

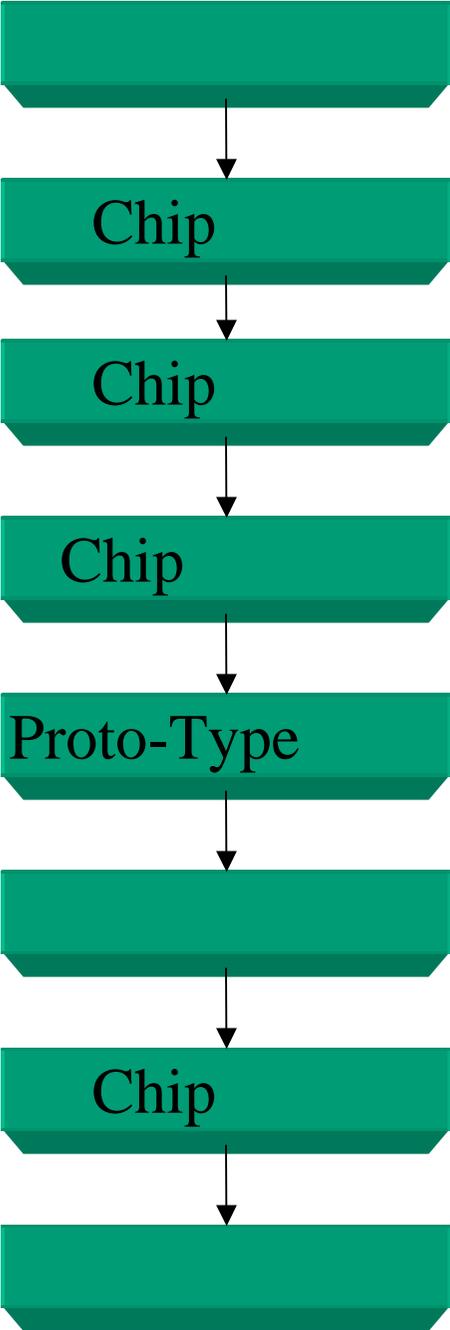
Chapter 1

Introduction to VHDL

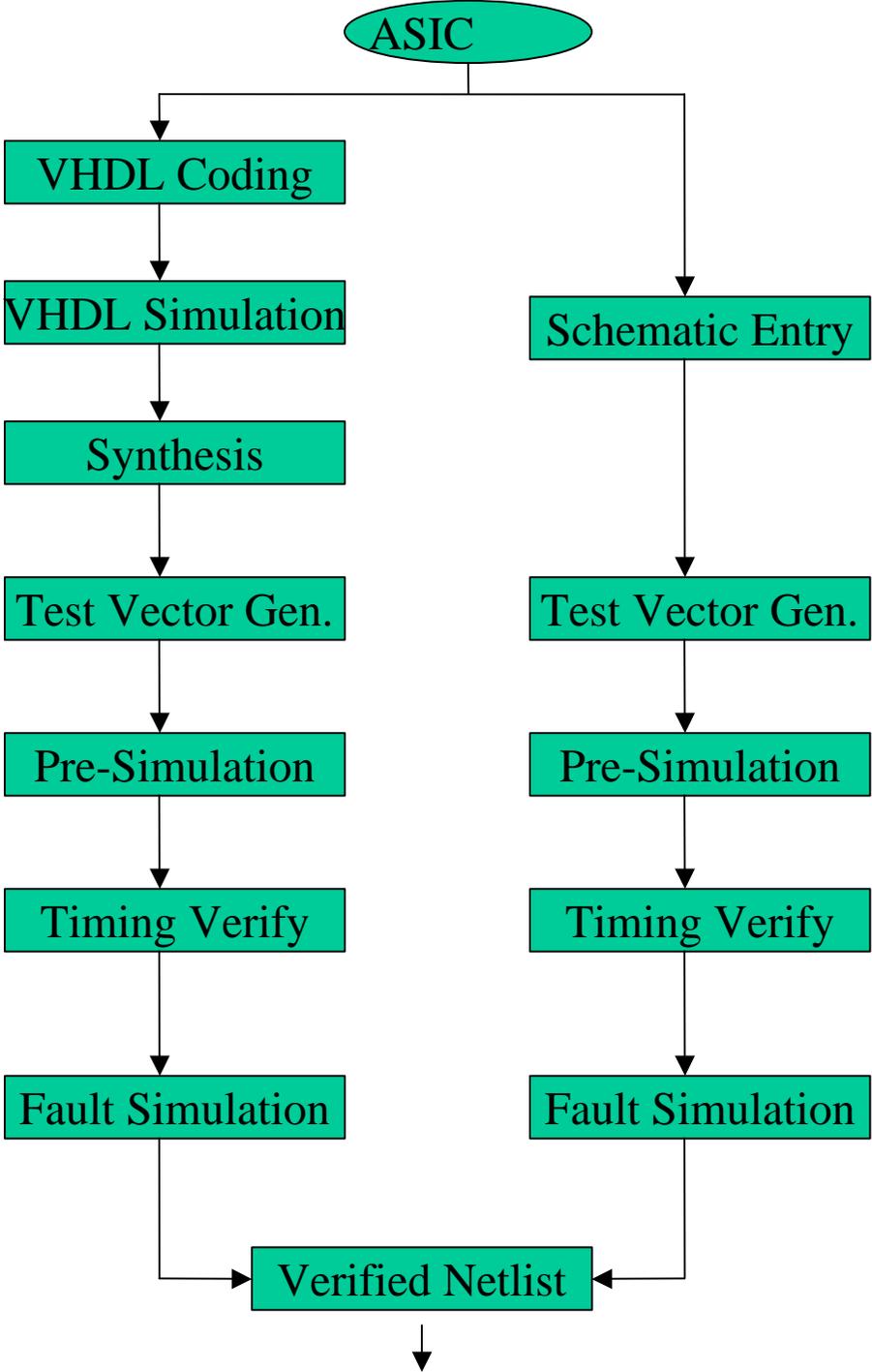
Chapter 2

Behavioral Modeling

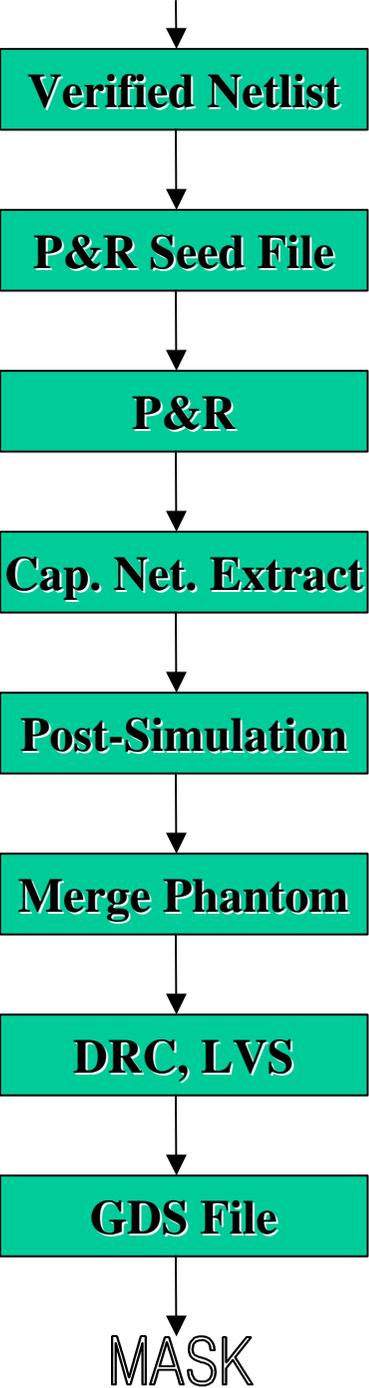
ASIC



ASIC Design Flow(Front-End)



ASIC Design Flow(Back-End)



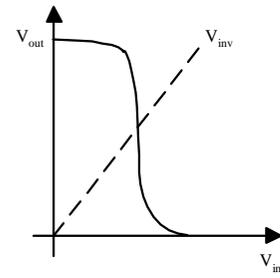
1.

```
#include <stdio.h>
main()
{
  int input,output;
  ...
  output=!input;
  ...
}
```

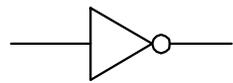
(a) C

0	1
1	0

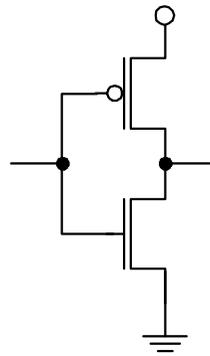
(b)



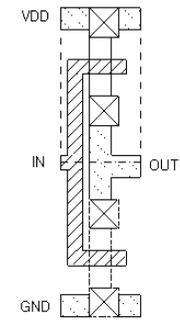
(c)



(d)

