1011" => Y1<='1'; Y2<='1';
            when "1100" => Y1<='1'; Y2<='0';
            when "1101" => Y1<='1'; Y2<='0';
            when "1110" => Y1<='1'; Y2<='0';
            when "1111" => Y1<='1'; Y2<='0';
            when others => Y1<='1'; Y2<='0';
        end case;
    end process;
end RTL;
    
```



Target Library : LSI_10K

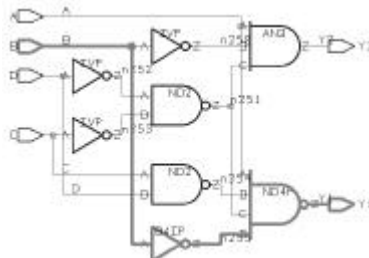
Area Area Design Constraint

5-cells(Area : 7.0), Critical Path(2.38ns)



Timing Design Constraint

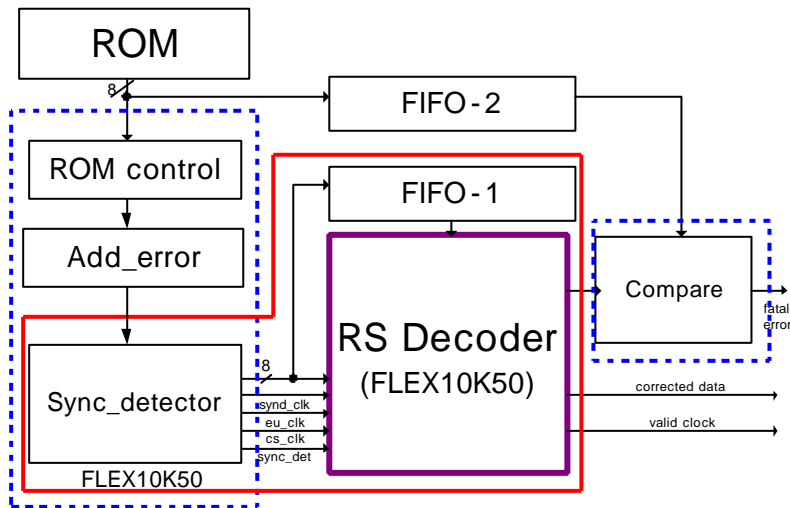
8-cells(Area : 15.0), Critical Path(0.84ns)



FPGA Fast Prototyping

FPGA Board

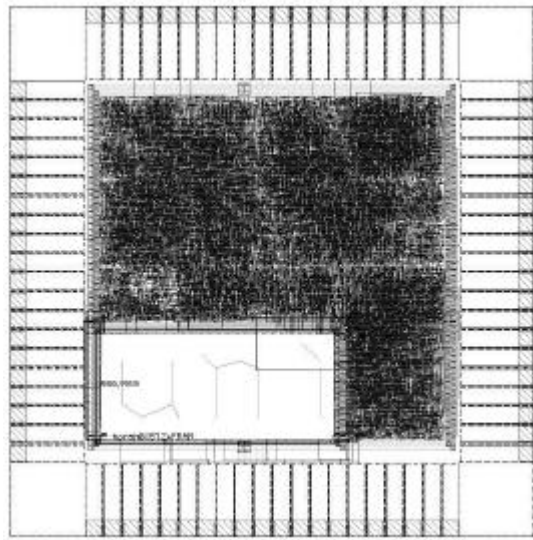
Reed-Solomon Decoder



ASIC

Reed-Solomon Decoder

Bonding Diagram

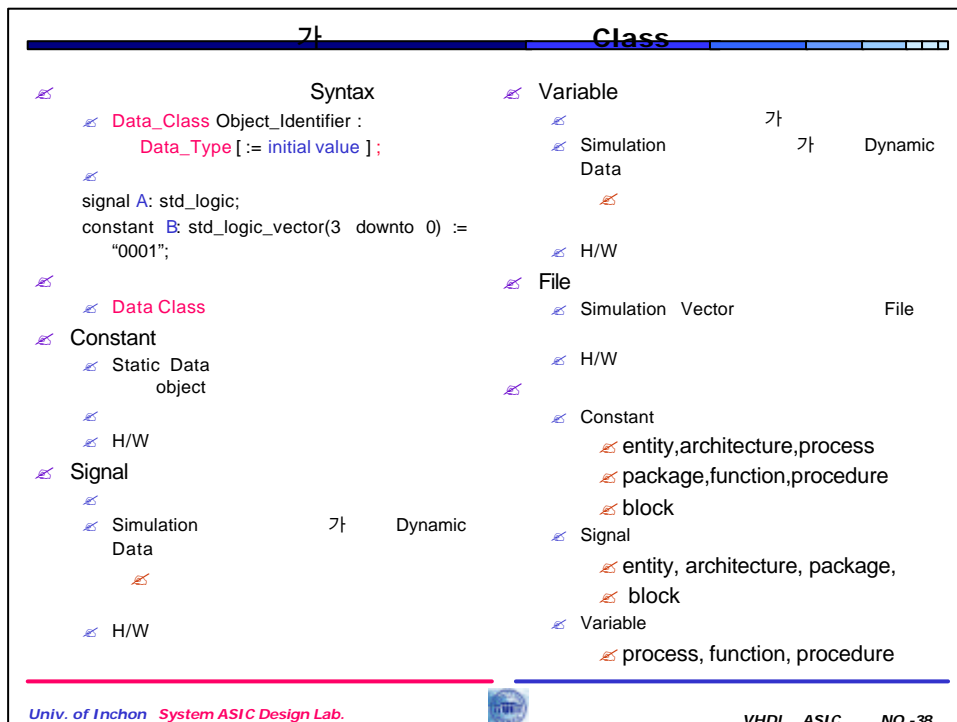
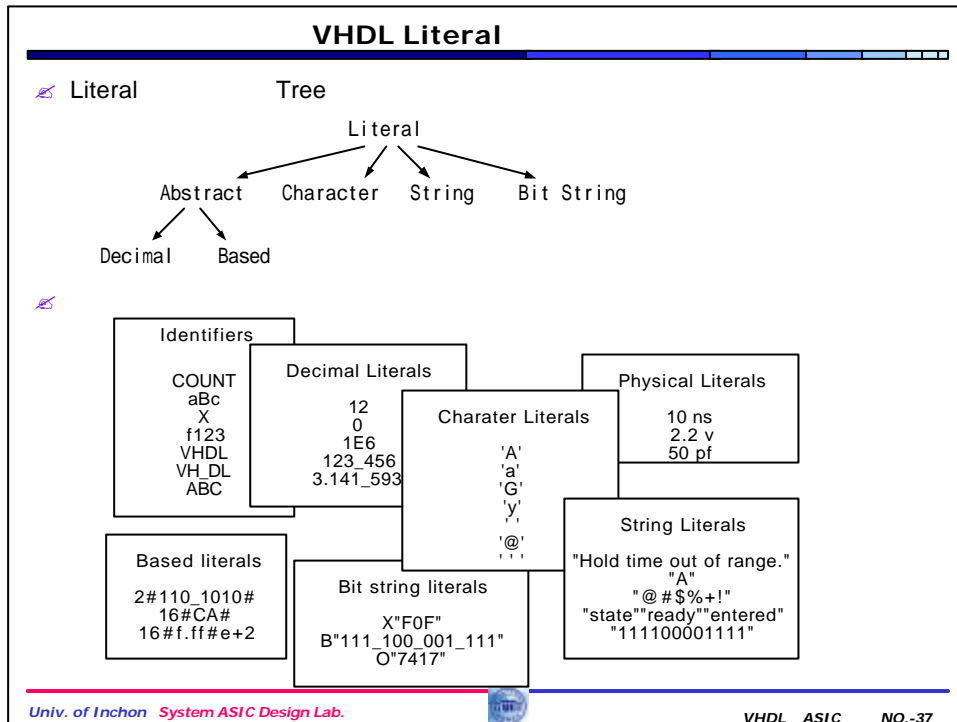


Chap. 2 VHDL



Character Set , Lexical Elements		Identifier	
<ul style="list-style-type: none"> ⌘ Character Sets <ul style="list-style-type: none"> ⌘ Upper-Case Letters ⌘ Lower-Case Letters ⌘ Special Characters ⌘ Space Character ⌘ Format Effectors ⌘ Lexical Elements <ul style="list-style-type: none"> ⌘ Delimiter <ul style="list-style-type: none"> ⌘ Single Character ⌘ Compound Character ⌘ Identifier <ul style="list-style-type: none"> ⌘ Reserved Identifier ⌘ 가 Identifier ⌘ Literal 	<ul style="list-style-type: none"> ⌘ VHDL-87 <ul style="list-style-type: none"> abs and assert buffer configuration downto entity function in library nand null others procedure record return subtype units when access architecture attribute bus constant else exit generate inout linkage new of out process register select then until while 	<ul style="list-style-type: none"> Reserved Word <ul style="list-style-type: none"> alias all array block body component disconnect end for guarded if label map mod nor not open or port range report signal severity transport type variable wait xor 	

VHDL-93 가 Reserved Word		Identifier	
<ul style="list-style-type: none"> ⌘ VHDL-93 가 Reserved Word <ul style="list-style-type: none"> group impure inertial literal postponed pure reject rol ror shared sla sll sra srl unaffected xnor ⌘ 가 Identifier <ul style="list-style-type: none"> ⌘ Alphabet ⌘ Alphabet ⌘ Alphabet, ⌘ ,_ ⌘ Alphabet ⌘ COUNT <ul style="list-style-type: none"> cout c_out AB2_5C VHSIC X1 FFT ⌘ Decoder <ul style="list-style-type: none"> A_B_C xyZ h333 STORE_NEXT_ITEM ⌘ 2CA <ul style="list-style-type: none"> My-name H\$B LOOP Decode_ N#3 			



Data

<ul style="list-style-type: none"> ☞ Data <ul style="list-style-type: none"> ☞ Data Type <ul style="list-style-type: none"> ☞ Group, Package, Data Type ☞ VHDL 가 Data Type ☞ : ** Type Synthesis ☞ Tool <ul style="list-style-type: none"> ☞ Scalar Data Types <ul style="list-style-type: none"> ☞ Enumeration Type, Numeric type, Physical Type** ☞ Composite Data Types <ul style="list-style-type: none"> ☞ Array Type, Record Type ☞ Access Data Type** ☞ File Data Type** ☞ Type package <ul style="list-style-type: none"> ☞ standard package <ul style="list-style-type: none"> ☞ 1987 ☞ std_logic_1164 ☞ 1993 	<ul style="list-style-type: none"> ☞ Standard Package <pre> package STANDARD is type BOOLEAN is (FALSE, TRUE); type BIT is ('0', '1'); type CHARACTER is (); type SEVERITY_LEVEL is (NOTE, WARNING, ERROR, FAILURE); type INTEGER is <i>implementation_defined</i>; type REAL is <i>implementation_defined</i>; type TIME is range <i>implementation_defined</i>; units fs: ps = 1000 fs; end units; subtype DELAY_LENGTH is TIME; function NOW return time; subtype NATURAL is integer range 0 to integer'high; subtype POSITIVE is integer range 1 to integer'high; type STRING is array(positive range <>) of character; type BIT_VECTOR is array(natural range <>) of bit; end STANDARD;</pre>
---	---

Univ. of Incheon System ASIC Design Lab. VHDL ASIC NO.-39

1993 Package Type

<ul style="list-style-type: none"> ☞ Std_logic_1164 Package <pre> package STD_LOGIC_1164 is type STD_ULOGIC is ('U', --Uninitialized 'X', --Forcing Unknown '0', --Forcing 0 '1', --Forcing 1 'Z', --High Impedance 'W', --Weak Unknown 'L', --Weak 0 'H', --Weak 1 '-' --Don't care); type STD_ULOGIC_VECTOR is array(natural range <>) of std_ulogic; function resolved(s: std_ulogic_vector) return std_ulogic; subtype STD_ULOGIC is resolved std_ulogic;</pre> 	<pre> type STD_LOGIC_VECTOR is array(natural range <>) of std_ulogic; subtype X01 is resolved std_ulogic range 'X' to '1'; subtype X01Z is resolved std_ulogic range 'X' to 'Z'; subtype UX01 is resolved std_ulogic range 'U' to '1'; subtype UX01Z is resolved std_ulogic range 'U' to 'Z'; function "and" (l: std_ulogic; r: std_ulogic) return std_ulogic; ; function "and" (l: std_ulogic_vector; r: std_ulogic_vector) return std_ulogic_vector; ; function is_X (s: std_ulogic) return boolean; end STD_LOGIC_1164;</pre>
---	--

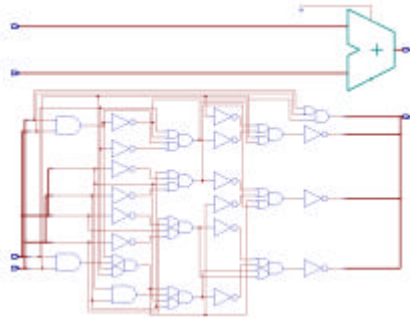
Univ. of Incheon System ASIC Design Lab. VHDL ASIC NO.-40

가 Data Type (1)

Integer Data Type

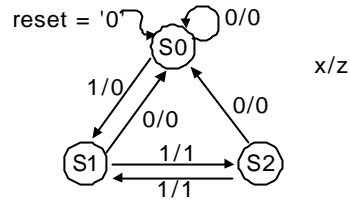
- type **COUNTVALUE** is range 0 to 15;
- type **TWENTIES** is range 20 to 29;
- Synthesis tool bit 0

- Integer
- 0 ~ 15



Enumeration Data Type

- FSM
- type **STATE_NAME** is (A, B, C, D);
- 2 가 (0)
- element 1 가



- type **STATE** is (S0, S1, S2);
- signal C_STATE, N_STATE : **STATE**;

가 Data Type (2)

8*5 Bit 2 Memory : -1

- Word Addressing Mode가 Memory
- 1
- subtype **WORD1** is
- std_logic_vector(4 downto 0);
- type **MEMORY1** is array(0 to 7) of
- WORD;

- 2
- type **WORD2** is array(4 downto 0) of
- std_logic;
- type **MEMORY2** is array(0 to 7) of
- WORD;

- 3
- type **MEMORY3** is array(0 to 7) of
- std_logic_vector(4 downto 0);

8*5 Bit 2 Memory : -2

- Bit Addressing Mode
- type **MEMORY4** is
- array(0 to 7, 4 downto 0) of std_logic;
- Bit-addressing Mode가 Memory 2

Composite Record Data Type

- type **FloatPointType** is
- record
- SIGN : std_logic;
- EXPONENT : unsigned(0 to 6);
- FRACTION : unsigned(24 downto 1);
- end record;

Composite Array Data Type

- type **unsigned** is array(natural range <>) of std_logic;
- type **unsigned_word** is array(natural range 7 downto 0) of std_logic;

Entity Declaration Unit

Entity Unit

Generic (static information: parameters)

Generic

Port(dynamic information)

Port

in

out

inout

buffer

Subtype constraint checks on generics
(values must belong to range) using passive process such as assertions

Dynamic checks on ports
(sequence of values must match arbitrary constraints - - set-up, hold)

Syntax

```
entity ENTITY_NAME is
  [ generic ( LIST_OF_GENERICS_AND_THEIR_TYPES ); ]
  [ port ( LIST_OF_PORTS_AND_THEIR_MODE ); ]
  [ DECLARATIONS ]
  [ begin
    { ENTITY_STATEMENT } ]
end [ ENTITY_NAME ] ;
```

Univ. of Incheon System ASIC Design Lab.
VHDL ASIC NO.-43

Out Mode Buffer Mode

Out Mode

out mode

Buffer Mode

buffer mode

Buffer Mode

Buffer Mode

Buffer Mode

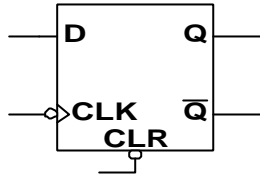
Buffer Mode

Buffer Mode Out Mode

Univ. of Incheon System ASIC Design Lab.
VHDL ASIC NO.-44

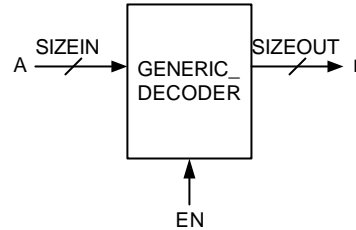
Entity Declaration Unit

D Flip-Flop Entity Unit



```
-- 9-value Package      Visibility
library IEEE; use IEEE.std_logic_1164.ALL;
-- Pin Name, Mode, Data Type
entity DFF is
port (D : in std_logic;
      CLK : in std_logic;
      CLR : in std_logic;
      -- D, CLK, CLR : in std_logic;
      Q, QBAR : out std_logic);
end DFF;
```

Generic Decoder Entity Unit

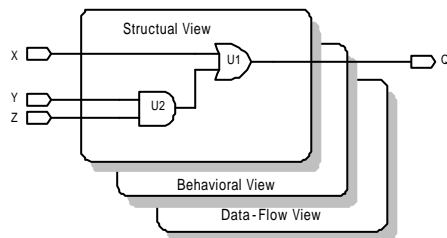


```
library IEEE; use IEEE.std_logic_1164.ALL;
entity GENERIC_DECODER is
-- Static Information
generic (SIZEIN, SIZEOUT : integer);
-- Dynamic Pin Data
port (EN: in std_logic;
      A: in std_logic_vector(SIZEIN-1
                             downto 0);
      B : out std_logic_vector(SIZEOUT-1
                              downto 0));
end GENERIC_DECODER;
```

Architecture Body Unit

Syntax

```
architecture A_NAME of E_NAME is
[ DECLARATIONS ]
begin
{CONCURRENT_STATEMENT}
end [A_NAME];
```



Behavioral view

```
Q='1'
x='1' , Y Z가 '1'
```

Data-flow view

```
Boolean state equation
```

Structural view

```
and gate or gate
```

```
and or gate Behavioral View
Data-flow View
```

Mixed view

```
Behavioral View
Data-flow View
Structural View
```

```
가
```

Architecture Body Unit

Entity design unit

```
library IEEE; use IEEE.std_logic_1164.all;
entity EXAMPLE is
  port (X, Y, Z: in std_logic;
        Q: out std_logic);
end EXAMPLE;
```

Behavioral View