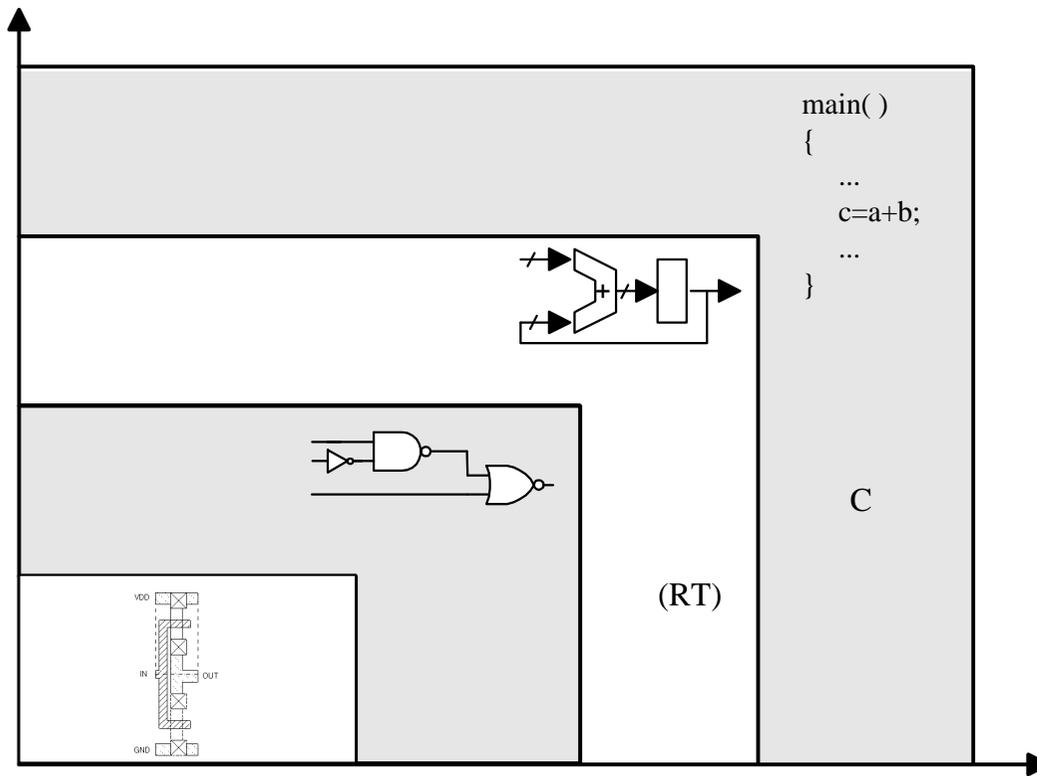


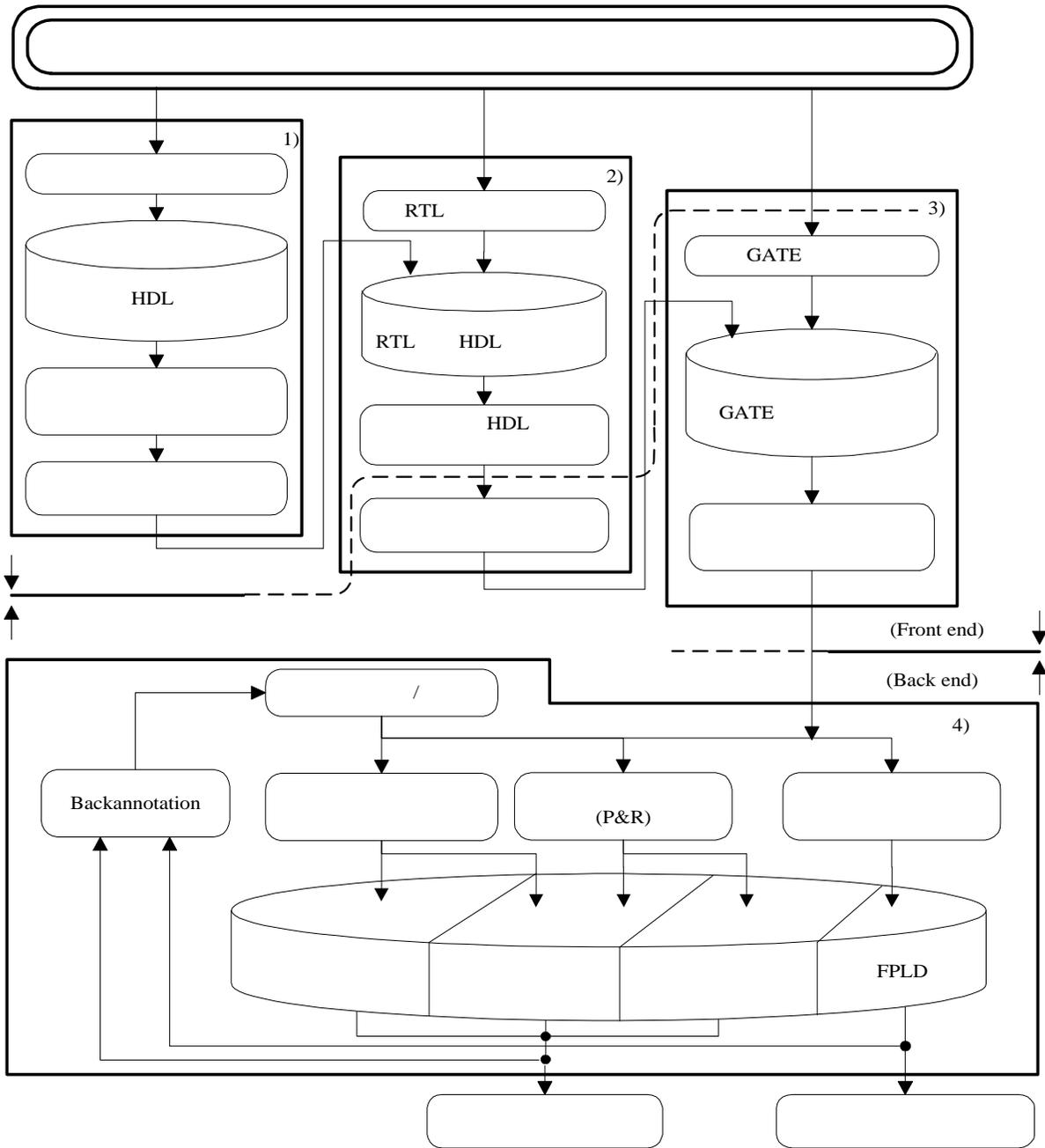
(e)



(f)

2.





3. Introduction to VHDL

VHDL

VHDL = Very High Speed Integrated Circuit
Hardware Description Language ()

1981	6	VHSIC Workshop(Woods Hole, Massachusetts) VHDL
1983	7	Intermetrics, IBM, Texas Instruments (HDL)
1985	8	VHDL Version 7.2
1986	3	IEEE Computer Society VHDL Analysis & Standardization Group (VASG) VHDL
1986	7	Intermetrics 가
1987	12	IEEE VHDL IEEE Std. 1076-1987
1990	6	VASG VHDL
1993	9	IEEE VHDL IEEE Std. 1076-1993

VHDL Language Reference Manual(LRM)

□ VHDL (Computer Language)

- , 가 .
 - Alphabet , .
- (Comment) '--'
 - , ('--') Line
- Line ';' .
 - ';' 가 .
- Space(' ') Tab .
 - , (:'; ' ' ; '+') .
- Data Type .
 - User가 Data Type
- Synthesizable Coding VHDL Simulation .
 - Synthesizable Coding RTL Coding .
 - Verilog HDL .

VHDL

(Design Unit)	
(Design Library)	
(Package Declaration) / (Package Body)	
(Entity Declaration)	
(Architecture Body)	
(Configuration Declaration)	



(Design Unit)

	(Primary Unit)	(Secondary Unit)
	Package Declaration	Package Body
/	Entity Declaration	Architecture Body
	Configuration Declaration	

- (Design Library)

-

- Working Library :
- Resource Library :

-

(Use)

- STD.STANDARD default 가 .

-

- STD library : VHDL standard library (STANDARD, TEXTIO)
- IEEE library : IEEE standard library (std_logic_1164, std_logic_arith)
- ASIC vendor library : logic gate entity, architecture body
- User-defined library : package, entity, architecture body, configuration
- WORK library : directory

- VHDL

-

LIBRARY *library_name* ;

-

(Use Clause)

USE *library_name.package_name.declaration_name* ;

USE *library_name.package_name.ALL* ;

-

/

(Binding Indication)

USE ENTITY *library_name.entity_name(architecture_body_name)* ;

USE CONFIGURATION *library_name.configuration_name* ;

- Entity

- Component , Type .

- Architecture Data Signal, Data Type

```

Entity COMP is
  port(
    a, b: in bit;
    c: out bit
  );
end COMP;
  
```

- Port : Entity Design Block 가 in, out, inout, buffer, linkage가 .

- . In : Input 가
 - . Out : Output Read() .
 - . Inout: Bidirection (가) .
 - . Buffer: Out 가 Read가 가 (가)
 - . linkage: 가

-

(Mode)

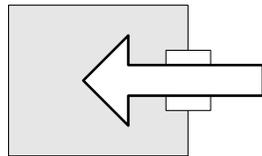
IN :

OUT :

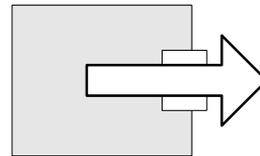
INOUT :

BUFFER :

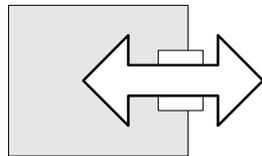
LINKAGE : VHDL



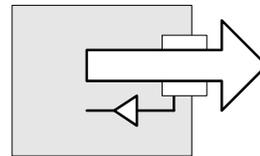
(IN)



(OUT)



(INOUT)



(BUFFER)

- Architecture Body()
 - Architecture
 - Entity Port
 - Entity Architecture 가 .

(1) Behavioral Style Architecture :

- . Function
- . Process .
- . if, for, case .
- . Design, Sequential Logic .

```

Architecture BEHAVE of COMP is
begin
  process(a, b)
  begin
    if (a = b) then
      c <= '1' ;
    else c <= '0' ;
    end if ;
  end process ;
end BEHAVE;

```

(2) Dataflow Style Architecture : RTL(Register Transfer Level) Design

```
Architecture RTL of COMP is
begin
  c <= not (a xor b);
end RTL;
```

(3) Structural Style Architecture

- . Netlist
- . Block Component
- . Top Level .

```
Architecture STRUCTURE of COMP is
signal I : bit ;
component XOR2 port (x , y : in bit; z : out bit);
end component;
component INV port(x : in bit; z : out bit);
end component;

begin
  u0 : XOR2 port map (x => a, y => b, z => I);
  u1 : INV port map (x => I, z => c);
end;
```

- Configuration Declaration

- VHDL File Design Unit(Design Block)
- Top level .
- Configuration Entity Architecture Component
- Entity, Architecture 가 Simulator
- Link .
- Test Bench Coding .

```

.
.
Architecture arch_add of adder is
  signal a, b, p : bit ;
  for u0 : and2 use entity work.and(comp) ;
begin
  u0: and2 port map(a, b, p) ;
.
.
.

```

❑ Reserved Word

abs access after alias all allow and architecture array assert
attribute begin block body buffer bus case component
configuration constant disconnect downto element else elsif end
entity exit file for function generate generic group guarded
if impure in initial inout is label library linkage literal loop
map mod nand new next nor not null of on open or
others out package port postponed private procedure process
pure range record register reject rem report return rol ror
select severity signal shared sla sll sra srl subtype
then to transport type unaffected units until use
variable wait when while with xnor xor

VHDL Terms

- Entity
- Architecture
- Configuration
- Package – a collection of commonly used data types and subprograms
- Bus – group of signal or particular method of communication
- Driver – source on a signal
- Attribute – data that is attached VHDL objects or predefined data about VHDL objects
- Generic – VHDL's term for a parameter that passes information to an entity
- Process – basic unit of execution in VHDL