```
architecture RTL1 of EXAMPLE is
begin
  process (X, Y, Z)
  begin
    if X='1' then
      Q <= '1';
    elsif (Y='1' and Z='1') then
      Q <= '1';
    else
      Q <= '0';
    end if;
  end process;
end RTL1;
```

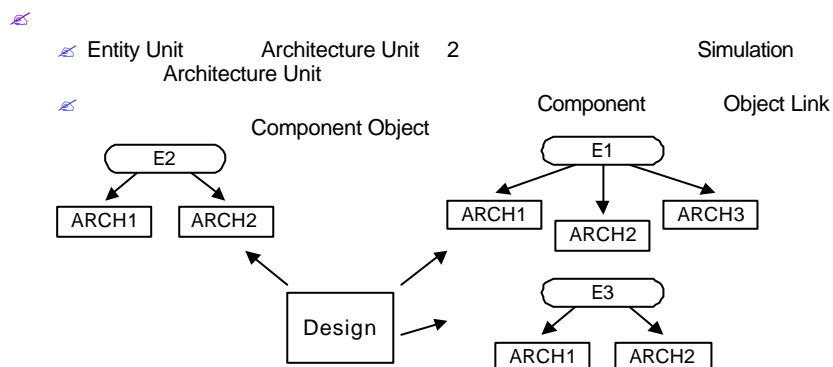
Data-flow View

```
architecture RTL2 of EXAMPLE is
begin
  Q <= X or (Y AND Z);
end RTL2;
```

Structural View

```
architecture RTL3 of EXAMPLE is
  component OR2
    port (I1, I2: in std_logic;
          O1: out std_logic);
  end component;
  component AND2
    port (I1, I2: in std_logic;
          O1: out std_logic);
  end component;
  for U0: OR2 use entity work.OR2(RTL);
  for U1: AND2 use entity work.AND2(RTL);
  signal S1: std_logic;
begin
  U0: OR2 port map (X, S1, Q);
  U1: AND2 port map (Y, Z, S1);
end RTL3;
```

Configuration



- Entity Unit EXAMPLE 37
- Simulation Entity Unit
- Architecture Unit
 - Configuration Specification
 - Configuration Unit
 - Configuration Declaration Unit

Configuration Specification :

- ✍ Architecture Unit Structural View
- ✍ component compile library entity unit
- architecture unit component binding
- ✍ 1-Level
- ✍ **architecture unit**
- ✍ Syntax
- for INSTANTIATION_LIST : COMPONENT_NAME
- [**use entity** ENTITY_NAME { (ARCHITECTURE_NAME) }]
- [**generic map** (GENERIC_ASSOCIATION_LIST)]
- [**port map** (PORT_ASSOCIATION_LIST)] ;

- ✍
- ✍ for U0 : NAND1, NAND2 **use entity** work.NAND_GATE(RTL);
- ✍ 가
- ✍ for U1 : INV1, INV2, INV3 **use entity** work.INVERTER;
- ✍ for ALL : NOR **use configuration** LIB1.NOR_C;
- ✍ NOR component 가 compile library LIB1
- ✍ for others : NOT **use entity** work.NOT(RTL);



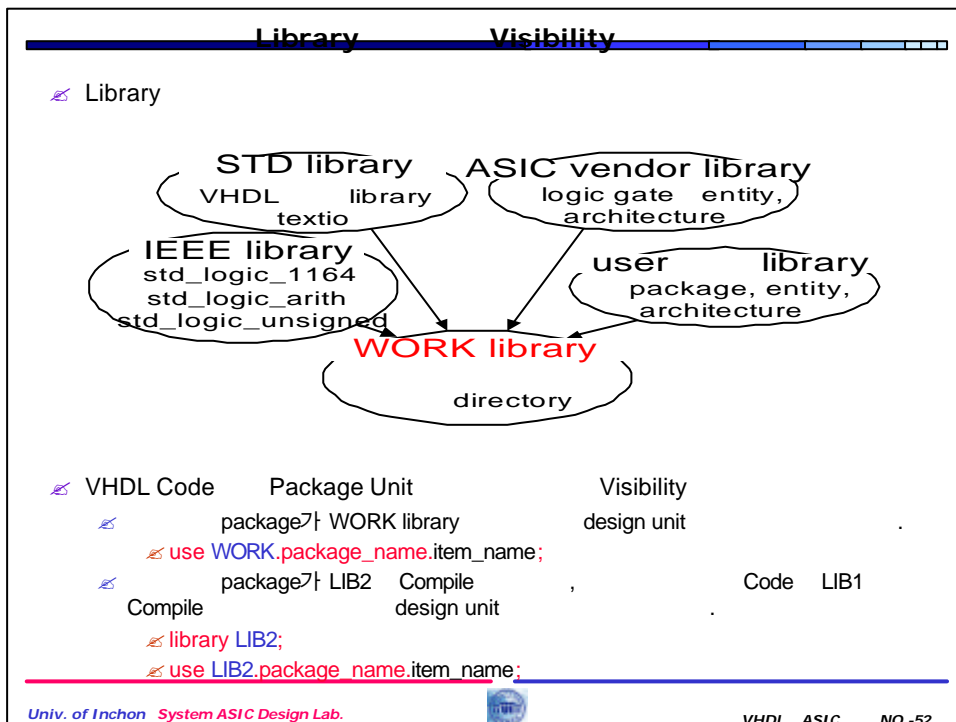
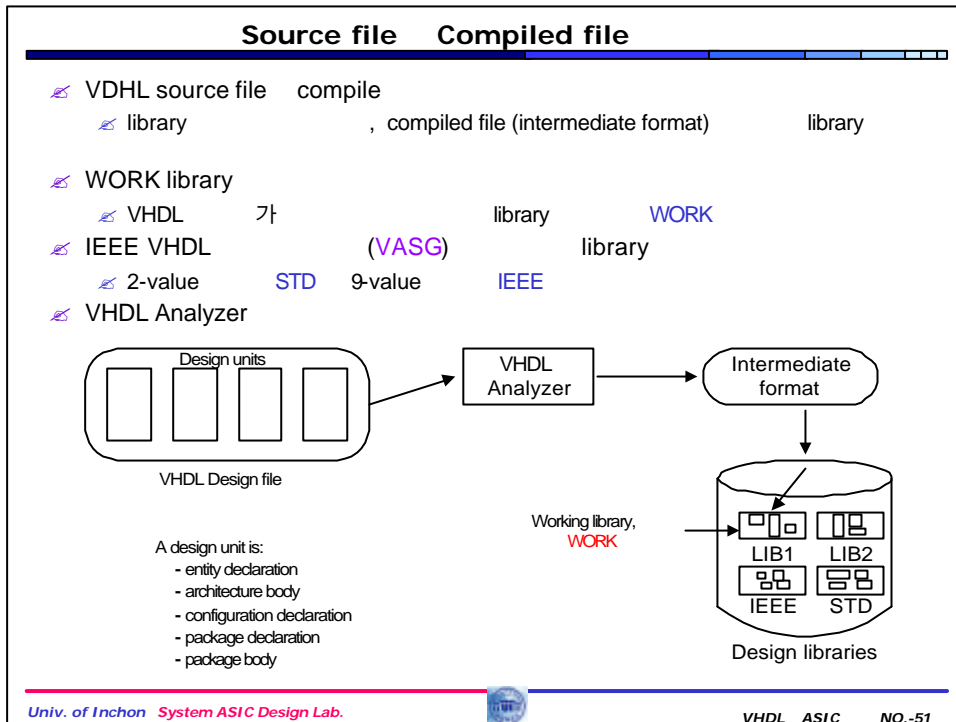
Configuration Declaration Unit :

- ✍ Behavioral View :
 - configuration TOP of XDOWN is
 - for BEHAVIORAL end for;
 - end TOP;
 - ✍ Structural View :
 - ✍
- ```

graph LR
 A[A] --- HA1[HA]
 B[B] --- HA1
 HA1 -- S1 --- HA2[HA]
 HA1 -- S2 --- OR2[OR2]
 CIN[CIN] --- HA2
 HA2 -- S2 --- OR2
 HA2 -- SUM --- SUM[SUM]
 OR2 -- COUT --- COUT[COUT]

```
- configuration FA\_C of FA\_E is
  - for FA\_A**
  - for u1,u2:** HA **use entity** WORK.HA\_E(HA\_A) **generic map** (3 ns, 2 ns);
  - for HA\_A**
  - for u1:** XOR2 **use entity** WORK.XOR2\_E(XOR2\_A) **generic map** (3 ns); **end for;**
  - for u2:** AND2 **use entity** WORK.AND2\_E(AND2\_A) **generic map** (2 ns); **end for;**
  - end for;**
  - end for;**
  - for u3:** OR2 **use entity** WORK.OR2\_E(OR2\_A) **generic map** (2 ns); **end for;**
  - end for;**
  - end;
  - ✍ **Generate**            configuration declaration unit            : generate





## Library Visibility

```

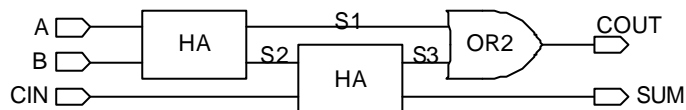
Entity Unit
 architecture configuration unit
--library visibility
library IEEE; use IEEE.std_logic_1164.ALL; use IEEE.std_logic_unsigned.all;
--entity unit
entity ADD_SUB is
 port (A, B : in std_logic_vector(3 downto 0);
 SEL : in std_logic;
 S : out std_logic_vector(3 downto 0);
 CF, ZF : out std_logic);
end ADD_SUB;

Package Declaration Unit
 package body unit
library IEEE; use IEEE.std_logic_1164.ALL;
use IEEE.std_logic_arith.all; use IEEE.std_logic_signed.all;
package PACK_MATCH is
 function COMP2C (A: std_logic_vector) return std_logic_vector;
 function TWO_MUL (A: std_logic_vector; B: std_logic) return std_logic_vector;
end PACK_MATCH;

```



## Structural View



```

Structural View VHDL
Component
Architecture Unit
Component D.B.
Wire
Architecture Unit
Port Map
Component
Component
I/O Port
Formal Name: Pin Name
component HA
 port (A, B: in std_logic;
 C, D: out std_logic);
end component;

Component
Configuration
for U0: HA use entity work.HA(RTL);
Component
3가
Component Instantiation
Actual Name: Wire Name
Pin Wire
H/W
if for generate
Component Instantiation
Pin Wire 2가
Positional Association Mapping
Named Association Mappin
Wire Data Type
Signal

```



## Component Instantiation

```

VHDL
library IEEE; use IEEE.std_logic_1164.all;
entity FA is
 port (A, B, CIN : in std_logic;
 SUM, COUT : out std_logic);
end FA;
--
architecture RTL of FA is
 component HA
 port (A, B : in std_logic;
 C, S : out std_logic);
 end component;
 component OR2
 port (A, B : in std_logic;
 Y : out std_logic);
 end component;
 for U0, U1: HA use entity work.HA(RTL);
 for U2: OR2 use entity work.OR2(RTL);
 signal S1, S2, S3 : std_logic;
begin
 --Positional Association Mapping
 U0 : HA port map (A, B, S1, S2);
 U1 : HA port map (CIN, S2, S3, SUM);
 U2 : OR2 port map (S1, S3, COUT);
 --Named Association Mapping
 --U0 : HA port map (A=>A, B=>B,
 -- C=>S1, S=>S2);
 --U1 : HA port map (A=>CIN, B=>S1,
 -- C=>S3, S=>SUM);
 --U2 : OR2 port map (A=>S1, B=>S3,
 -- Y=>COUT);
end RTL;

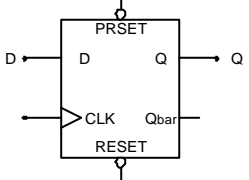
```

## Unconnected Port

Unconnected Port

- Pin Wire7 Port

Unconnected Port



Unconnected Output Port

Unconnected Input Port

VHDL

- Unconnected Input Port
  - VCC '1'
  - GND '0'
- Unconnected Output Port
  - Reserved Identifier OPEN

Unconnected Port

```

architecture RTL of EX1 is
 component EX2
 port (CLK,PRESET,RESET,D:
 in std_logic;
 Q, QBAR : out std_logic);
 end component;
 for U1: EX2 use entity work.DSR(RTL);
 signal VCC : std_logic;
begin
 --VHDL-87
 VCC <= '1';
 U1 : EX2 port map (D => D,
 CLK =>CLK, PRESET =>VCC,
 RESET=>CLR, Q=>Q,
 QBAR=>open);
 --VHDL-93
 --U1 : EX2 port map (D => D,
 -- CLK=>CLK, PRESET =>'1',
 -- RESET=>CLR, Q=>Q,
 -- QBAR=>open);
end RTL;

```

## Generic Map Command

**Generic Behavioral**  
 entity AND\_GATE is  
   generic (TDELAY : time := 10 ns;  
           N : positive := 2);  
   port (A : in std\_logic\_vector(N-1 downto 0);  
        C : out std\_logic);  
 end AND\_GATE;  
 architecture RTL of AND\_GATE is  
 begin  
   P0 : process (A)  
     variable INT : std\_logic;  
   begin  
     INT := '1';  
     for I in (A'length-1) downto 0 loop  
       if (A(I)='0') then  
         INT := '0';  
       end if;  
     end loop;  
     C <= INT after Tdelay;  
   end process;  
 end RTL;

**Generic Map Structural**  
 entity EX is  
   port (D1, D2, D3, D4, D5 : in std\_logic;  
        Q1, Q2 : out std\_logic);  
 end EX;  
 architecture RTL of EX is  
   component AND\_COMP  
     generic (TDELAY : time; N : positive);  
     port (A : in std\_logic\_vector(N-1 downto 0);  
           C : out std\_logic);  
   end component;  
   for all : AND\_COMP use entity  
     work.AND\_GATE(RTL);  
 begin  
   U1: AND\_COMP  
     generic map (n=>2, TDELAY=>8 ns)  
     port map (A(0)=>D1, A(1)=>D2, C=>Q1);  
   U2: AND\_COMP  
     generic map (n=>3, TDELAY=>12 ns)  
     port map (A(0)=>D3, A(1)=>D4,  
               A(2)=>D5, C=>Q2);  
 end RTL;

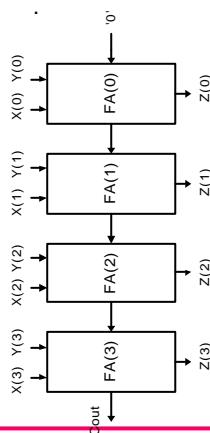


## For Generate Structure View

4-Bit Ripple Carry Adder

| Name | Component Formal Name | Symbol |
|------|-----------------------|--------|
|      |                       |        |

Component Instantiation



Architecture Configuration Unit

```

architecture RTL1 of FA4 is
 -- FA Component
 signal TMP : std_logic_vector(4 downto 0);
begin
 TMP(0) <= '0';
 G: for I in 0 to 3 generate
 L : FA port map (X(I), Y(I), TMP(I),
 Z(I), TMP(I+1));
 end generate;
 COUT <= TMP(4);
end RTL1;
--
configuration CONF_FA4 of FA4 is
 for RTL
 for G
 for L: FA use entity work.FA(RTL); end for;
 end for;
 end for;
end CONF_FA4;

```

Port Map  
Configuration



| If For..Generate           |           | Structure View                            |  |
|----------------------------|-----------|-------------------------------------------|--|
| ✎ 4-Bit Ripple Carry Adder |           | ✎ Architecture Unit                       |  |
| ✎ Name                     | Component | ✎ architecture RTL1 of FA4 is             |  |
| ✎ Instantiation            | Component | -- FA, HA Component                       |  |
|                            |           | --                                        |  |
|                            |           | signal T : std_logic_vector(4 downto 1);  |  |
|                            |           | begin                                     |  |
|                            |           | G0: for I in 0 to 3 generate              |  |
|                            |           | G1 : if I=0 generate                      |  |
|                            |           | L0 : HA port map (X(I),Y(I),Z(I),T(I+1)); |  |
|                            |           | end generate;                             |  |
|                            |           | G2 : if I>=1 and I<=3 generate            |  |
|                            |           | L1 : FA port map                          |  |
|                            |           | (X(I),Y(I),T(I),Z(I),T(I+1));             |  |
|                            |           | end generate;                             |  |
|                            |           | end generate;                             |  |
|                            |           | COUT <= TMP(4);                           |  |
|                            |           | end RTL1;                                 |  |

Univ. of Incheon System ASIC Design Lab.
VHDL ASIC NO.-59

| Generate                            | Configuration Declaration Unit               |
|-------------------------------------|----------------------------------------------|
| ✎ Configuration Unit                | ✎ Component                                  |
| configuration CONF_FA4 of FA4 is    | ✎ component configuration                    |
| for RTL1                            | architecture RTL of TB_FA4 is                |
| for G0                              | signal X, Y: std_logic_vector(N-1 downto 0); |
| for G1                              | signal Z: std_logic_vector(N-1 downto 0);    |
| for L0: HA use entity work.HA(RTL); | signal COUT: std_logic;                      |
| end for;                            | component FA4                                |
| end for;                            | generic(N: integer := 4);                    |
| end for;                            | port (X: in std_logic_vector(N-1 downto 0);  |
| for G2                              | Y: in std_logic_vector(N-1 downto 0);        |
| for L1: FA use entity work.FA(RTL); | Z: out std_logic_vector(N-1 downto 0);       |
| end for;                            | COUT: out std_logic);                        |
| end for;                            | end component;                               |
| end for;                            | for U0: FA4 use configuration                |
| end CONF_FA4;                       | work.CONF_FA4;                               |
|                                     | begin                                        |
|                                     | U0: FA4 port map (X, Y, Z, COUT);            |
|                                     | X <= "0011", "1010" after 40 ns;             |
|                                     | Y <= "1000", "1110" after 20 ns;             |
|                                     | end RTL;                                     |

Univ. of Incheon System ASIC Design Lab.
VHDL ASIC NO.-60

## H/W Sensitivity Signal

reset  
chip clock

Spreading Code  
8-Bit Shift Register

I  
Q

I-Data  
Q-Data

Sensitivity Signal

Block

- ✂ Shift Register Block
  - ✂ reset , chip clock
- ✂ XOR Gates Block
  - ✂ I, Q Shift Register

- ✂ Simulation
  - ✂ Active State:
  - ✂ Suspend State: Sensitivity Signal
- ✂ 8-Bit Shift Register
  - ✂ Reset='0' Falling Edge Chip Clock
  - ✂ Shift
- ✂ 2 XOR Gate Group
  - ✂ Register , I Q
  - ✂ XOR
  - ✂ Gate
- ✂ VHDL
  - ✂ 3 Concurrent
  - ✂ 2 Concurrent
  - ✂ 1 Concurrent

---

Univ. of Incheon System ASIC Design Lab. VHDL ASIC NO.-61

## VHDL

- ✂ Behavioral View Concurrent
  - ✂ signal assignment statement
    - ✂ basic signal assignment
    - ✂ conditional signal assignment
    - ✂ selected signal assignment
  - ✂ process statement with many sequential statements
  - ✂ concurrent subprogram call statement
  - ✂ block statement
    - ✂ basic block
    - ✂ guarded block signal assignment
  - ✂ concurrent assert statement
- ✂ Structural View Concurrent
  - ✂ component instantiation statement
  - ✂ Generate statement

- ✂ Behavioral View
  - ✂ process
  - ✂ function procedure subprogram
  - ✂ wait statement
    - ✂ Vector
  - ✂ basic signal assignment statement
  - ✂ variable assignment statement
  - ✂ if statement
  - ✂ case statement
  - ✂ loop statements
    - ✂ for..loop statement
    - ✂ while loop statement
    - ✂ infinite loop statement
  - ✂ next statement, exit statement
  - ✂ null statement
  - ✂ sequential subprogram call statement
  - ✂ sequential assert statement

---

Univ. of Incheon System ASIC Design Lab. VHDL ASIC NO.-62

| Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>✎ Logical Operators <ul style="list-style-type: none"> <li>✎ not and or nand nor xor xnor(VHDL-93)</li> <li>✎ Operand Type : std_logic, boolean</li> <li>✎ Result Type : std_logic, boolean</li> </ul> </li> <li>✎ Relational Operators <ul style="list-style-type: none"> <li>✎ = /= &lt; &lt;= &gt; &gt;=</li> <li>✎ Operand Type : any type</li> <li>✎ Result Type: boolean</li> </ul> </li> <li>✎ Arithmetic Operators <ul style="list-style-type: none"> <li>✎ + - * / ** MOD REM ABS</li> <li>✎ Operand Type : integer, real, physical</li> <li>✎ Result Type: integer, real, physical</li> </ul> </li> <li>✎ Concatenation Operators : &amp; <ul style="list-style-type: none"> <li>✎ Operand Type : array of any type</li> <li>✎ Result Type : array of any type</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>✎ MOD REM <ul style="list-style-type: none"> <li>✎ <math>A = B*N + (A \text{ mod } B)</math> (N: )</li> <li>✎ <math>A = (A/B)*B + (A \text{ rem } B)</math></li> <li>✎ A=10, B=3: A mod B = 1, A rem B = 1</li> <li>✎ A=-10, B=3: A mod B = 2, A rem B = -1</li> <li>✎ A=11, B=- 4: A mod B = -1, A rem B = 3</li> </ul> </li> <li>✎ operator <ul style="list-style-type: none"> <li>✎ reset='0' or (clk='0' and clk'event)</li> <li>✎ <math>A \leftarrow ((B \text{ nand } C) \text{ nand } D) \text{ nand } E;</math></li> <li>✎ <math>A \leftarrow (B \text{ and } C) \text{ or } (D \text{ and } E)</math></li> </ul> </li> </ul> |



| Predefined Type and Array Attributes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                               |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|--------------|----------------------|----------------------|-------------------|-------------------|------------------|-----------------|-----------------|------------------|----------------------------|---------------------|------------------------------|-----------------|-----------------|---------------------|----------------|------------------------|--------------------|------------------|--------------------|-----------------|---------------------------|----------------------|-------------------|-----------------------|--|-------------------------------|
| <ul style="list-style-type: none"> <li>✎ Attribute <ul style="list-style-type: none"> <li>✎ type attribute, array attribute, signal attribute, entity attribute</li> </ul> </li> <li>✎ Array Attribute <ul style="list-style-type: none"> <li>✎ Type 가</li> <li>type COLOR is (RED, ORANGE, YELLOW, GREEN, BLUE, INDIGO, VIOLET);</li> <li>type STATE is (S0, S1, S2, S3, S4);</li> <li>type INDEXD is range 7 downto 0;</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                               |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| <table border="0" style="width: 100%;"> <thead> <tr> <th style="text-align: left;">Signal Type Attribute</th> <th style="text-align: left;">Return Value</th> </tr> </thead> <tbody> <tr> <td>COLOR'pos(GREEN) = 3</td> <td>COLOR'val(3) = GREEN</td> </tr> <tr> <td>STATE'pos(S2) = 2</td> <td>STATE'val(0) = S0</td> </tr> <tr> <td>COLOR'left = RED</td> <td>STATE'left = S0</td> </tr> <tr> <td>COLOR'low = RED</td> <td>STATE'right = S4</td> </tr> <tr> <td>COLOR'pred(GREEN) = YELLOW</td> <td>COLOR'high = VIORET</td> </tr> <tr> <td>COLOR'leftof(GREEN) = YELLOW</td> <td>STATE'high = S4</td> </tr> <tr> <td>INDEXD'left = 7</td> <td>STATE'succ(S3) = S4</td> </tr> <tr> <td>INDEXD'low = 0</td> <td>STATE'rightof(S2) = S3</td> </tr> <tr> <td>INDEXD'pred(6) = 5</td> <td>INDEXD'right = 0</td> </tr> <tr> <td>INDEXD'succ(6) = 7</td> <td>INDEXD'high = 7</td> </tr> <tr> <td>INDEXD'range = 7 downto 0</td> <td>INDEXD'leftof(6) = 7</td> </tr> <tr> <td>INDEXD'length = 8</td> <td>INDEXD'rightof(6) = 5</td> </tr> <tr> <td></td> <td>INDEXD'reverse_range = 0 to 7</td> </tr> </tbody> </table> | Signal Type Attribute         | Return Value | COLOR'pos(GREEN) = 3 | COLOR'val(3) = GREEN | STATE'pos(S2) = 2 | STATE'val(0) = S0 | COLOR'left = RED | STATE'left = S0 | COLOR'low = RED | STATE'right = S4 | COLOR'pred(GREEN) = YELLOW | COLOR'high = VIORET | COLOR'leftof(GREEN) = YELLOW | STATE'high = S4 | INDEXD'left = 7 | STATE'succ(S3) = S4 | INDEXD'low = 0 | STATE'rightof(S2) = S3 | INDEXD'pred(6) = 5 | INDEXD'right = 0 | INDEXD'succ(6) = 7 | INDEXD'high = 7 | INDEXD'range = 7 downto 0 | INDEXD'leftof(6) = 7 | INDEXD'length = 8 | INDEXD'rightof(6) = 5 |  | INDEXD'reverse_range = 0 to 7 |
| Signal Type Attribute                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Return Value                  |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| COLOR'pos(GREEN) = 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | COLOR'val(3) = GREEN          |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| STATE'pos(S2) = 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | STATE'val(0) = S0             |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| COLOR'left = RED                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | STATE'left = S0               |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| COLOR'low = RED                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | STATE'right = S4              |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| COLOR'pred(GREEN) = YELLOW                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | COLOR'high = VIORET           |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| COLOR'leftof(GREEN) = YELLOW                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | STATE'high = S4               |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| INDEXD'left = 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | STATE'succ(S3) = S4           |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| INDEXD'low = 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | STATE'rightof(S2) = S3        |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| INDEXD'pred(6) = 5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | INDEXD'right = 0              |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| INDEXD'succ(6) = 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | INDEXD'high = 7               |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| INDEXD'range = 7 downto 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | INDEXD'leftof(6) = 7          |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
| INDEXD'length = 8                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | INDEXD'rightof(6) = 5         |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | INDEXD'reverse_range = 0 to 7 |              |                      |                      |                   |                   |                  |                 |                 |                  |                            |                     |                              |                 |                 |                     |                |                        |                    |                  |                    |                 |                           |                      |                   |                       |  |                               |





### Predefined Signal Attributes

|                               |                          |      |             |                |
|-------------------------------|--------------------------|------|-------------|----------------|
| $\Rightarrow$ X2'active       | Simulation Cycle         | X2   | Transaction | True           |
| $\Rightarrow$ X3'event        | Simulation Cycle         | X3   | Event가      | True           |
| $\Rightarrow$ X2'stable(5 ns) | Simulation Cycle         | 5 ns | X2          | Event가<br>True |
| $\Rightarrow$ X2'quiet(5 ns)  | Simulation Cycle<br>True | 5 ns | X2          | Transaction    |
| $\Rightarrow$ X2'last_event   | Simulation Cycle         | 가    |             | Event          |
| $\Rightarrow$ X2'last_active  | Simulation Cycle         | 가    |             | Active         |
| $\Rightarrow$ X2'last_value   | Simulation Cycle         | 가    |             | Event          |



### Predefined Signal Attributes

$\Rightarrow$  Signal X1

$\Rightarrow$  Simulation = 15 ns Attribute

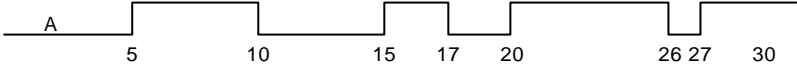
|                                            |         |
|--------------------------------------------|---------|
| $\Rightarrow$ X1'stable(8 ns)              | : FALSE |
| $\Rightarrow$ X1'stable(2 ns)              | : TRUE  |
| $\Rightarrow$ X1'event                     | : FALSE |
| $\Rightarrow$ X1'last_event                | : 5 ns  |
| $\Rightarrow$ X1'last_value                | : '1'   |
| $\Rightarrow$ Simulation = 20 ns Attribute |         |
| $\Rightarrow$ X1'event and X1='1'          | : TRUE  |
| $\Rightarrow$ X1='1' and (not X1'stable)   | : TRUE  |
| $\Rightarrow$ Simulation = 25 ns Attribute |         |
| $\Rightarrow$ X1'event and X1='0'          | : TRUE  |
| $\Rightarrow$ X1='0' and (not X1'stable)   | : TRUE  |


$\Rightarrow$  X1'delayed(2 ns)

$\Rightarrow$  Value가 2 ns delay

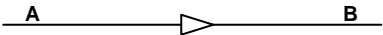
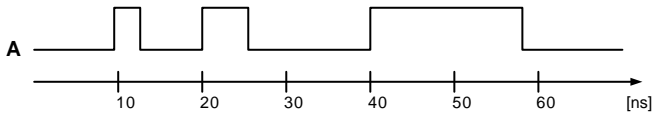
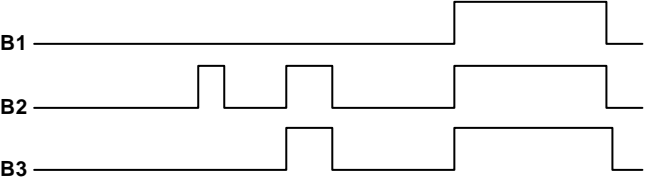



## Delay Model (VHDL-87)

- Delay
  - Delay Selection
    - Inertial Delay : VHDL default delay
      - delay 가 noise
      - delay 가
    - Component delay
    - Transport Delay
      - 가 delay
      - Interconnect delay modeling
  - Internal Delay
    - Delta Delay : concurrent delay event driven simulation
- Simulation : Plot
  - A 
  - B <= transport A after 3 ns;
  - C <= A after 3 ns;
  - D <= A;

Univ. of Incheon System ASIC Design Lab.  VHDL ASIC NO.-67

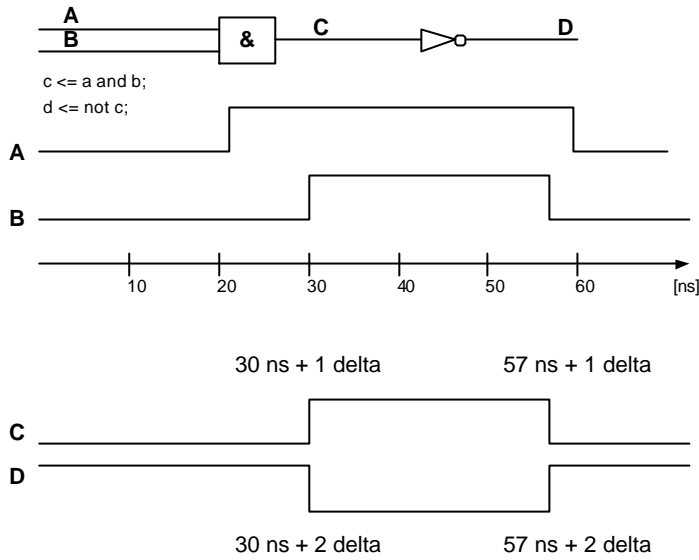
## Delay Model (VHDL-93)

- VHDL-87 Inertial Delay Model
  - Reject Delay Model
    - 
    - 
    - b1 <= inertial a after 10 ns;
    - b2 <= transport a after 10 ns;
    - b3 <= reject 4 ns inertial a after 10 ns;
    - 

Univ. of Incheon System ASIC Design Lab.  VHDL ASIC NO.-68

## Delta Delay Model

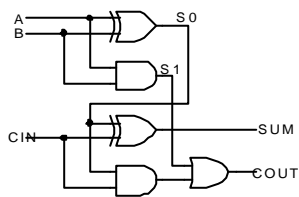
VHDL Simulation Engine                      Delay Model



## Basic Signal Assignment

Full-Adder Gate                      Signal Assignment

Sensitivity Signal



Basic Signal Assignment  
Architecture Unit

architecture RTL of ADDER is

```

signal S0, S1 : std_logic;
begin
 -- gate signal assignment
 SUM <= S0 xor CIN;
 COUT <= S1 or (S0 and CIN);
 S0 <= A xor B;
 S1 <= A and B;

```

end RTL;

Aggregate :                      Target Signal Source

Assign

```

architecture RTL of EX is
 signal A : std_logic_vector(4 downto 0);
 signal B : std_logic_vector(4 downto 0);
begin
 A <= (others => '0');
 -- A <= "00000";
 -- A <= ('0', '0', '0', '0', '0');

 B <= (1=>C(2), 3=>C(1), others => D(0));
 -- B <= D(0)&C(1)&D(0)&C(2)&D(0);
end RTL;

```

Source                      Target Signal

## Conditional Signal Assignment

### Syntax

```

signal_identifier <= waveform_1 when condition_1 else
 waveform_2 when condition_2 else

 waveform_(N-1) when condition_(N-1) else
 waveform_N ;

```

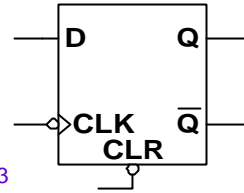
### D Type Flip-Flop VHDL Symbol

architecture A2 of DFF is  
begin

```

Q <= '0' when CLR='0' else
 D when (CLK='0' and CLK'EVENT) else
 Q;
-- D when (CLK='0' and CLK'EVENT); Valid in VHDL-93
QBAR <= not Q;

```



### Data Selector Conditional Signal Assignment

```

MUXOUT <= I0 when (SEL='0') else
 I1; -- when (SEL='1') else 'X';

```



## Selected Signal Assignment

### Syntax

```

with expression select
signal_identifier <=
 waveform1 when expression_1,

 waveform(n-1) when expression(n-1),
 waveform(n) when others ;

```

### 2\*1 MUX

```

library IEEE;
use IEEE.std_logic_1164.ALL;
entity MUX21 is
 port (SEL, A, B : in std_logic;
 C : out std_logic);
end MUX21;
architecture RTL of MUX21 is
begin
 with SEL select
 C <= A after 10 ns when '0',
 B after 10 ns when others;
end RTL;

```

### Half-Adder

```

library IEEE;
use IEEE.std_logic_1164.ALL;
entity HA is
 port (A, B : in std_logic;
 SUM, COUT : out std_logic);
end HA;
architecture RTL of HA is
 signal TMP1, TMP2 :
 std_logic_vector(1 downto 0);
begin
 TMP1 <= A & B;
 with TMP1 select
 TMP2 <= "00" when "00",
 "01" when "01",
 "01" when "10",
 "10" when others;
 (COUT, SUM) <= TMP2;
end RTL;

```



## Process

---

☞ Syntax

```
[Label :] process [(Sensitivity_Signals)]
-- declaration statements
begin
-- sequential activity statements
end process [Label] ;
```

☞ Sensitivity Signal

- ☞ Wait
- ☞ Simulation Signal
- ☞ Process Sensitivity Signal List
- ☞ Event가
- ☞ Event가
- ☞ Simulation
- ☞ H/W

☞ Sensitivity Signal

- ☞ 1 Wait
- ☞ Simulation Wait
- ☞ Wait Simulation
- ☞ Stimulus Vector

☞ Sensitivity Signal

- ☞ Half-Adder

```
process (A, B) process (A)
begin begin
SUM <= A xor B; SUM<=A xor B;
COUT <= A and B; COUT<=A and B;
end process; end process;
```

- ☞ Simulation ( )
- ☞ Synthesis ( )

---

Univ. of Incheon System ASIC Design Lab. VHDL ASIC NO.-73

## Signal Variable Assignment (1)

---

☞ Assignment : Target Identifier Object 2가

- ☞ Signal Assignment
  - ☞ [label:] signal\_identifier <= [transport] expression [after expression];
- ☞ Variable Assignment
  - ☞ [label:] variable\_identifier := value\_expression;

☞ 2가 Assignment Code -1