Traditional Design Methods

**Figure 1-1 : Example of a symbol used in a schematic
(2-input, 2-output)**

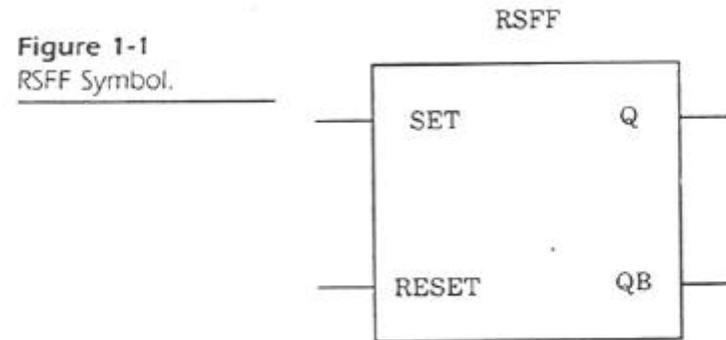
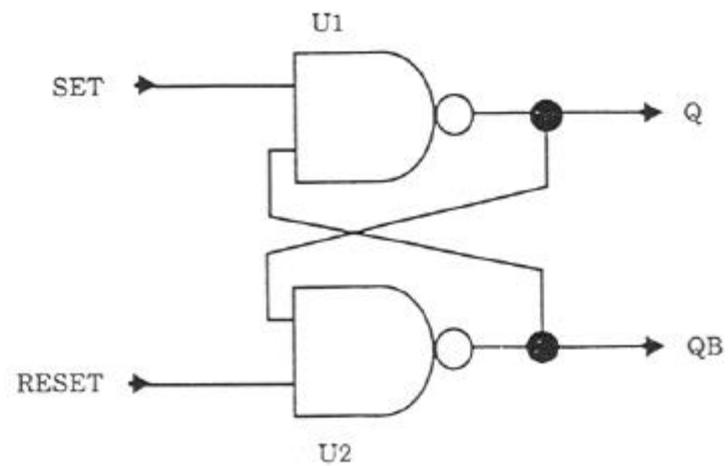


Figure 1-2 : Example of a schematic for an RSFF

Figure 1-2
RSFF Schematic.



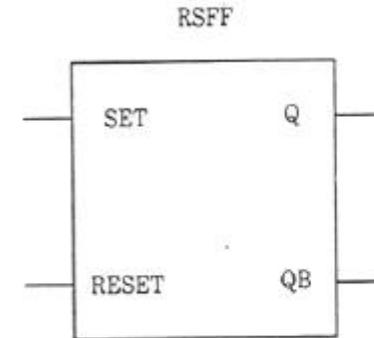
Symbol vs Entities

Entity rsff IS

**PORT (set, reset : IN BIT;
q, qb: INOUT BIT);**

END rsff;

Figure 1-1
RSFF Symbol.



Schematic vs Architectures (Structural Model)

ARCHITECTURE netlist OF rsff IS

COMPONENT nand2

**PORT (a, b : IN BIT;
c : OUT BIT);**

END COMPONENT;

BEGIN

U1 : nand2

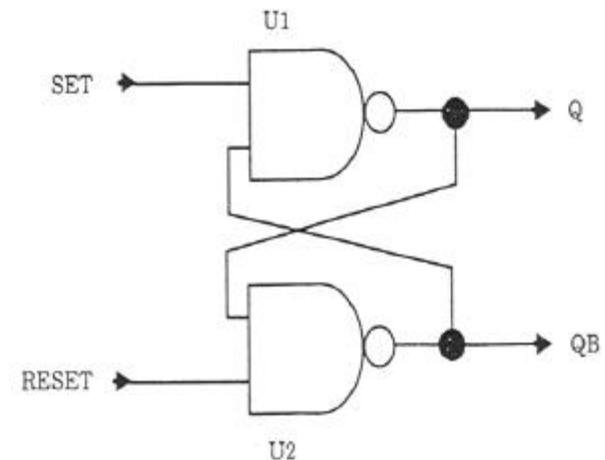
PORT MAP (set, qb, q);

U2 : nand2

PORT MAP (reset, q, qb);

END netlist;

Figure 1-2
RSFF Schematic.



Behavioral Descriptions

(Execute parallel(concurrently, not serially))

ARCHITECTURE behave OF rsff IS

BEGIN

q <= NOT(qb AND set) AFTER 2 ns;

qb <= NOT(q AND reset) AFTER 2 ns;

END behave;

Sequential Behavior

```
ARCHITECTURE sequential OF rsff IS
BEGIN
PROCESS (set, reset)
BEGIN
  IF set = '1' AND reset = '0' THEN
    q <= '0' AFTER 2 ns;
    qb <= '1' AFTER 4 ns;
  ELSIF set = '0' AND reset = '1' THEN
    q <= '1' AFTER 4 ns;
    qb <= '0' AFTER 2 ns;
  ELSIF set = '0' AND reset = '0' THEN
    q <= '1' AFTER 2 ns;
    qb <= '1' AFTER 2 ns;
  END IF;
END PROCESS;
END sequential;
```

Configuration Statement

(Architecture to use in a given simulation.

You can create simulatable model)

```
CONFIGURATION rsffcon1 OF rsff IS
```

```
FOR netlist
```

```
FOR U1, U2 : nand2 USE ENTITY WORK.mynand(version1);
```

```
END FOR;
```

```
END FOR;
```

```
END rsffcon1;
```

WORK: library name

mynand: entity

version1: architecture

4. Behavioral Modeling

o Most basic form of behavioral modeling in VHDL is signal assignment statement

a <= b;

a <= b after 10ns;

o AND gate modeling

ENTITY and2 IS

**PORT (a, b : IN BIT;
 c : OUT BIT);**

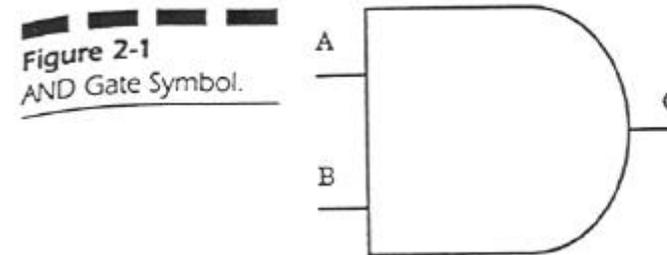
END and2;

ARCHITECTURE and2_behav OF and2 IS

BEGIN

c <= a AND b AFTER 5 ns;

END and2_behav;



o Four-input Multiplexer modeling

```
LIBRARY IEEE;  
USE IEEE.std_logic_1164.ALL;  
  
ENTITY mux4 IS  
PORT ( i0, i1, i2, i3, a, b : IN std_logic;  
       q : OUT std_logic);  
  
END mux4;
```

```
ARCHITECTURE mux4 OF mux4 IS  
SIGNAL sel : INTEGER;  
BEGIN  
WITH sel SELECT  
  q <= i0 AFTER 10 ns WHEN 0,  
    i1 AFTER 10 ns WHEN 1,  
    i2 AFTER 10 ns WHEN 2,  
    i3 AFTER 10 ns WHEN 3,  
    'X' AFTER 10 ns WHEN OTHERS;  
sel <= 0 WHEN a = '0' AND b = '0' ELSE  
  1 WHEN a = '1' AND b = '0' ELSE  
  2 WHEN a = '0' AND b = '1' ELSE  
  3 WHEN a = '1' AND b = '1' ELSE  
  4 ;  
END mux4;
```

Figure 2-2
Mux4 Symbol.

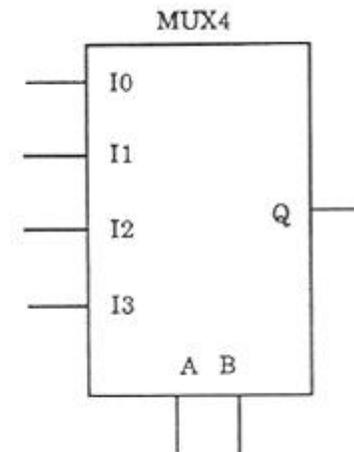


Figure 2-3
Mux Functional
Table.

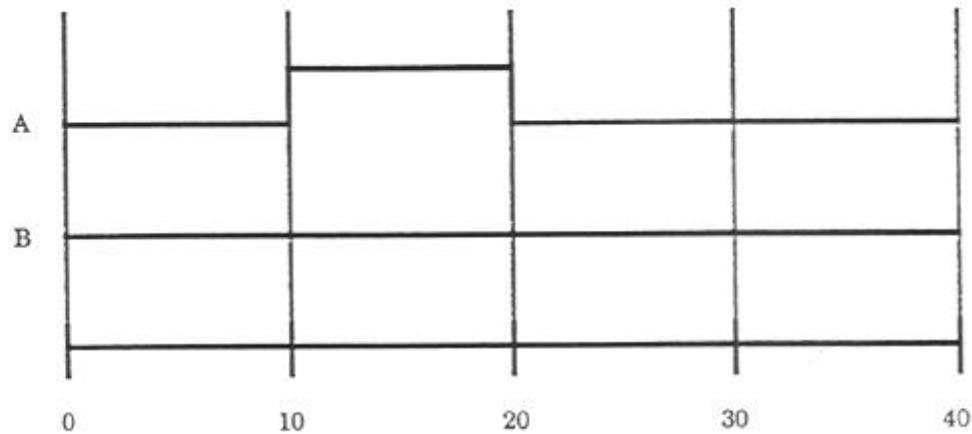
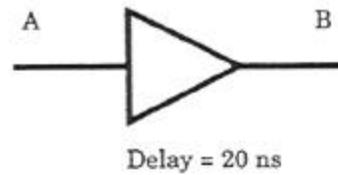
A	B	Q
0	0	I0
1	0	I1
0	1	I2
1	1	I3

Transport Versus Inertial Delay

o Inertial Delay()

.

Figure 2-4
Inertial Delay Buffer
Waveforms.