```
process
variable num,sum:integer := 0;
begin
wait for 10 ns;
num := num + 1;
sum := sum + num;
end process;
--
process begin
wait for 10 ns;
num <= num + 1;
sum <= sum + num;
end process;
```

|                 | <u>variable process</u> |            |            | <u>signal process</u> |            |            |
|-----------------|-------------------------|------------|------------|-----------------------|------------|------------|
|                 | <i>time</i>             | <i>num</i> | <i>sum</i> | <i>time</i>           | <i>num</i> | <i>sum</i> |
|                 | 0                       | 0          | 0          | 0                     | 0          | 0          |
|                 | 10                      | 1          | 1          | 10                    | 0          | 0          |
|                 | 10+                     | 1          | 1          | 10+                   | 1          | 0          |
| --              | 20                      | 2          | 3          | 20                    | 1          | 0          |
| process begin   | 20+                     | 2          | 3          | 20+                   | 2          | 1          |
| wait for 10 ns; | 30                      | 3          | 6          | 30                    | 2          | 1          |
| num <= num + 1; | 30+                     | 3          | 6          | 30+                   | 3          | 3          |

---

Univ. of Incheon System ASIC Design Lab. VHDL ASIC NO.-74

## Signal Variable Assignment

### Signal Assignment

```

library IEEE; use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity EX_1 is end EX_1;
architecture RTL of EX_1 is
 signal A, B, C, D, X, Y:
 std_logic_vector(3 downto 0);
begin
 process (A, B, C, D)
 begin
 D <= A;
 X <= B + D;
 D <= C;
 Y <= B + D;
 end process;
 A <= "0001", "0010" after 10 ns,
 "0011" after 20 ns, "0100" after 30 ns;
 B <= "1000";
 C <= "0101", "0110" after 10 ns,
 "0111" after 20 ns, "1000" after 30 ns;
end RTL;

```

### Variable Assignment

```

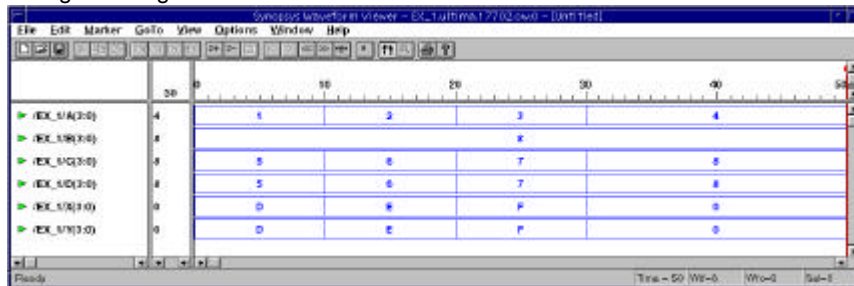
library IEEE; use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity EX_2 is end EX_2;
architecture RTL of EX_2 is
 signal A, B, C, X, Y:
 std_logic_vector(3 downto 0);
begin
 process (A, B, C)
 variable D: std_logic_vector(3 downto 0);
 begin
 D := A;
 X <= B + D;
 D := C;
 Y <= B + D;
 end process;
 A <= "0001", "0010" after 10 ns,
 "0011" after 20 ns, "0100" after 30 ns;
 B <= "1000";
 C <= "0101", "0110" after 10 ns,
 "0111" after 20 ns, "1000" after 30 ns;
end RTL;

```

## Simulation Waveform

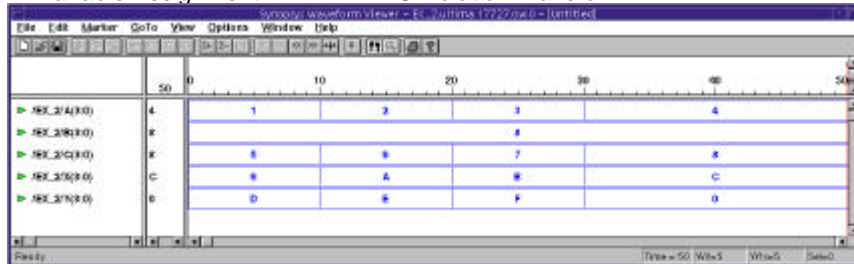
### Signal Assignment

### Simulation Waveform



### Variable Assignment

### Simulation Waveform



## Wait

### Syntax

`wait [ on list ] [ until condition ] [ for time ] ;`



`wait;`

Simulation Restart

`wait on A;`

Signal A Simulation

`wait until (CLK='0' and CLK'event);`

CLK Signal Falling Edge Simulation

`wait for 100 ns;`

100 ns Simulation

`wait until A='1' for 50 ns;`

50 ns A 가 A='1'

`wait on A until (C='1') for 50 ns;`

50 ns A 가 C='1'

### Wait

Synthesis Code : Clock wait Tool

wait until CLK Edge

Test Bench Modeling 가

## Process

### Process

```

process (A)
begin
 C1<=NOT A;
end process;

process
begin
 wait on A;
 C3<=NOT A;
end process;

process
begin
 wait until A='1';
 C4<=not A;
end process;

```

### Simulation

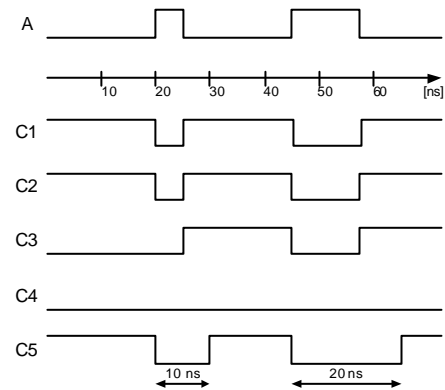
```


process
begin
 C2<=NOT A;
 wait on A;
end process;


process
begin
 C5<=NOT A;
 wait until A='1' for 10 ns;
end process;

```

### Simulation



| Clock                                                                                                                                                                                                                                                                                                                                   | Wait                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>☞ Test Bench Modeling</p> <p>☞ 가 : CLK1, CLK2, CLK3, CLK4, CLK5, CLK6</p> <pre> process begin   CLK1 &lt;= not CLK1;   wait for 25 ns; end process; -- process (CLK3) begin   CLK3 &lt;= not CLK3 after 25 ns; end process; -- process begin   CLK5 &lt;= '0'; wait for 20 ns;   CLK5 &lt;= '1'; wait for 30 ns; end process; </pre> | <p style="text-align: center;">VHDL</p> <p>☞ '0' .</p> <pre> process begin   CLK2 &lt;= not CLK2 after 25 ns;   wait on CLK2; end process; -- CLK4 &lt;= not CLK4 after 25 ns; -- process begin   if (now = 0 ns) then     CLK6 &lt;= '0'; wait for 100 ns;   else     CLK6 &lt;= NOT CLK6; wait for 25 ns;   end if; end process; </pre> |
| <p>☞ Quiz: Simulation</p>                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                           |
| <p>Univ. of Incheon System ASIC Design Lab.</p>                                                                                                                                                                                                       | <p>VHDL ASIC NO.-79</p>                                                                                                                                                                                                                                                                                                                   |

| If                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                         |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>☞ Syntax</p> <ul style="list-style-type: none"> <li>☞ if...then...end if;</li> <li>☞ if... then...else...end if;</li> <li>☞ if...then...elsif ...elsif ...else...end if;</li> <li>☞ if...then...if...then.....end if....end if;</li> </ul> <p>☞ Latch</p> <p>☞ IF</p> <p>☞ -1: 2*1 MUX</p> <pre> process (A, B, SEL) begin   if SEL='0' then     Y &lt;= A;   else     Y &lt;= B;   end if; end process; </pre> | <p>☞ -2: 4*1 MUX</p> <pre> process (SEL, A, B, C, D) begin   if SEL="00" then     Y &lt;= A;   elsif SEL="01" then     Y &lt;= B;   elsif SEL="10" then     Y &lt;= C;   else     Y &lt;= D;   end if; end process; </pre> <p>☞ -3: Latch</p> <pre> process (EN, D) begin   if EN='1' THEN     Q &lt;= D;   end if; end process; </pre> |
| <p>Univ. of Incheon System ASIC Design Lab.</p>                                                                                                                                                                                                                                                                                 | <p>VHDL ASIC NO.-80</p>                                                                                                                                                                                                                                                                                                                 |



## Case

### Syntax

```

case expression is
 when condition_1 =>
 (sequential_statements;)

 when condition_(n-1) =>
 (sequential_statements;)
 when others =>
 (sequential_statements;)
end case ;

```

- when condition
- when Value =>
- when Value | Value | Value... =>
- when Value to Value =>
- when others =>

### Truth Table

- 1 - Bit operator Variable
- Multi-Bits & expression

### 1-Bit Half-Adder

```

architecture RTL1 of EX is
 --signal TMP1 :
 -- std_logic_vector(1 downto 0);
begin
 --TMP1 <= A & B;
 --process (TMP1)
 process (A, B)
 variable TMP1, TMP2 :
 std_logic_vector(1 downto 0);
 begin
 TMP1 := A & B;
 case TMP1 is
 when "00" => TMP2 := "00";
 when "01" | "10" => TMP2 := "01";
 when others => TMP2 := "10";
 end case;
 (COUT, SUM) <= TMP2;
 end process;
end RTL;

```



## For..Loop

### Syntax

```

[Label:]
for loop_parameter in discrete_range loop
 --sequential statements
end loop [Label] ;

```

### Range

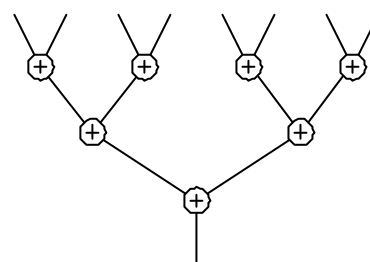
- integer\_expression to ( downto)
- integer\_expression array\_attribute'range
- array\_attribute'reverse\_range

### loop\_parameter

- Signal Variable

### For..Loop

Data-flow Graph  
VHDL



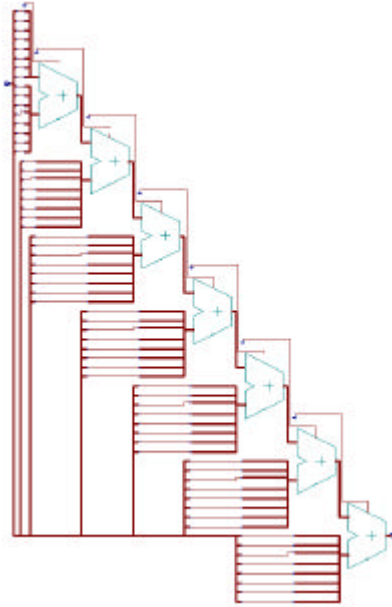
- Signal Assignment
- Data-Flow
- For..Loop



## For..Loop

(1)

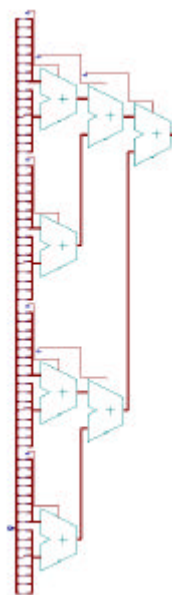
```
library IEEE; use IEEE.std_logic_1164.all;
package PACK is
 type RAM_TYPE is array(0 to 7) of
 std_logic_vector(7 downto 0);
end PACK;
library IEEE; use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use work.PACK.all;
entity EX_3 is
 port (A: in RAM_TYPE;
 Y: out std_logic_vector(7 downto 0));
end EX_3;
architecture RTL of EX_3 is
begin
 process (A)
 variable SUM: std_logic_vector(7 downto 0);
 begin
 SUM := (others => '0');
 for I in 0 to 7 loop
 SUM := SUM + A(I);
 end loop;
 Y <= SUM;
 end process;
end RTL;
```



## For..Loop

(2)

```
library IEEE; use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use work.PACK.all;
entity EX_4 is
 port (A: in RAM_TYPE;
 Y: out std_logic_vector(7 downto 0));
end EX_4;
architecture RTL of EX_4 is
begin
 process (A)
 variable COUNT: integer;
 variable B: RAM_TYPE;
 begin
 COUNT := 8;
 for I in 0 to 7 loop
 B(I) := A(I);
 end loop;
 for LEVEL in 0 to 2 loop
 COUNT := COUNT / 2;
 for I in 0 to (COUNT-1) loop
 B(I) := B(I*2) + B(I*2+1);
 end loop;
 end loop;
 Y <= B(0);
 end process;
end RTL;
```



## While..Loop

✎ While..Loop      Syntax  
 [Label:] **while** condition **loop**  
     --sequential statements  
**end loop** [Label] ;  
 ✎ 1 ~ N  
 entity ADD\_LOOP is  
     port ( N : in integer;  
           SUM : out integer);  
 end ADD\_LOOP;  
 architecture RTL of ADD\_LOOP is  
 begin  
     process (N)  
         variable TMP, I : integer;  
     begin  
         TMP := 0; I := 1;  
         **while** (I <= N) **loop**  
             TMP := TMP + I;   I := I + 1;  
         **end loop**;  
         SUM <= TMP;  
     end process;  
 end RTL;

✎ Infinite Loop  
 process (N)  
     variable TMP, I : integer;  
 begin  
     TMP := 0; I := 0;  
     FIRST: loop  
         I := I + 1;  
         TMP := TMP + 1;  
         **exit** when (I = N) ;  
     end loop FIRST;  
 end process;

✎ Advanced Synthesis Tool  
     가  
 ✎ Behavioral Compiler



## Next Exit

✎ Next      Syntax  
     ✎ **next**;  
     ✎ **next** loop\_label ;  
     ✎ **next when** boolean\_expression ;  
     ✎ **next** loop\_label  
         **when** boolean\_expression ;

✎  
 LBL1 : process (S)  
     variable TMP, J : integer := 0;  
 begin  
     TMP := 0; J := 0;  
     A\_LOOP: for I in 0 to 7 loop  
         J := J + 1;  
         if J > S then  
             **next** A\_LOOP;  
         end if;  
         TMP := TMP + 1;  
     end loop;  
 end process LBL1;

✎ Exit      Syntax  
     ✎ **exit**;  
     ✎ **exit** loop\_label ;  
     ✎ **exit when** boolean\_expression ;  
     ✎ **exit** loop\_label  
         **when** boolean\_expression ;

✎  
 LBL2 : process (S)  
     variable SUM, CNT : integer := 0;  
 begin  
     SUM:= 0; CNT:= 0;  
     FIRST: loop  
         CNT:=CNT+1; SUM:=SUM+CNT;  
         **exit** when SUM > 100;  
     end loop FIRST;  
     --SECOND: loop  
     -- CNT:=CNT+1; SUM:=SUM+CNT;  
     -- if SUM > 100 then **exit**;  
     --end loop SECOND;  
 end process LBL2;



## Block

### Basic Block Statement

```

library IEEE; use IEEE.std_logic_1164.all;
entity BLK_FA is
 port (A, B, C_IN: in std_logic;
 C_OUT, SUM: out std_logic);
end BLK_FA;
architecture RTL of BLK_FA is
 signal S0, S1, S2 : std_logic;
begin
 NAND_1: block begin
 S0 <= A xor B;
 S1 <= A and B;
 end block;
 NAND_2: block begin
 SUM <= S0 xor C_IN;
 S2 <= S0 and C_IN;
 end block;
 OR2: block begin
 C_OUT <= S1 or S2;
 end block;
end RTL;

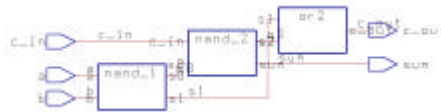
```

### Script File

```

read -f vhdl blk_fa.vhd
group -hdl_block nand_1
group -hdl_block nand_2
group -hdl_block or2
/* create_clock clk */
max_delay 0 -to all_outputs()
compile -map_effort high
create_schematic
write -f db
write -f edif -hierarchy -o blk_fa.edf

```



### Guarded Block Statement

```

D_FF: block(clk='0' and clk'event)
begin
 Q <= guarded D;
end block;

```

## Assert

(1)

### Syntax

- assert condition ;
- assert condition report "message" ;
- assert condition severity level ;
- assert condition report "message" severity level ;

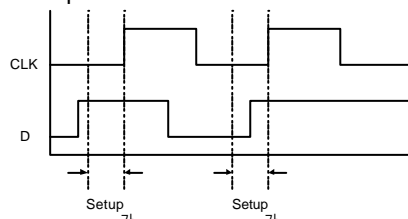
### 

- Passive : HW
- Concurrent, Sequential Assert

### D Flip-Flop Assert

- Setup Time Violation

### Setup Time



### Assert

```

library IEEE; use IEEE.std_logic_1164.all;
entity DFF_A_SR is
 generic (D_SETUP : time := 2 NS);
 port (SD_N, RD_N, CLK, DIN : in std_logic;
 Q, Q_N : out std_logic);
begin
 assert (SD_N='1' or RD_N='1')
 report "PRESET and CLEAR are both '0'" ;
 process (CLK, DIN)
 variable D_CHANGE : time := 0 ns;
 variable D_VALUE, CLK_VALUE : std_logic;
 begin
 if (D_VALUE /= DIN) then
 D_CHANGE := now; D_VALUE := DIN;
 end if;
 if (CLK_VALUE /= CLK) then
 CLK_VALUE := CLK;
 if (CLK='1') then
 assert (now - D_CHANGE >= D_SETUP)
 report "SETUP VIOLATION" ;
 end if;
 end if;
 end process;
end DFF_A_SR;

```

## Assert

(2)

✍ Assert



## Package

✍ Package

✍

✍ Syntax

```
package package_name is
--exported_subprogram_declarations
--exported_constant_declarations
--exported_components
--exported_type_declarations
--attribute_declarations
--attribute_specifications
end [package_name] ;
package body package_name is
--exported_subprogram_bodies
--internal_subprogram_declarations
--internal_subprogram_bodies
--internal_constant_declarations
--internal_components
--internal_type_declarations
end [package_name] ;
```

✍ Package