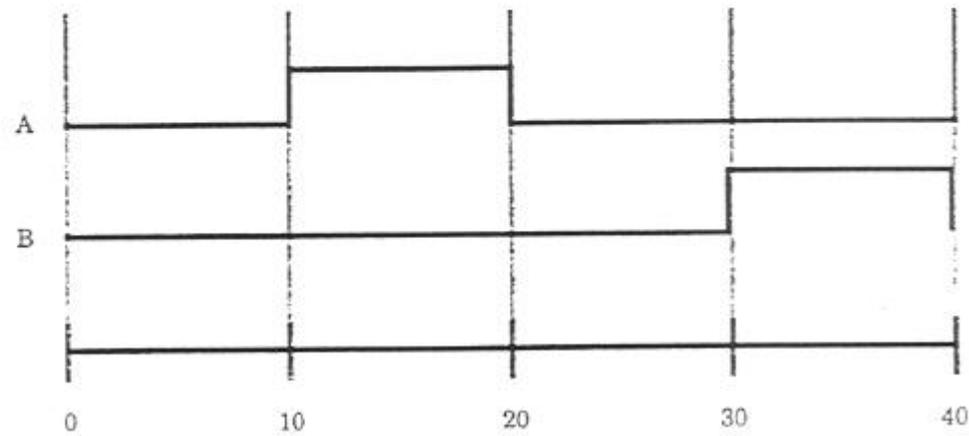
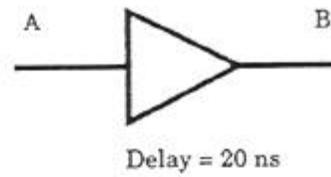
```
LIBRARY IEEE;  
USE IEEE.std_logic_1164.ALL;
```

```
ENTITY buf IS  
PORT (a : IN std_logic;  
            b : OUT std_logic);  
END buf;
```

```
ARCHITECTURE buf OF buf IS  
BEGIN  
   b <= a AFTER 20 ns;  
END buf;
```

o Transport Delay

Figure 2-5
Transport Delay
Buffer Waveforms.



```
LIBRARY IEEE;  
USE IEEE.std_logic_1164.ALL;
```

```
ENTITY delay_line IS  
PORT (a : IN std_logic;  
            b : OUT std_logic);  
END delay_line;
```

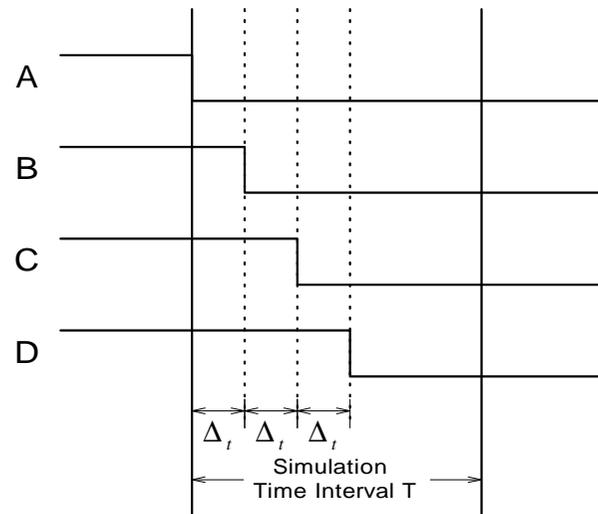
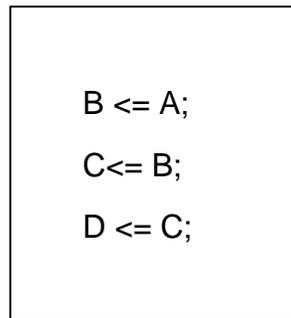
```
ARCHITECTURE delay_line OF delay_line IS  
BEGIN  
   b <= TRANSPORT a AFTER 20 ns;  
END delay_line ;
```