```
package EX_PKG is
subtype INT8 is INTEGER range 0 to 255;
constant ZERO : INT8 := 0;
constant MAX : INT8 := 100;
procedure Increment (variable Count:
inout INT8);
end EX_PKG;
```

✍ Package

```
package body EX_PKG is
procedure Increment (variable Count:
inout INT8) is
begin
if (Count >= MAX) then
Count := Zero;
else
Count := Count + 1;
end if;
end Increment;
end EX_PKG;
```



## Predefined Package

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                   |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>Predefined Packages             <ul style="list-style-type: none"> <li>Group Package.</li> <li>Library Name : <b>STD</b></li> <li>Package Name                 <ul style="list-style-type: none"> <li>standard</li> <li>textio</li> <li>textio package</li> <li>visibility</li> </ul> </li> <li>Library Name : <b>IEEE</b></li> <li>Package Name                 <ul style="list-style-type: none"> <li>std_logic_1164</li> <li>std_logic_textio</li> </ul> </li> <li>Visibility</li> </ul> </li> <li>Synopsys Package.             <ul style="list-style-type: none"> <li>Library Name : <b>IEEE</b></li> <li>Package Name                 <ul style="list-style-type: none"> <li>std_logic_unsigned</li> <li>std_logic_signed</li> <li>std_logic_arith</li> </ul> </li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>Package             <ul style="list-style-type: none"> <li>\$/synopsys/packages/IEEE/src/*.vhd</li> <li>std_logic_1164.vhd</li> <li>std_logic_arith.vhd</li> <li>std_logic_signed.vhd</li> <li>std_logic_unsigned.vhd</li> <li>std_logic_textio.vhd</li> </ul> </li> </ul> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Unsigned Data

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>Data Type: std_logic_vector</li> <li>Package: std_logic_unsigned</li> </ul> <p>VHDL</p> <pre> library ieee; use ieee.std_logic_1164.all; use ieee.std_logic_unsigned.all; entity TB_UNSIGNED is end TB_UNSIGNED; architecture RTL of TB_UNSIGNED is   signal s1, s2 : std_logic_vector(3 downto 0);   signal a, b : std_logic_vector(4 downto 0);   signal c, d : std_logic_vector(7 downto 0); begin   process   begin     s1 &lt;= "0011"; s2 &lt;= "0101"; wait for 50 ns;     s1 &lt;= "1101"; s2 &lt;= "0101"; wait for 50 ns;     s1 &lt;= "1101"; s2 &lt;= "1011"; wait for 50 ns;     s1 &lt;= "0011"; s2 &lt;= "1011"; wait for 50 ns;   end process;   a &lt;= (0&amp;s1) + s2; b &lt;= (0&amp;s1) - s2;   c &lt;= s1 * s2; d &lt;= s1 * "1000"; end RTL;         </pre> | <ul style="list-style-type: none"> <li>Data Type: unsigned</li> <li>Package: std_logic_arith</li> </ul> <p>VHDL</p> <pre> library ieee; use ieee.std_logic_1164.all; use ieee.std_logic_arith.all; entity TB_ARITH_UNSIGNED is end TB_ARITH_UNSIGNED; architecture RTL of TB_ARITH_UNSIGNED is   signal s1, s2 : unsigned(3 downto 0);   signal a, b : unsigned(4 downto 0);   signal c : unsigned(7 downto 0); begin   process   begin     s1 &lt;= "0011"; s2 &lt;= "0101"; wait for 50 ns;     s1 &lt;= "1101"; s2 &lt;= "0101"; wait for 50 ns;     s1 &lt;= "1101"; s2 &lt;= "1011"; wait for 50 ns;     s1 &lt;= "0011"; s2 &lt;= "1011"; wait for 50 ns;   end process;   a &lt;= (0&amp;s1) + s2;   b &lt;= (0&amp;s1) - s2;   c &lt;= s1 * s2; end RTL;         </pre> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Unsigned Data      Simulation

---

std\_logic\_vector    std\_logic\_unsigned package      Simulation

unsigned    std\_logic\_arith package      simulation

Univ. of Incheon System ASIC Design Lab.      VHDL ASIC NO.-93

## Signed Data

---

Data Type: std\_logic\_vector  
Package: std\_logic\_signed

**VHDL**

```

library ieee; use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;
entity TB_SIGNED is end TB_SIGNED;
architecture RTL of TB_SIGNED is
 signal s1, s2 : std_logic_vector(3 downto 0);
 signal a,b : std_logic_vector(4 downto 0);
 signal c, d : std_logic_vector(7 downto 0);
begin
 process
 begin
 s1 <= "0011"; s2 <= "0101"; wait for 50 ns;
 s1 <= "1101"; s2 <= "0101"; wait for 50 ns;
 s1 <= "1101"; s2 <= "1011"; wait for 50 ns;
 s1 <= "0011"; s2 <= "1011"; wait for 50 ns;
 end process;
 a <= (s1(3)&s1) + (s2(3)&s2);
 b <= (s1(3)&s1) - (s2(3)&s2);
 c <= s1 * s2; d <= s1 * "1000";
end RTL;

```

Data Type: signed  
Package: std\_logic\_arith

**VHDL**

```

library ieee; use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity TB_ARITH_SIGNED is
end TB_ARITH_SIGNED;
architecture RTL of TB_ARITH_SIGNED is
 signal s1, s2 : signed(3 downto 0);
 signal a, b : signed(4 downto 0);
 signal c : signed(7 downto 0);
begin
 process
 begin
 s1 <= "0011"; s2 <= "0101"; wait for 50 ns;
 s1 <= "1101"; s2 <= "0101"; wait for 50 ns;
 s1 <= "1101"; s2 <= "1011"; wait for 50 ns;
 s1 <= "0011"; s2 <= "1011"; wait for 50 ns;
 end process;
 a <= (s1(3)&s1) + (s2(3)&s2);
 b <= (s1(3)&s1) - (s2(3)&s2);
 c <= s1 * s2;
end RTL;

```

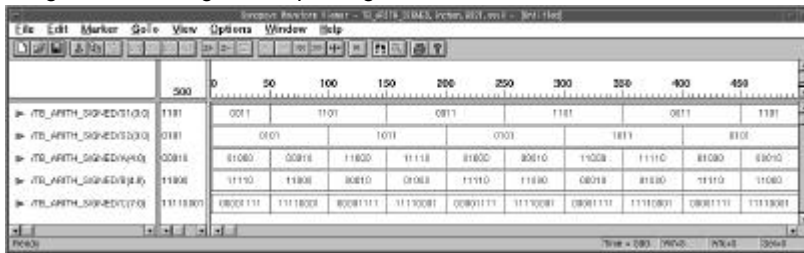
Univ. of Incheon System ASIC Design Lab.      VHDL ASIC NO.-94

## Signed Data                      Simulation

std\_logic\_vector    std\_logic\_signed package                      simulation



signed    std\_logic\_arith package                      simulation



## Subprogram                      Function Subprogram

- Subprogram
    - Function
    - Procedure
  - Subprogram
    - Package
    - Package
    - Architecture
    - Process
  - Function                      Syntax
    - Package
  - function name (formal\_parameter\_list)  
return data\_type is
  - Package
  - function name (formal\_parameter\_list)  
return data\_type is
  - declarative statements
  - begin
  - sequential statements
  - including return expression
  - end name ;
- Parameter
    - Data Mode
      - in
    - Data Object
      - constant                      signal
      - constant
    - Formal Name                      Actual Name
    - signal                      :signal 가
    - constant                      :
    - signal,constant,variable 가
  - Function                      Visibility
    - Package
      - Package 가
    - Package Body
      - Package Body 가
    - Architecture
      - Architecture 가
    - Process
      - Process 가



## Function Subprogram

### Function Subprogram

```

package MYPACK is
 function MAX (A, B: in std_logic_vector)
 return std_logic_vector;
 function EQUAL (A, B : in std_logic_vector)
 return boolean;
end;
package body MYPACK is
 function MAX (A, B: in std_logic_vector)
 return std_logic_vector is
 begin
 if A > B then
 return (A);
 else
 return (B);
 end if;
 end MAX ;
 function EQUAL(A, B : in std_logic_vector)
 return boolean is
 begin
 return (A=B);
 end EQUAL;
end MYPACK;

```

### Function Subprogram

```

library IEEE; use IEEE.std_logic_1164.all;
use work.MYPACK.all;
entity EX is
 port (D1, D2: in
 std_logic_vector(3 downto 0);
 DATA1, DATA2: in
 std_logic_vector(7 downto 0);
 Q: out std_logic_vector(3 downto 0);
 DATA_OUT: out boolean);
end EX;
architecture RTL of EX is
begin
 Q <= MAX(d1, d2);
 --Concurrent function call
 process (DATA1, DATA2)
 begin
 DATA_OUT <= EQUAL(DATA1, DATA2);
 --Sequential function call
 end process;
end RTL;

```

## Procedure Subprogram

### Procedure Syntax

```

Package
procedure name (formal_parameter_list) ;
Package
procedure name (formal_parameter_list) is
-- declarative statements
begin
-- sequential statements
end name ;
Parameter
Data Mode
in, out, inout 가
in mode
Data Class
Mode가 in ,
constant, signal, variable
가
Mode가 in ,
constant

```

```

Mode가 inout out ,
variable, signal 가
Mode가 inout out ,
variable
Formal Name Actual Name
signal :signal 가
variable : variable 가
constant :
signal,constant,variable 가
Procedure Visibility
Package
Package 가
Package Body
가 Package Body
Architecture
가 Architecture
가
Process
Process 가

```

## Procedure Subprogram (1)

```

✎ Procedure
package MYPACK is
 procedure COMPUTE (A, B: in integer;
 MEAN, MAX: out integer);
end MYPACK;
package body MYPACK is
 procedure COMPUTE (A, B: in integer;
 MEAN, MAX: out integer) is
 begin
 MEAN := (A+B)/2;
 if A > B then
 MAX := A;
 else
 MAX := B;
 end if;
 end COMPUTE;
end MYPACK;

```

```

✎ Procedure
use work.MYPACK.all;
entity EX is
 port (D1, D2, D3, D4: in integer;
 Q1, Q2, Q3, Q4: out integer);
architecture RTL1 of EX is
begin
 COMPUTE(D1, D2, Q1, Q2);
 -- , Q1, Q2 not variables
 --Concurrent procedure call
 process(D3, D4)
 variable A, B: integer;
 begin
 COMPUTE(D3, D4, A, B);
 -- , A and B variables
 --Sequential procedure call
 Q3 <= A;
 Q4 <= B;
 end process;
end RTL1;

```



## Procedure Subprogram (2)

```

✎ Procedure
package MYPACK is
 procedure COMPUTE (A, B: in integer;
 signal MEAN, MAX: out integer);
end MYPACK;
package body MYPACK is
 procedure COMPUTE (A, B: in integer;
 signal MEAN, MAX: out integer) is
 begin
 MEAN <= (A+B)/2;
 if A > B then
 MAX <= A;
 else
 MAX <= B;
 end if;
 end COMPUTE;
end MYPACK;

```

```

✎ Procedure
use work.MYPACK.all;
entity EX is
 port (D1, D2, D3, D4: in integer;
 Q1, Q2, Q3, Q4: out integer);
architecture RTL2 of EX is
begin
 COMPUTE(D1, D2, Q1, Q2);
 -- , Q1 and Q2 signals
 --Concurrent procedure call
 process(D3, D4)
 begin
 COMPUTE(D3, D4, Q3, Q4);
 -- , Q3 and Q4 signals
 end process;
end RTL2;

```



## Procedure Subprogram (3)

```

library IEEE; use IEEE.std_logic_1164.ALL;
package EXAMPLE is
 procedure RCA (A, B : in std_logic_vector;
 CIN : in std_logic; SUM : out std_logic_vector;
 COUT : out std_logic);
end EXAMPLE;
package body EXAMPLE is
 function XOR3 (A, B, C : in std_logic) return
 std_logic is
 begin
 return (A xor B xor C);
 end XOR3;
 procedure RCA (A, B : in std_logic_vector;
 CIN : in std_logic; SUM : out std_logic_vector;
 COUT : out std_logic) is variable C :
 std_logic_vector(A'high-A'low+1 downto 0);
 begin
 C(0) := CIN;
 for I in 0 to (A'high-A'low) loop
 SUM(I+SUM'low) :=
 XOR3(A(I+A'low), B(I+B'low), C(I));
 C(I+1) := (A(I+A'low) and (I+B'low))
 or (C(I) and (A(I+A'low) or B(I+B'low)));
 end loop;
 COUT := C(C'high);
 end RCA;
end EXAMPLE;

```

```

library IEEE; use IEEE.std_logic_1164.ALL;
use work.EXAMPLE.all;
entity ADDER_TEST is
 port (A, B: in
 std_logic_vector(15 downto 0);
 CIN : in std_logic;
 SUM : out
 std_logic_vector(15 downto 0);
 COUT : out std_logic);
end ADDER_TEST;
architecture RTL of ADDER_TEST is
begin
 process (A,B,CIN)
 variable TMP_SUM:
 std_logic_vector(SUM'range);
 variable TMP_COUT : std_logic;
 begin
 RCA(A, B, CIN, TMP_SUM, TMP_COUT);
 SUM <= TMP_SUM;
 COUT <= TMP_COUT;
 end process;
end RTL;

```



## Operator Overload

|                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                          |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> type 가   type   overload   function   Function " "   " "   Synopsis Vendor가   Package   function "+" (L: std_logic_vector,     R: std_logic_vector)     return std_logic_vector;   function "+" (L: std_logic_vector,     R: integer)     return std_logic_vector;   </pre> | <pre> VHDL Compiler   Compile VHDL Code +   Operator   Compiler Operand Data Type   Function   Type Function   Function Algorithm   Function   Error Message   Function   Algorithm   C &lt;= A + B   A, B가 std_logic_vector   A std_logic_vector , B   integer   Function Name   </pre> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



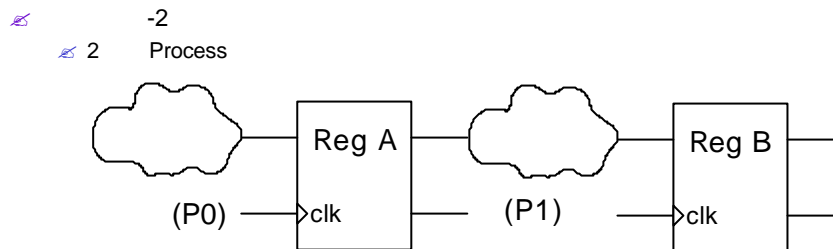
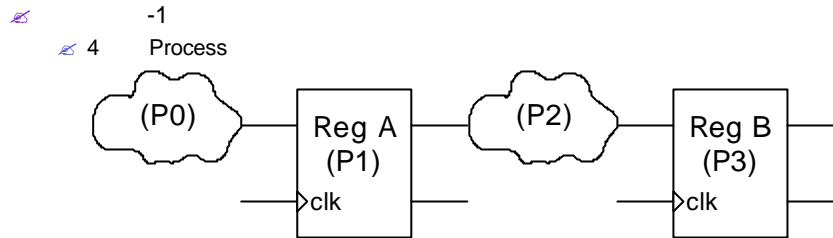
## Subprogram Overload

|                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>✍ Subprogram<br/>         , Formal Parameter<br/>         Return Value      Data Type</p> <p>✍ Subprogram</p> <p>✍</p> <pre> library IEEE; use IEEE.std_logic_1164.ALL; use IEEE.std_logic_arith.all; use IEEE.std_logic_unsigned.all; package EXAMPLE is   function CONV_2COMP     (A : std_logic_vector; K : integer)   return std_logic_vector;   function CONV_2COMP     (A : integer; K : integer)   return std_logic_vector; end EXAMPLE;</pre> | <pre> package body EXAMPLE is   function CONV_2COMP     (A : std_logic_vector; K : integer)   return std_logic_vector is   begin     return (not (A(K-1) downto 0)+1);   end;   --   function CONV_2COMP     (A : integer; K : integer)   return std_logic_vector is   begin     return (not       conv_std_logic_vector(A,K)+1);   end; end EXAMPLE;</pre> <p>✍ VHDL Compiler</p> <pre> ✍ B &lt;= CONV_2COMP(A, 8);  ✍ A7↑ std_logic_vector ✍ A7↑ integer</pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



## Chap 3. Digital System VHDL Modeling

## Digital System RTL



## Digital System VHDL Modeling Template

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>✎ H/W Model</li> <li>✎ Entity + Architecture</li> <li>✎ Entity Unit             <ul style="list-style-type: none"> <li>✎ Visibility                 <ul style="list-style-type: none"> <li>✎ 9-value Visibility</li> <li>✎ Arithmetic Operator Visibility</li> </ul> </li> <li>✎ Package                 <ul style="list-style-type: none"> <li>✎ Visibility</li> </ul> </li> <li>✎ Block Diagram</li> <li>✎ Generic, Port Data</li> <li>✎ Data Identifier, Mode, Type</li> </ul> </li> <li>✎ Architecture Unit             <ul style="list-style-type: none"> <li>✎</li> <li>✎</li> <li>✎</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>✎ Basic Modeling Template</li> </ul> <pre> library IEEE; use IEEE.std_logic_1164.ALL; entity name is   port (     end name; architecture architecture_name of name is   ( : signal); begin   -- 가   sync : process (clock, reset )   begin     VHDL statements for state elements   end process sync;   -- ( )   --   comb : process (     ( : variable)   begin   end process comb; end architecture_name;           </pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



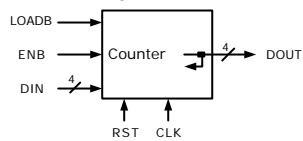
## 4-Bit Up-Counter

## VHDL Modeling

Counter Table

| RST | CLK | ENB | LOADB | DOUT   |
|-----|-----|-----|-------|--------|
| 0   | X   | X   | X     | "0000" |
| 1   |     | 1   | 0     | DIN    |
| 1   |     | 1   | 1     | DOUT+1 |

### Block Diagram



### VHDL