Simulation Deltas

Delta Delay

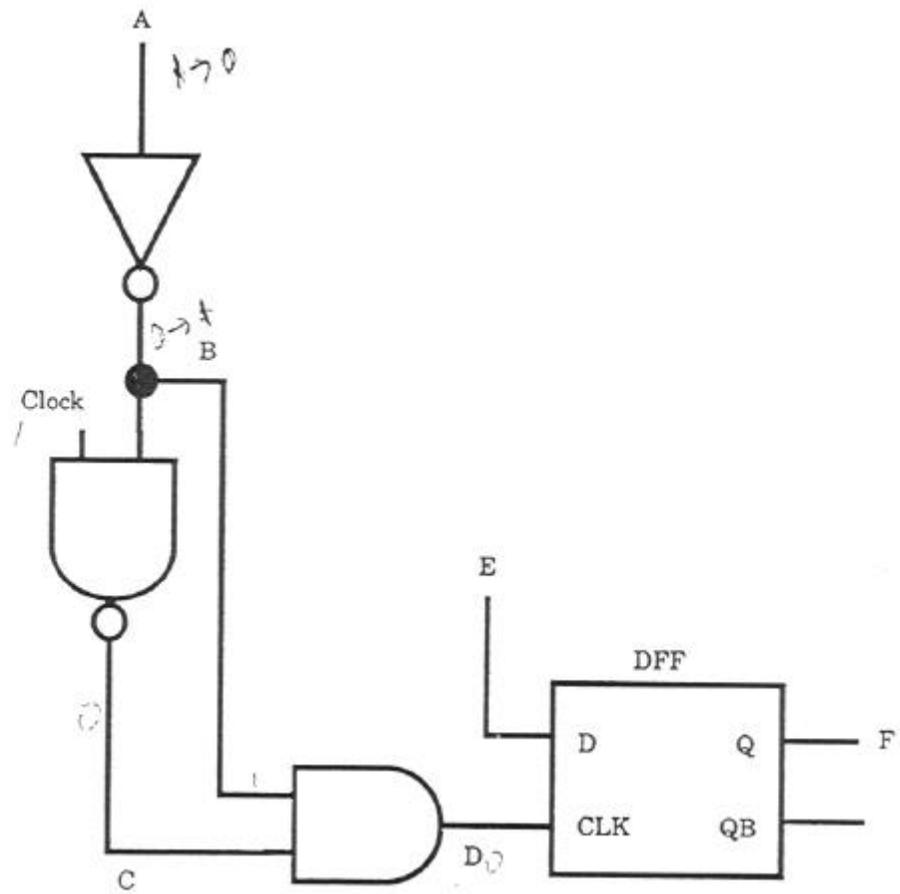
-

Event



$$T \geq n * \Delta_t$$

Figure 2-6
Simulation Delta
Circuit.



```
ENTITY reg IS  
PORT (a, clock : IN std_logic;  
      d : OUT std_logic);  
END reg;
```

```
ARCHITECTURE test OF reg IS  
SIGNAL b, c : BIT  
BEGIN  
    b <= NOT(a);  
    c <= NOT(clock AND b);  
    d <= c AND b;  
END test ;
```

Drivers

```
ARCHITECTURE test OF test IS
BEGIN
  a <= b AFTER 10ns;
  a <= c AFTER 10ns;
END test ;
```

o Bad Multiple Driver Model

```
USE WORK.std_logic_1164.ALL;
```

```
ENTITY mux IS
PORT (i0, i1, i2, i3, a, b : IN std_logic;
      q : OUT std_logic);
END mux;
```

```
ARCHITECTURE bad OF mux IS
BEGIN
  q <= i0 WHEN a = '0' AND b = "0" ELSE '0' ;
  q <= i1 WHEN a = '1' AND b = "0" ELSE '0' ;
  q <= i2 WHEN a = '0' AND b = "1" ELSE '0' ;
  q <= i3 WHEN a = '1' AND b = "1" ELSE '0' ;
END bad ;
```

ARCHITECTURE better OF mux IS

BEGIN

**q <= i0 WHEN a = '0' AND b = "0" ELSE
i1 WHEN a = '1' AND b = "0" ELSE
i2 WHEN a = '0' AND b = "1" ELSE
i3 WHEN a = '1' AND b = "1" ELSE
'X';**

END better;

Block Statement

- (Block Statement)
 -
 - (Guarded Signal)
 - . (BLOCK / .)
 - (Guarded Signal)
 - GUARD 가
Register Bus

```

blk : BLOCK (clk = '1')    -- guarded condition
BEGIN
    s1 <= GUARDED d WHEN enable=1 ELSE
        '0 ;                -- Guarded Signal Assignment
END BLOCK;

```

□ Synthesis 가 VHDL

- Target Library가
- RTL Description 가
- Structural Description 가
- Behavioral Description 가
(Tool 가)
- IP
- 가
 - 가
Multiplexer, Decoder, Encoder, Comparator, Adder/Subtractor, ALU, Multiplier, Lookup Table, Parity Generator, PLA
 - 가
Counter, Register, Latch, Parallel/Serial Converter, Sequencer, Controller, Finite State Machine, Synchronizer

□ VHDL Synthesis

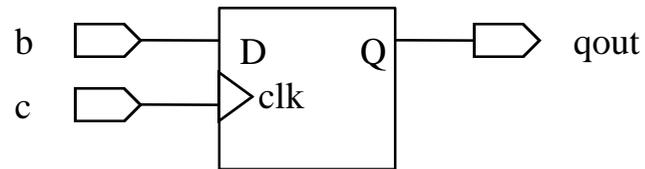
- Synthesis Path, Gate Count
VHDL 가
VHDL 가
Synthesis Critical 가
- Simulation Code
VHDL Source Code 가
Synthesis 가
RTL
- Synthesis
- VHDL Synthesis
Top-Down Simulation
Test Bench
- Tool Synthesis 가
Tool 가
- Testable Design
Toggle Rate
Fault Simulation
- Synthesis 가
- Synthesis

•VHDL Coding Synthesis

·가. D Flip Flop

```
Entity D_FF is  
  port (b, c : in bit;  
        qout : out bit;  
end D_FF;
```

```
architecture arch_D_FF of D_FF is  
begin  
  process  
  begin  
    wait until clk' event and clk = '1' ;  
    qout <= b ;  
  end process ;  
end arch_D_FF ;
```



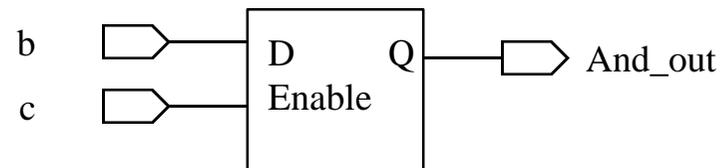
- . Latch

Entity Latch is

```
port (b, c : in bit;  
      and_out : out bit);  
end Latch;
```

architecture arch_Latch of Latch is

```
begin  
  process(b, c)  
  begin  
    if (c = '1' ) then  
      and_out <= b ;  
    end if;  
  end process;  
end arch_Latch;
```



▪ Signal Variable

- Simulation Signal Update Delta Time
Variable .
- Variable Simulation Glitch
.
- Variable Simulation
.

▪ VHDL Coding

- 'buffer'
- if Synthesis if .
- Latch가 Synthesis if, case
- Coding .
- Reset 가 if Clock
- Reset 가 if Clock