```

library IEEE; use IEEE.std_logic_1164.ALL;
use IEEE.std_logic_unsigned.all;
entity COUNTER is
port (RST,CLK,LOADB,ENB: in std_logic;
 DIN : in std_logic_vector(3 downto 0);
 DOUT : out std_logic_vector(3 downto 0));
end COUNTER;

```

```

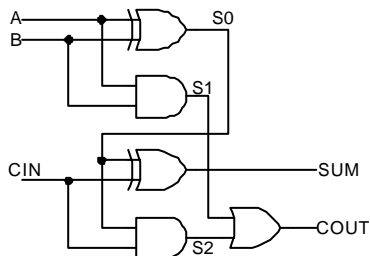
architecture RTL of COUNTER is
signal T1,T2: std_logic_vector(3 downto 0);
begin
A: process (RST, CLK) begin
if RST = '0' then
T1 <= (others => '0');
elsif CLK'event and CLK = '1' then
if ENB = '1' then
T1 <= T2;
end if;
end if;
end process;
--
B: process (LOADB, DIN, T1) begin
if LOADB='0' then
T2 <= DIN;
else
T2 <= T1 + 1;
end if;
end process;
--
DOUT <= T1;
end RTL;

```

### Gate-Level

- Boolean Equation
- Shift
- Arithmetic Logic
- 3-state Buffer

### 가



### Gate Logic Operator

### Wire Signal

### Signal Assign

```

library IEEE; use IEEE.std_logic_1164.ALL;
entity ADDER is
port (A, B, CIN : in std_logic;
 SUM, COUT : out std_logic);
end ADDER;
architecture RTL1 of ADDER is
-- Wire Signal
signal S0, S1, S2 : std_logic;
begin
-- Gate Logical Operator
S0 <= A and B;
S1 <= A and B;
SUM <= S0 xor CIN;
S2 <= S0 and CIN;
COUT <= S1 or S2;
end RTL1;

```

| Boolean Equation |                                                                                                                                                                                                                                                                                              | VHDL      |                                                                                                                                                                                                                                         |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 가                | Equation                                                                                                                                                                                                                                                                                     | Multi-Bit | Entity Unit                                                                                                                                                                                                                             |
|                  | $sum = \bar{a}b\bar{c}_{in} + a\bar{b}\bar{c}_{in} + \bar{a}b\bar{c}_{in} + abc_{in}$                                                                                                                                                                                                        |           | entity MULTIBIT is<br>port (A,B,C : IN<br>std_logic_vector(3 DOWNT0 0);<br>Z: OUT std_logic_vector(3 DOWNT0 0));<br>end MULTIBIT;                                                                                                       |
|                  | $cout = \bar{a}b\bar{c}_{in} + \bar{a}bc_{in} + a\bar{b}c_{in} + abc_{in}$                                                                                                                                                                                                                   | 2가        | Architecture Unit                                                                                                                                                                                                                       |
|                  | Boolean Equation Data Flow                                                                                                                                                                                                                                                                   |           | architecture RTL2 of MULTIBIT is<br>begin<br>--Multi-Bit<br>Z <= A or (B nand C);<br>end RTL2;                                                                                                                                          |
|                  | Signal Assign                                                                                                                                                                                                                                                                                |           | --<br>architecture RTL1 of MULTIBIT is<br>begin<br>-- Bit Equation<br>Z(3) <= A(3) nor (B(3) nand C(3));<br>Z(2) <= A(2) nor (B(2) nand C(2));<br>Z(1) <= A(1) nor (B(1) nand C(1));<br>Z(0) <= A(0) nor (B(0) nand C(0));<br>end RTL1; |
|                  | architecture RTL2 of ADDER is<br>begin<br>SUM <= (not A and not B and Cin) or<br>(not A and B and not Cin) or<br>(A and B and not Cin) or<br>(A and B and Cin);<br>COUT <= (not A and B and Cin) or<br>(A and not B and Cin) or<br>(A and B and not Cin) or<br>(A and B and C);<br>end RTL2; |           |                                                                                                                                                                                                                                         |

| Truth Table |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | VHDL              |     |      |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-----|------|-----|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 1-Bit 가     | Truth Table                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Architecture Unit |     |      |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
|             | <table border="1"> <thead> <tr> <th>a</th> <th>b</th> <th>cin</th> <th>sum</th> <th>cout</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </tbody> </table> | a                 | b   | cin  | sum | cout | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | architecture RTL2 of ADDER is<br>begin<br>process (A, B, CIN)<br>variable TMP :<br>std_logic_vector(2 downto 0);<br>begin<br>-- variable<br>TMP := A & B & CIN;<br>case TMP is<br>when "000" =><br>SUM <='0'; COUT <='0';<br>when "001"   "010"   "100" =><br>SUM <='1'; COUT <='0';<br>when "011"   "101"   "110" =><br>SUM <='0'; COUT <='1';<br>when others =><br>SUM <='1'; COUT <='1';<br>end case;<br>end process;<br>end RTL2; |  |
| a           | b                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | cin               | sum | cout |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 0           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 0                 | 0   | 0    |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 0           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 1                 | 1   | 0    |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 0           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 0                 | 1   | 0    |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 0           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 1                 | 0   | 1    |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 1           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 0                 | 1   | 0    |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 1           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 1                 | 0   | 1    |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 1           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 0                 | 0   | 1    |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 1           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 1                 | 1   | 1    |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
| 1 - Bit     | Variable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | &                 |     |      |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
|             | 가                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Case              |     |      |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
|             | Signal Assign                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                   |     |      |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |
|             | Variable Assign                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                   |     |      |     |      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                       |  |

## VHDL

### 4\*2 Priority Encoder

| a | b | c | d | enc out |
|---|---|---|---|---------|
| - | - | - | 0 | 00      |
| - | - | 0 | 1 | 01      |
| - | 0 | 1 | 1 | 10      |
| 0 | 1 | 1 | 1 | 11      |

```

Conditional Signal Assign
Process IF
Conditional Signal Assign
architecture RTL1 of PRIO_ENC is
begin
ENC_OUT <= "00" when D = '0' else
 "01" when C = '0' else
 "10" when B = '0' else
 "11";
end RTL1;

```

```

Process IF
가
1
architecture RTL2 of PRIO_ENC is
begin
process (A,B,C,D)
begin
if D='0' then
ENC_OUT <= "00";
elsif C='0' then
ENC_OUT <= "01";
elsif B='0' then
ENC_OUT <= "10";
else
ENC_OUT <= "11";
end if;
end process;
end RTL2;

```

## (1)

### Enable 4\*1 MUX

| ENABLE | SEL | Y      |
|--------|-----|--------|
| 0      | 00  | I0     |
| 0      | 01  | I1     |
| 0      | 10  | I2     |
| 0      | 11  | I3     |
| 1      | --  | "0000" |

```

Process Case
process(SEL,ENABLE,I0,I1,I2,I3) begin
if ENABLE='0' then
case SEL is
when "00" => Y <= I0;
when "01" => Y <= I1;
when "10" => Y <= I2;
when others => Y <= I3;
end case;
else
Y <= "0000";
end if;
end process;

```

```

Selected Signal Assign
architecture RTL2 of MUX41 is
signal TMP : std_logic_vector(1 downto 0);
begin
with SEL select
TMP <= I0 when "00",
 I1 when "01",
 I2 when "10",
 I3 when others;
with ENABLE select
Y <= TMP when '0',
 "0000" when others;
end RTL2;

```

```

MUX
MUX Component
, Structural Modeling
process
case
, Tool

```



(2)

Enable 2\*4 DEC

| Enable<br>G | DATA | Y      |
|-------------|------|--------|
| 0           | 00   | "1110" |
| 0           | 01   | "1101" |
| 0           | 10   | "1011" |
| 0           | 11   | "0111" |
| 1           | --   | "1111" |

```

Process
process(G,DATA) begin
 if G='0' then
 case DATA is
 when "00" => Y <= "1110";
 when "01" => Y <= "1101";
 when "10" => Y <= "1011";
 when others => Y <= "0111";
 end case;
 else
 Y <= "1111";
 end if;
end process;

```

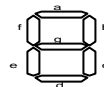
```

Selected Signal Assign
architecture RTL2 of EDECODER is
 signal TMP : std_logic_vector(2 downto 0);
begin
 TMP <= G & DATA;
 with TMP select
 Y <= "1110" when "000",
 "1101" when "001",
 "1011" when "010",
 "0111" when "011",
 "1111" when others;
end RTL2;

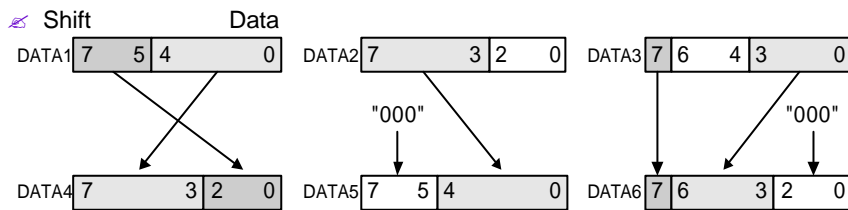
```

BCD-to-7 Segment Decoder

Don't Care  
VHDL



Shift Data VHDL



```

Basic Signal Assign & Operator
-- & Operator : Shift Rotate
-- DATA4(7 downto 3) <= DATA1(4 downto 0);
-- DATA4(2 downto 0) <= DATA1(7 downto 5);
DATA4 <= DATA1(4 downto 0) & DATA1(7 downto 5);
-- DATA5(7 downto 5) <= "000";
-- DATA5(4 downto 0) <= DATA2(7 downto 3);
DATA5 <= "000" & DATA2(7 downto 3);
-- DATA6(7) <= DATA3(7);
-- DATA6(6 downto 3) <= DATA3(3 downto 0);
-- DATA6(2 downto 0) <= "000";
DATA6 <= DATA3(7) & DATA3(3 downto 0) & "000";

```

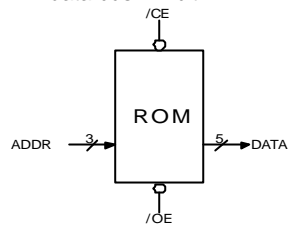
| Arithmetic                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | VHDL                     |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-----------|-----|---------|-----|-----------|-----|---------|-----|-------------|-----|------------|-----|---------|-----|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>8가 ALU</p> <table border="1" style="margin-left: 20px; border-collapse: collapse;"> <tr><td>000</td><td>op1 + op2</td></tr> <tr><td>001</td><td>op1 - 1</td></tr> <tr><td>010</td><td>op1 - op2</td></tr> <tr><td>011</td><td>op1 + 1</td></tr> <tr><td>100</td><td>op1 and op2</td></tr> <tr><td>101</td><td>op1 or op2</td></tr> <tr><td>110</td><td>not op1</td></tr> <tr><td>111</td><td>op1</td></tr> </table> <p>Entity Unit</p> <pre> library IEEE; use IEEE.std_logic_1164.ALL; use IEEE.std_logic_unsigned.all; entity SIM_ALU is port (OP1,OP2:in       std_logic_vector(7 downto 0);       CIN : in std_logic;       SEL : in std_logic_vector(2 downto 0);       ALU_OUT : out       std_logic_vector(7 downto 0);       ZERO_FLAG : out std_logic ); end SIM_ALU; </pre> | 000                      | op1 + op2 | 001 | op1 - 1 | 010 | op1 - op2 | 011 | op1 + 1 | 100 | op1 and op2 | 101 | op1 or op2 | 110 | not op1 | 111 | op1 | <p>Case</p> <pre> architecture RTL of SIM_ALU is begin process (OP1, OP2, CIN, SEL) variable TMP:std_logic_vector(7 downto 0); begin case SEL is when '000'=&gt;TMP:= OP1+OP2+CIN; when '001'=&gt;TMP:= OP1-1; when '010'=&gt;TMP:= OP1-OP2+CIN; when '011'=&gt;TMP:= OP1+1; when '100'=&gt;TMP:= OP1 and OP2; when '101'=&gt;TMP:= OP1 or OP2; when '110' =&gt;TMP:= not OP1; when others =&gt;TMP:= OP1; end case; ALU_OUT &lt;= TMP; if TMP=0 then ZERO_FLAG &lt;= '1'; else ZERO_FLAG &lt;= '0'; end if; end process; end RTL; </pre> |
| 000                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | op1 + op2                |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 001                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | op1 - 1                  |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 010                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | op1 - op2                |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 011                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | op1 + 1                  |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 100                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | op1 and op2              |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 101                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | op1 or op2               |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 110                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | not op1                  |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 111                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | op1                      |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <p>Univ. of Incheon System ASIC Design Lab.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <p>VHDL ASIC NO.-115</p> |           |     |         |     |           |     |         |     |             |     |            |     |         |     |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

| 3-State Buffer                                                                                                                                                                                                                                                                            | VHDL                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Resolved Signal</p> <p>2 Data Source Wire</p> <p>Resolved Signal</p> <p>3-State Buffer</p> <p>MUX</p> <p>Selected Signal Assign</p> <pre> architecture RTL1 of TRISTATE is begin TBUS&lt;= D1 when (Y1='1') else 'Z'; TBUS&lt;= D2 when (Y1='0' and Y2='1') else 'Z'; end RTL1; </pre> | <p>Process IF</p> <p>3-Buffer가 OFF 'Z'</p> <p>architecture RTL2 of TRISTATE is</p> <pre> begin process (Y1, D1) begin if (Y1='1') then TBUS&lt;=D1; else TBUS&lt;='Z'; end if; end process; process (Y1, Y2, D2) begin if (Y1='0' and Y2='1') then TBUS&lt;=D2; else TBUS&lt;='Z'; end if; end process; end RTL2; </pre> <p>Resolved Function</p> <p>'0' '1'</p> |
| <p>Univ. of Incheon System ASIC Design Lab.</p>                                                                                                                                                                                                                                           | <p>VHDL ASIC NO.-116</p>                                                                                                                                                                                                                                                                                                                                         |

## VHDL : ROM

### ROM

- ⌘ address bus : N-bit
- ⌘ data bus : M-bit



### Entity Unit

```

library IEEE; use IEEE.std_logic_1164.ALL;
use IEEE.std_logic_unsigned.ALL;
entity ROM is
 generic (N: integer := 3; M: integer := 5);
 port (
 ADDR: in std_logic_vector(N-1 downto 0);
 CE_N, OE_N: in std_logic;
 DATA: out std_logic_vector(M-1 downto 0));
end ROM;

```

### Architecture Unit

```

⌘ 2
⌘ ROM address Integer
architecture RTL of ROM is
 subtype WORD is
 std_logic_vector(M-1 downto 0);
 type TABLE is array(0 to 2**N-1) of WORD;
 -- type TABLE is array(0 to 2**N-1) of
 -- std_logic_vector(M-1 downto 0);
 constant ROM_DATA : TABLE :=
 TABLE("10101", "10000", "11111", "11011",
 "10001", "01100", "00101", "10011");
begin
 process (CE_N,OE_N,ADDR) begin
 if CE_N='0' and OE_N='0' then
 DATA <=
 ROM_DATA(conv_integer(ADDR));
 else
 DATA <= (others=>'Z');
 end if;
 end process;
end RTL;

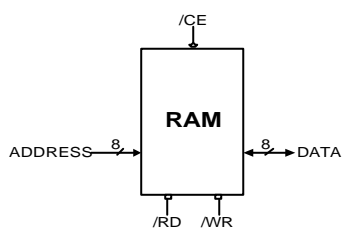
```



## VHDL : RAM

### RAM

- ⌘ address bus : 8-bit
- ⌘ data bus : 8-bit



### Entity Unit

```

library IEEE; use IEEE.std_logic_1164.ALL;
use IEEE.std_logic_unsigned.ALL;
entity RAM is
 port (
 ADDR: in std_logic_vector(7 downto 0);
 CE_N, RD_N, WR_N: in std_logic;
 DATA: inout
 std_logic_vector(7 downto 0));
end RAM;

```

### Architecture Unit

```


⌘ 2
⌘ RAM address Integer
⌘ Read Data Bus Bit
 'Z' : Resolved Signal
architecture RTL of RAM is
 subtype RAM_WORD is
 std_logic_vector(7 downto 0);
 type RAM_TABLE is array(0 to 255) of
 RAM_WORD;
 signal RAM_DATA : RAM_TABLE;
begin
 process (CE_N,RD_N,WR_N,ADDR,DATA)
 variable TMP : integer;
 begin
 DATA <= "ZZZZZZZZ";
 TMP := conv_integer(ADDR);
 if CE_N='0' and RD_N='0' then
 DATA <= RAM_DATA(TMP);
 elsif CE_N='0' and WR_N='0' then
 RAM_DATA(TMP) <= DATA;
 end if;
 end process;
end RTL;


```




| Latch                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | VHDL                                    |        |   |      |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|--------|---|------|---|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>✂ D-Latch</p> <table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="padding: 2px;">G</td> <td style="padding: 2px;">Q(t+1)</td> </tr> <tr> <td style="padding: 2px;">0</td> <td style="padding: 2px;">Q(t)</td> </tr> <tr> <td style="padding: 2px;">1</td> <td style="padding: 2px;">D</td> </tr> </table> <div style="text-align: center; margin-top: 10px;"> </div> <p>✂ Process IF</p> <pre>architecture RTL1 of DLATCH1 is begin   process (G, D) begin     if G='1' then       Q &lt;= D;       QBAR &lt;= not D;     end if;   end process; end RTL1;</pre> | G                                       | Q(t+1) | 0 | Q(t) | 1 | D | <p>✂ Signal Assign</p> <pre>architecture RTL2 of DLATCH1 is   signal TMP : std_logic; begin   TMP &lt;= D when G='1' else TMP;   Q &lt;= TMP;   QBAR &lt;= not TMP; end RTL2;</pre> <p>✂ Latch</p> <pre>architecture RTL of EX is   signal A, B, C, D, E:     std_logic_vector(1 downto 0); begin   process (C, D, E, EN) begin     if EN='1' then       A &lt;= C; B &lt;= D;     else       A &lt;= E;     end if;   end process; end RTL;</pre> |
| G                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Q(t+1)                                  |        |   |      |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Q(t)                                    |        |   |      |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | D                                       |        |   |      |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <p><small>Univ. of Incheon System ASIC Design Lab.</small></p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | <p><small>VHDL ASIC NO.-119</small></p> |        |   |      |   |   |                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

| Flip-Flop                                                                                                                                                                                                                                                                                | VHDL                                                                                                                                                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>✂ Synchronous D Flip-Flop</p> <div style="text-align: center; margin-top: 10px;"> </div> <p>✂ 가 VHDL</p> <pre>A1: process (CLK) begin   if (CLK='1' and CLK'event) then     Q &lt;= D;   end if; end process; A2: process begin   wait until CLK='1';   Q &lt;= D; end process;</pre> | <p>✂ Asynchronous D Flip-Flop</p> <div style="text-align: center; margin-top: 10px;"> </div> <p>✂ VHDL (Synopsys Model)</p> <pre>A1: process (RESET, CLK) begin   if RESET='1' then     DATA &lt;= '0';   elsif (CLK='1' and CLK'event) then     DATA &lt;= IN_DATA;   end if; end process;</pre> |
| <p><small>Univ. of Incheon System ASIC Design Lab.</small></p>                                                                                                                                                                                                                           | <p><small>VHDL ASIC NO.-120</small></p>                                                                                                                                                                                                                                                           |


| Flip-Flop                                                                                                                                                                                                                                                                                      | VHDL                                                                                                                                                                                                                                                                                                                                                                   | (1)               |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| <p style="text-align: center;">-1</p> <pre> architecture RTL1 of EX is   signal A, B: std_logic_vector(3 downto 0); begin   process (CLK)   begin     if (CLK='1' AND CLK'EVENT) then       if Q(3)/='1' then         Q &lt;= A + B;       end if;     end if;   end process; end RTL1; </pre> | <p style="text-align: center;">-2</p> <pre> architecture RTL2 of EX is   signal A, B: std_logic_vector(3 downto 0); begin   process (CLK)   variable INT:     std_logic_vector(3 downto 0);   begin     if (CLK='1' AND CLK'EVENT) then       if INT(3)/='1' then         INT := A + B;         Q &lt;= INT;       end if;     end if;   end process; end RTL2; </pre> |                   |
|                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                        | VHDL ASIC NO.-121 |

| Flip-Flop                                                                                                                                                                                                                                                                                              | VHDL                                                                                                                                                                                                                                                                             | (2)               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| <p style="text-align: center;">-3</p> <pre> architecture RTL3 of EX is begin   process (RESET_N, CLK)   begin     if (RESET_N='0') then       Q1 &lt;= '0'; Q2 &lt;= '0';     elsif (CLK='1' AND CLK'EVENT) then       Q1 &lt;= A and B;       Q2 &lt;= C;     end if;   end process; end RTL3; </pre> | <p style="text-align: center;">-4</p> <pre> architecture RTL4 of EX is   signal Q1, I2: std_logic; begin   process (CLK)   begin     if (CLK='1' AND CLK'EVENT) then       Q1 &lt;= D;       I2 &lt;= Q1 and A;     end if;   end process;   Q2 &lt;= I2 and B; end RTL4; </pre> |                   |
|                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                  | VHDL ASIC NO.-122 |

| Variable                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Signal Assignment | Flip-Flop                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>Signal Shift-Register</li> </ul> <pre>--4 Flip-Flop library IEEE; use IEEE.std_logic_1164.all; entity SHREG4 is   port ( D_IN, CLK : in std_logic;          D_OUT: out std_logic); end SHREG4; architecture RTL of SHREG4 is   signal S0, S1, S2: std_logic; begin   process (CLK) begin     if (CLK='1' and CLK'event) then       S0 &lt;= D_IN;  S1 &lt;= S0;       S2 &lt;= S1;   D_OUT &lt;= S2;     end if;   end process; end RTL;</pre> |                   | <ul style="list-style-type: none"> <li>Variable Shift-Register</li> </ul> <pre>--1 Flip-Flop process (CLK)   variable S0, S1, S2: std_logic; begin   if (CLK='1' and CLK'event) then     S0 := D_IN;  S1 := S0;     S2 := S1;   D_OUT &lt;= S2;   end if; end process; --4 Flip-Flop process (CLK)   variable S0, S1, S2: std_logic; begin   if (CLK='1' and CLK'event) then     D_OUT &lt;= S2;  S2 := S1;     S1 := S0;    S0 := D_IN;   end if; end process;</pre> |

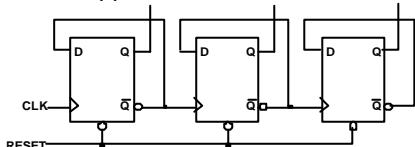
Univ. of Incheon System ASIC Design Lab.  VHDL ASIC NO.-123

| Register                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | VHDL  |        |        |            |     |   |   |   |   |            |   |   |   |   |        |   |   |   |   |     |   |   |   |      |     |   |   |   |      |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|--------|--------|------------|-----|---|---|---|---|------------|---|---|---|---|--------|---|---|---|---|-----|---|---|---|------|-----|---|---|---|------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>Multi-Bits           <ul style="list-style-type: none"> <li>Register</li> <li>Shift-Register</li> <li>FIFO</li> <li>Cycle Buffer</li> </ul> </li> <li>8-Bit Register</li> </ul> <table border="1"> <thead> <tr> <th>RESET</th> <th>LD</th> <th>ENABLE</th> <th>CLK</th> <th>REG</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>-</td> <td>-</td> <td>-</td> <td>"00000000"</td> </tr> <tr> <td>1</td> <td>1</td> <td>-</td> <td>-</td> <td>LDDATA</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>-</td> <td>REG</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>?(x)</td> <td>REG</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>?(o)</td> <td>REGIN</td> </tr> </tbody> </table> | RESET | LD     | ENABLE | CLK        | REG | 0 | - | - | - | "00000000" | 1 | 1 | - | - | LDDATA | 1 | 0 | 0 | - | REG | 1 | 0 | 1 | ?(x) | REG | 1 | 0 | 1 | ?(o) | REGIN | <pre>library IEEE; use IEEE.std_logic_1164.ALL; entity REGISTER1 is   port (RESET, LD, ENABLE, CLK : in         std_logic;         REGIN : in std_logic_vector(7 downto 0);         REG : out std_logic_vector(7 downto 0)); end REGISTER1; architecture RTL2 of REGISTER1 is begin   process (RESET, LD, CLK) begin     if RESET='0' then       REG &lt;= (others =&gt;'0');     elsif LD='1' then       REG &lt;= LDDATA;     elsif ENABLE='1' then       if (CLK='0' and CLK'event) then         REG &lt;= REGIN;       end if;     end if;   end process; end RTL2;</pre> |
| RESET                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | LD    | ENABLE | CLK    | REG        |     |   |   |   |   |            |   |   |   |   |        |   |   |   |   |     |   |   |   |      |     |   |   |   |      |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | -     | -      | -      | "00000000" |     |   |   |   |   |            |   |   |   |   |        |   |   |   |   |     |   |   |   |      |     |   |   |   |      |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 1     | -      | -      | LDDATA     |     |   |   |   |   |            |   |   |   |   |        |   |   |   |   |     |   |   |   |      |     |   |   |   |      |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 0     | 0      | -      | REG        |     |   |   |   |   |            |   |   |   |   |        |   |   |   |   |     |   |   |   |      |     |   |   |   |      |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 0     | 1      | ?(x)   | REG        |     |   |   |   |   |            |   |   |   |   |        |   |   |   |   |     |   |   |   |      |     |   |   |   |      |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 0     | 1      | ?(o)   | REGIN      |     |   |   |   |   |            |   |   |   |   |        |   |   |   |   |     |   |   |   |      |     |   |   |   |      |       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

Univ. of Incheon System ASIC Design Lab.  VHDL ASIC NO.-124

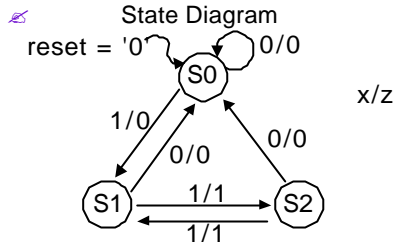
| Counter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | VHDL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>✍ Out Mode</p> <pre> library IEEE; use IEEE.std_logic_1164.ALL; use IEEE.std_logic_unsigned.all; entity CNT8 is   port (RESET,CLK: in std_logic;         CNT: out std_logic_vector(2 downto 0)); end CNT8; architecture RTL1 of CNT8 is   signal TMP1,TMP2:     std_logic_vector(2 downto 0); begin   CNT &lt;= TMP2;   TMP1 &lt;= TMP2 + '1';   process (RESET, CLK)   begin     if (RESET='0') then       TMP2 &lt;= "000";     elsif (CLK='1' and CLK'event) then       TMP2 &lt;= TMP1;     end if;   end process; end RTL1; </pre> | <p>✍ Buffer Mode</p> <p>Signal</p> <p>✍ Buffer Mode</p> <pre> library IEEE; use IEEE.std_logic_1164.ALL; use IEEE.std_logic_unsigned.all; entity CNT8 is   port (RESET,CLK: in std_logic;         CNT: buffer std_logic_vector(2 downto 0)); end CNT8; architecture RTL2 of CNT8 is begin   process (RESET, CLK)   begin     if (RESET='0') then       CNT &lt;= "000";     elsif (CLK='1' and CLK'event) then       CNT &lt;= CNT + '1';     end if;   end process; end RTL2; </pre> |



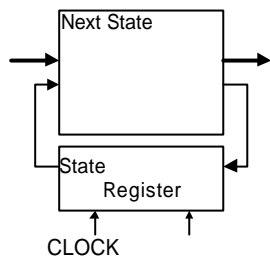
| Counter                                                                                                                                                                                                                                                                                                                                                                                                                                              | VHDL                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>✍ 8 Ripple Counter</p>  <p>✍ Generate</p> <p>✍ D Flip-Flop</p> <pre> architecture RTL1 of DFFR is   signal TMP_Q : std_logic; begin   Q &lt;= TMP_Q; QBAR &lt;= not TMP_Q;   process (RESET, CLK) begin     if (RESET='0') then       TMP_Q &lt;= '0';     elsif (CLK='1' and CLK'event) then       TMP_Q &lt;= D;     end if;   end process; end RTL1; </pre> | <p>✍ Generate</p> <pre> architecture RTL2 of RPLCNT8 is   -- Clock Wire   signal CNTB: std_logic_vector(3 downto 0);   -- D Flip-Flop Component   component DFFR     port (RESET, CLK, D : in std_logic;           Q, QBAR : out std_logic);   end component;   -- Component Specification   for U: use entity DFFR work.DFFR(RTL1); begin   CNTB(0) &lt;= CLK;   GEN1: for I in 0 to 2 generate     U: DFFR port map       (CLK =&gt; CNTB(I),        RESET =&gt; RESET,        D =&gt; CNTB(I+1),        Q =&gt; CNT(I),        QBAR =&gt; CNTB(I+1));   end generate GEN1; end RTL2; </pre> |



## Finite State Machine (1)



- FSM Hardware
  - Moore Type Machine
  - Mealy Type Machine



```

entity MEALYFSM is
 port (RESET, CLK, X : in std_logic;
 Z : out std_logic);
end MEALYFSM;

architecture RTL1 of MEALYFSM is
 --Symbolic State Name Type
 type STATE is (S0, S1, S2);
 --State Signal
 signal C_STATE, N_STATE : STATE;
begin

```

## Finite State Machine (2)

```

SYNC: process (RESET, CLK) begin
 if RESET='0' then
 C_STATE <= S0;
 elsif CLK='0' and CLK'event then
 C_STATE <= N_STATE;
 end if;
end process;

COMB : process (C_STATE, X) begin
 case C_STATE is
 when S0 =>
 Z <= '0';
 if X='0' then
 N_STATE <= S0;
 else
 N_STATE <= S1;
 end if;
 when S1 =>
 if X='0' then
 N_STATE <= S0; Z <= '0';
 else
 N_STATE <= S2; Z <= '1';
 end if;
 end case;
end process;

```

```

when others =>
 if X='0' then
 N_STATE <= S0; Z <= '0';
 else
 N_STATE <= S1; Z <= '1';
 end if;
end case;
end process;
end RTL1;

```

- State Code Assignment
  - Tool
  - Tool
    - Symbolic State Name
    - Simulation
      - Code Assign
      - Symbolic Name
    - Synthesis
      - Binary Code Assign
      - State Name
      - Binary Code Encoding

S0=(00), S1=(01), S2=(10)



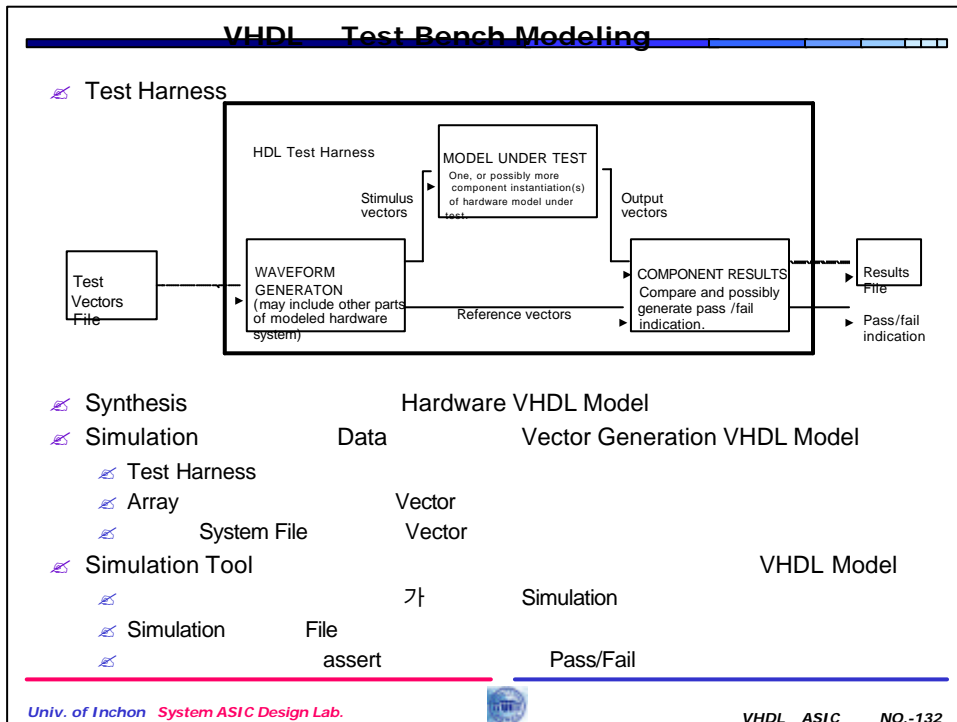
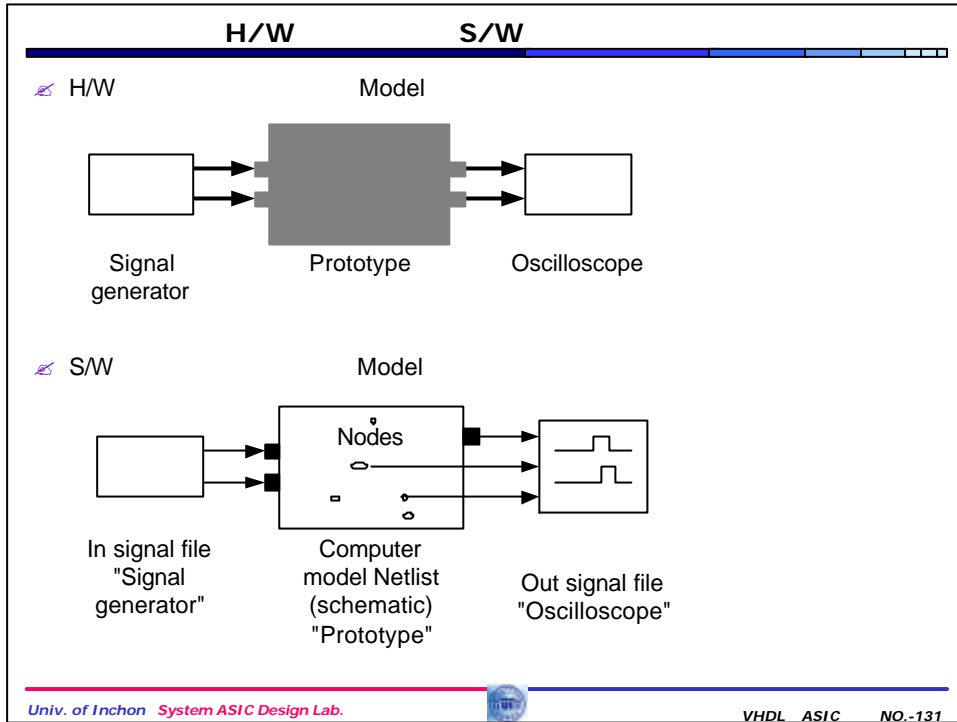
## Finite State Machine (3)

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>✎ State Code가</p> <p>    ✎ State Code Assignment      -1</p> <pre> library IEEE; use IEEE.std_logic_1164.ALL; entity MEALYFSM is   port (RESET, CLK, X : in std_logic;         Z : out std_logic); end MEALYFSM; architecture RTL2 of MEALYFSM is   constant S0: std_logic_vector(1 downto 0)     := "00";   constant S1: std_logic_vector(1 downto 0)     := "01";   constant S2: std_logic_vector(1 downto 0)     := "11";   signal C_STATE, N_STATE :     std_logic_vector(1 downto 0); begin   --      VHDL Code end RTL2;                 </pre> | <p>✎ State Code Assignment      -2</p> <pre> library IEEE; use IEEE.std_logic_1164.ALL; entity MEALYFSM is   port (RESET, CLK, X : in std_logic;         Z : out std_logic); end MEALYFSM; architecture RTL3 of MEALYFSM is   --Symbolic State Name      Type   type STATE is (S0, S1, S2);   attribute ENUM_ENCODING: string;   attribute ENUM_ENCODING of STATE:     type is "00 01 11";   -- State Signal   signal C_STATE, N_STATE : STATE; begin   --      VHDL Code end RTL3;                 </pre> |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



## Chap. 4 VHDL Bench Model

## Test

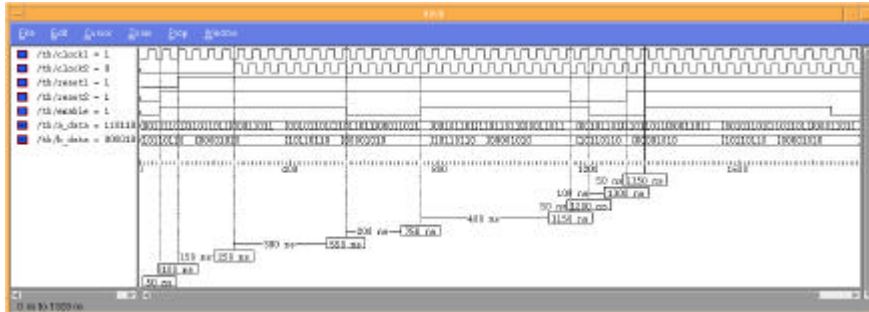


| Test Vector                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 37   VHDL                                                                                                                                                                                      |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Case1 :</b></p> <pre> signal clock, reset : std_logic := '0'; begin clock &lt;= not clock after 20 ns; reset &lt;= '1' after 120 ns; </pre> <p><b>case2 : Random</b> (1)</p> <pre> x &lt;= "00", "01" after 50 ns, "10" after 100 ns, "11" after 200 ns, "10" after 220 ns; </pre> <p><b>case3 : Random</b> ( )</p> <pre> process begin a &lt;= '0'; b &lt;= '1'; wait for 50 ns; a &lt;= '1'; wait for 70 ns; b &lt;= '0'; wait for 100 ns; a &lt;= '0'; wait for 50 ns; end process; </pre> | <p style="text-align: center;"><b>Clock</b></p> <p style="text-align: center;"><b>reset</b></p> <p style="text-align: center;"><b>clock</b></p> <p style="text-align: center;"><b>time</b></p> |
| <p>Univ. of Incheon System ASIC Design Lab.  VHDL ASIC NO.-133</p>                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                |

| Test Vector                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Modeling                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> --9-value visibility library IEEE; use IEEE.std_logic_1164.ALL; --entity unit entity TB is port (CLOCK1 : buffer std_logic := '0';       CLOCK2, RESET1, RESET2 : out std_logic ;       ENABLE : out std_logic;       A_DATA, B_DATA : out           std_logic_vector(7 downto 0)); end TB; --architecture unit architecture TB_A of TB is begin CLOCK1 &lt;= not CLOCK1 after 20 ns; RESET1 &lt;= '0', '1' after 100 ns; process begin if now = 0 ns then CLOCK2 &lt;= '0'; wait for 250 ns; else CLOCK2 &lt;= '1'; wait for 20 ns; CLOCK2 &lt;= '0'; wait for 20 ns; end if; end process; </pre> | <pre> process begin A_DATA &lt;= "00101101"; B_DATA &lt;= "10110110"; wait for 120 ns; A_DATA &lt;= "11011011"; wait for 30 ns; B_DATA &lt;= "00001010"; wait for 90 ns; A_DATA &lt;= "00011011"; wait for 150 ns; end process; process begin if now = 0 ns then RESET2 &lt;= '0'; ENABLE &lt;= '0'; wait for 50 ns; else RESET2 &lt;= '1'; ENABLE &lt;= '1'; wait for 300 ns; ENABLE &lt;= '0' after 200 ns, '1' after 400 ns; wait for 800 ns; RESET2 &lt;= '0', '1' after 150 ns; ENABLE &lt;= '0' after 50 ns; wait for 200 ns; end if; end process; end TB_A; </pre> |
| <p>Univ. of Incheon System ASIC Design Lab.  VHDL ASIC NO.-134</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## Test Vector      Waveform

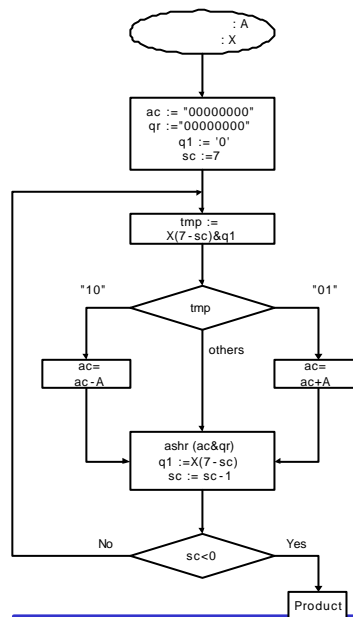
Test Vector      Waveform



- ☞ CLOCK1, CLOCK2
- ☞ RESET1, RESET2      가
- ☞ A\_DATA, B\_DATA      가
- ☞ Process      Data 가

## -1 : Booth Multiplier

| Booth Multiplier |                 | Algorithm           |
|------------------|-----------------|---------------------|
| A                | 1 0 1 1         | -5                  |
| X                | x 1 1 0 1       | -3                  |
| Y                | 0 / 1 1 / 1     | recorded multiplier |
| Add -A           | 0 1 0 1         |                     |
| Shift            | 0 0 1 0 1       |                     |
| Add A            | + 1 0 1 1       |                     |
|                  | 1 1 0 1 1       |                     |
| Shift            | 1 1 1 0 1 1     |                     |
| Add -A           | + 0 1 0 1       |                     |
|                  | 0 0 1 1 1 1     |                     |
| Shift            | 0 0 0 1 1 1 1   |                     |
| Shift            | 0 0 0 0 1 1 1 1 |                     |



## - 1 : Booth Multiplier      VHDL

```

library IEEE; use IEEE.std_logic_1164.ALL;
use IEEE.numeric_std.ALL;
--
entity BOOTH is
generic (N : integer := 8);
port (RESET, CLOCK, LOAD : in std_logic;
 MULTIPLICAND, MULTIPLIER : in
 std_logic_vector(N-1 downto 0);
 PRODUCT : out
 std_logic_vector(2*N-1 downto 0));
end BOOTH;
--
architecture RTL of BOOTH is
signal Q1 : std_logic;
signal AC, BR : std_logic_vector(N-1 downto 0);
signal QR : std_logic_vector(N-1 downto 0);
signal SC : integer;
begin
process (RESET, CLOCK)
variable TMP_AC : std_logic_vector(N-1 downto 0);
begin
if RESET='0' then
BR <= (OTHERS => '0');
QR <= (OTHERS => '0');
AC <= (OTHERS => '0');
Q1 <= '0';
SC <= N;
PRODUCT <= (OTHERS => '0');
elsif (CLOCK='1' AND CLOCK'event) then
if LOAD='1' then
BR <= MULTIPLICAND;
QR <= MULTIPLIER;
AC <= (OTHERS => '0');
Q1 <= '0';
SC <= N;
PRODUCT <= (OTHERS => '0');
else
if (SC=0) then
PRODUCT <= AC & QR;
else
if QR(0)='0' and Q1='1' then
TMP_AC := AC + BR;
elsif QR(0)='1' and Q1='0' then
TMP_AC := AC + not BR + '1';
else
TMP_AC := AC;
end if;
end if;
Q1 <= QR(0);
QR <= TMP_AC(0) & QR(N-1 downto 1);
AC <= TMP_AC(N-1) & TMP_AC(N-1 downto 1);
SC <= SC - 1;
end if;
end if;
end process;
end RTL;
--

```



## - 1 : Test Bench

```

Vector
library IEEE; use IEEE.std_logic_1164.ALL;
library IEEE.numeric_std.ALL;
entity VEC_GEN is
port (CLOCK, RESET, LOAD : out std_logic;
 A_PORT, B_PORT : out std_logic_vector(7 downto 0));
end VEC_GEN;
architecture VEC_GEN_A of VEC_GEN is
signal CLOCK_S : std_logic := '0';
begin
RESET <= '1' after 50 ns;
CLOCK_S <= not CLOCK_S after 2 ns;
CLOCK <= CLOCK_S;
LOAD <= '0', '1' after 20 ns, '0' after 70 ns,
 '1' after 210 ns, '0' after 250 ns,
 '1' after 390 ns, '0' after 430 ns,
 '1' after 570 ns, '0' after 610 ns;
A_PORT_DRV : process begin
A_PORT <= "00000101"; wait for 200 NS;
A_PORT <= "11111011"; wait for 400 NS;
end process;
B_PORT_DRV : process begin
B_PORT <= "00000011"; wait for 400 NS;
B_PORT <= "11111101"; wait for 400 NS;
end process;
end VEC_GEN_A;

Test Bench
library IEEE; use IEEE.std_logic_1164.ALL;
use IEEE.numeric_std.ALL;
entity TB_BOOTH1 is end TB_BOOTH1;
architecture TB_BOOTH1_A of TB_BOOTH1 is
component BOOTH
generic (n : integer := 8);
port (RESET, CLOCK, LOAD : in std_logic;
 MULTIPLICAND, MULTIPLIER : in
 std_logic_vector(n-1 downto 0);
 PRODUCT : out
 std_logic_vector(2*n-1 downto 0));
end component;
-- Vector Generation Block
signal A_PORT, B_PORT : std_logic_vector(7 downto 0);
signal MUL_OUT : std_logic_vector(15 downto 0);
signal RESET, CLOCK, LOAD : std_logic;
for u0 : BOOTH use entity work.BOOTH(RTL);
for u1 : VEC_GEN use entity
work.VEC_GEN(VEC_GEN_A);
begin
u0 : BOOTH port map(RESET, CLOCK, LOAD,
A_PORT, B_PORT, MUL_OUT);
u1 : VEC_GEN port map (CLOCK, RESET, LOAD,
A_PORT, B_PORT);
end TB_BOOTH1_A;

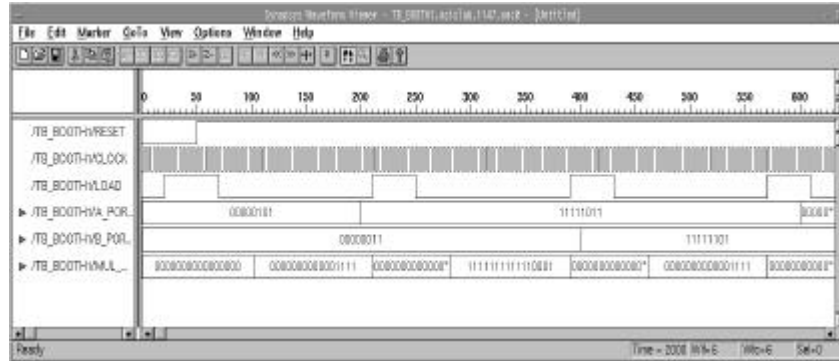
```



## - 1 : Test Bench VHDL

## Simulation

### VHDL Functional Simulation



- Simulation -1 :  $(+5) = (00000101)_2$ ,  $(+3) = (00000011)_2$   
 $= (00000000\_00001111)_2$
- Simulation -2 :  $(-5) = (11111011)_2$ ,  $(+3) = (00000011)_2$   
 $= (11111111\_11110001)_2$
- Simulation -3 :  $(-5) = (11111011)_2$ ,  $(-3) = (11111101)_2$
- Simulation -4 :  $(+5) = (00000101)_2$ ,  $(-3) = (11111101)_2$

## - 1 : ASSERT

## Test Bench

### Compare

```

library IEEE; use IEEE.std_logic_1164.ALL;
use IEEE.numeric_std.ALL;
entity COMP is
 port (RESET, LOAD : in std_logic;
 A_PORT, B_PORT : in signed(7 downto 0);
 MUL_OUT : in signed(15 downto 0));
end COMP;
architecture COMP_A of COMP is
 constant CLOCK_PERIOD : time := 5 ns;
begin
 process begin
 wait on RESET, LOAD;
 if RESET='0' then
 wait for (CLOCK_PERIOD);
 assert (MUL_OUT=0) report "assert violation-1";
 elsif LOAD='1' then
 wait for (CLOCK_PERIOD);
 assert (MUL_OUT=0) report "assert violation-2";
 else
 wait until (LOAD='0' and load'event);
 wait on MUL_OUT;
 assert (MUL_OUT=(A_PORT*B_PORT))
 report "assert violation-3";
 end if;
 end process;
end COMP_A;

```

### Test Bench

```

library IEEE; use IEEE.std_logic_1164.ALL;
use IEEE.numeric_std.ALL;
entity TB_BOOTH2 is end TB_BOOTH2;
architecture TB_BOOTH2_A of TB_BOOTH2 is
 -- Compare Component
 component COMP
 port (RESET, LOAD : in std_logic;
 A_PORT, B_PORT : in signed(7 downto 0);
 MUL_OUT : in signed(15 downto 0));
 end component;
 signal A_PORT, B_PORT : signed(7 downto 0);
 signal MUL_OUT : signed(15 downto 0);
 signal RESET, CLOCK, LOAD : std_logic;
 for u0 : BOOTH use entity work.BOOTH(RTL);
 for u1 : VEC_GEN use entity
 work.VEC_GEN(VEC_GEN_A);
 for u2 : COMP use entity work.COMP(COMP_A);
begin
 u0 : BOOTH port map(RESET, CLOCK, LOAD,
 A_PORT, B_PORT, MUL_OUT);
 u1 : VEC_GEN port map (CLOCK, RESET, LOAD,
 A_PORT, B_PORT);
 u2 : COMP port map (RESET, LOAD, A_PORT,
 B_PORT, MUL_OUT);
end TB_BOOTH2_A;

```

## - 1 : File

## Test Bench

### Test Bench

```
library IEEE; use IEEE.std_logic_1164.ALL;
use IEEE.std_logic_signed.ALL;
use IEEE.std_logic_textio.ALL; use std.textio.ALL;
entity TB_BOOTH3 is end TB_BOOTH3;
architecture TB_BOOTH3_A of TB_BOOTH3 is
 component BOOTH
 generic (n : integer := 8);
 port (RESET, CLOCK, LOAD : in std_logic;
 MULTIPLICAND, MULTIPLIER : in
 std_logic_vector(n-1 downto 0);
 PRODUCT: out std_logic_vector(2*n-1 downto
 0));
 end component;
 signal A_PORT, B_PORT: std_logic_vector(7 downto
 0);
 signal MUL_OUT: std_logic_vector(15 downto 0);
 signal RESET, CLOCK, LOAD: std_logic := '0';
 for u0 : BOOTH use entity work.BOOTH(RTL);
begin
 RESET <= '1' after 50 ns;
 CLOCK <= not CLOCK after 2 ns;
 u0 : BOOTH port map(RESET, CLOCK, LOAD,
 A_PORT, B_PORT, MUL_OUT);
```

```
process
 file VECTOR_FILE : text is in "vectors";
 variable LOAD_V : std_logic;
 variable A_PORT_V, B_PORT_V :
 std_logic_vector(7 downto 0);
 variable DELAY : time;
 variable L : line;
 variable GOOD_NUMBER : boolean;
begin
 while not endfile(VECTOR_FILE) loop
 readline (VECTOR_FILE, L);
 read (L, DELAY, GOOD => GOOD_NUMBER);
 next when not GOOD_NUMBER;
 READ(L, LOAD_V);
 READ(L, A_PORT_V);
 READ(L, B_PORT_V);
 WAIT FOR delay - now;
 LOAD <= LOAD_V;
 A_PORT <= A_PORT_V;
 B_PORT <= B_PORT_V;
 end loop;
 wait;
end process;
```



## - 1 : File

## Test Bench

```
process
 file TABLE_FILE : text is out "results";
 variable TABLE_LINE : line;
begin
 wait on MUL_OUT;
 write (TABLE_LINE, now, RIGHT, 10, ns);
 write (TABLE_LINE, '');
 write (TABLE_LINE, MUL_OUT(15));
 write (TABLE_LINE, MUL_OUT(14));
 write (TABLE_LINE, MUL_OUT(13));
 write (TABLE_LINE, MUL_OUT(12));
 write (TABLE_LINE, MUL_OUT(11));
 write (TABLE_LINE, MUL_OUT(10));
 write (TABLE_LINE, MUL_OUT(9));
 write (TABLE_LINE, MUL_OUT(8));
 write (TABLE_LINE, MUL_OUT(7));
 write (TABLE_LINE, MUL_OUT(6));
 write (TABLE_LINE, MUL_OUT(5));
 write (TABLE_LINE, MUL_OUT(4));
 write (TABLE_LINE, MUL_OUT(3));
 write (TABLE_LINE, MUL_OUT(2));
 write (TABLE_LINE, MUL_OUT(1));
 write (TABLE_LINE, MUL_OUT(0));
 writeln (TABLE_FILE, TABLE_LINE);
end process;
end TB_BOOTH3_A;
```

### Stimulus Vector File : "vectors"

|        |                     |
|--------|---------------------|
| 0 NS   | 0 00000101 00000011 |
| 20 NS  | 1 00000101 00000011 |
| 70 NS  | 0 00000101 00000011 |
| 200 NS | 0 11111011 00000011 |
| 210 NS | 1 11111011 00000011 |
| 250 NS | 0 11111011 00000011 |
| 390 NS | 1 11111011 00000011 |
| 400 NS | 1 11111011 11111101 |
| 430 NS | 0 11111011 11111101 |
| 570 NS | 1 11111011 11111101 |
| 600 NS | 1 00000101 11111101 |
| 610 NS | 0 00000101 11111101 |
| 800 NS | 0 11111011 00000011 |

### File : "results"

|        |                  |
|--------|------------------|
| 0 NS   | 0000000000000000 |
| 106 NS | 0000000000001111 |
| 214 NS | 0000000000000000 |
| 286 NS | 1111111111110001 |
| 394 NS | 0000000000000000 |
| 466 NS | 0000000000001111 |
| 574 NS | 0000000000000000 |
| 646 NS | 1111111111110001 |



# - 1 : FPGA

# Prototyping

Booth Multiplier

FPGA

Board

