

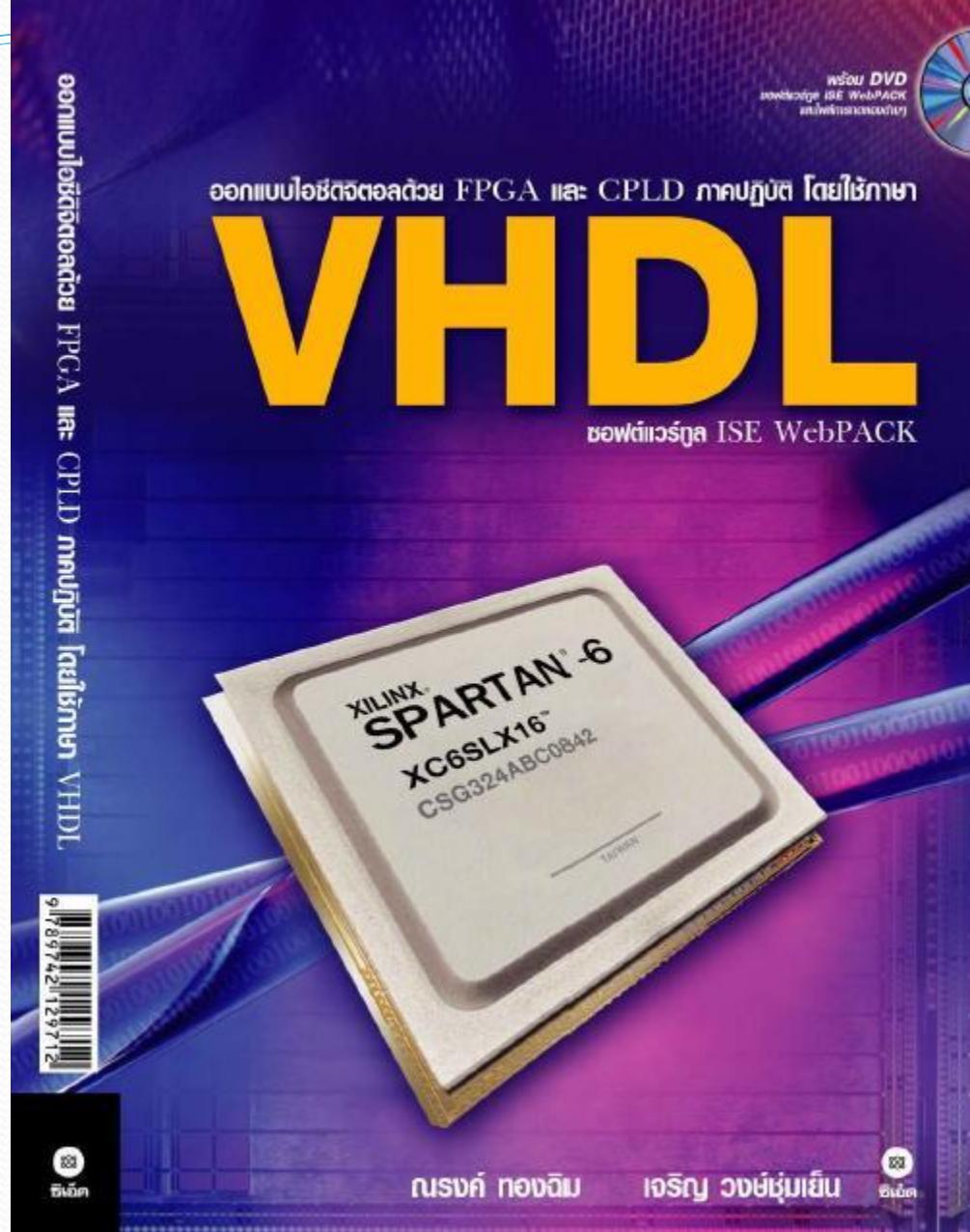
Digital System Fundamental

[Introduction to VHDL]

Charoen Vongchumyen

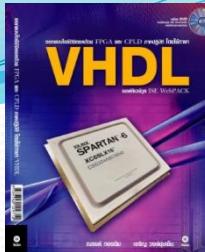
Computer engineering
King Mongkut's Institute of Technology Ladkrabang, Thailand

Nov 2018

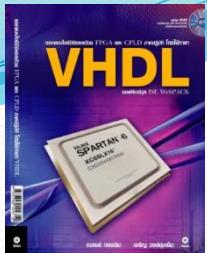




VHDL



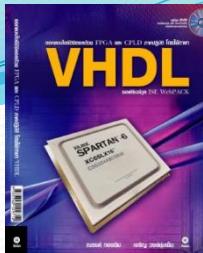
- What is VHDL?
- Why we have to learn VHDL?
- How do we learn VHDL?
- When do we learn VHDL?
- Where do we learn VHDL?



Digital design

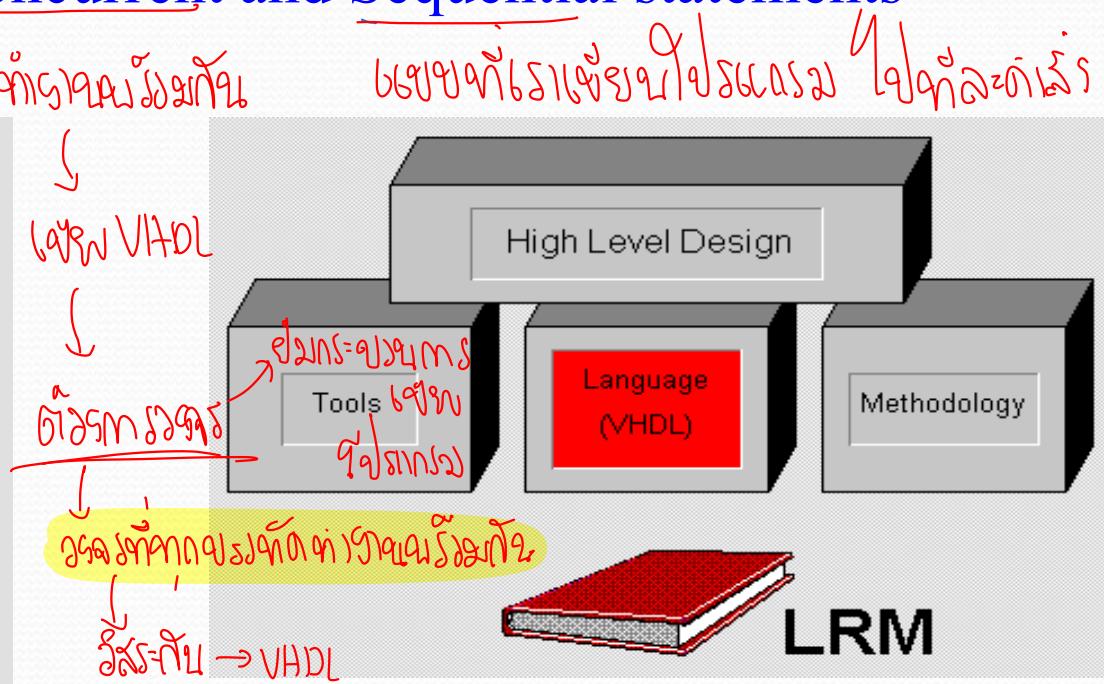
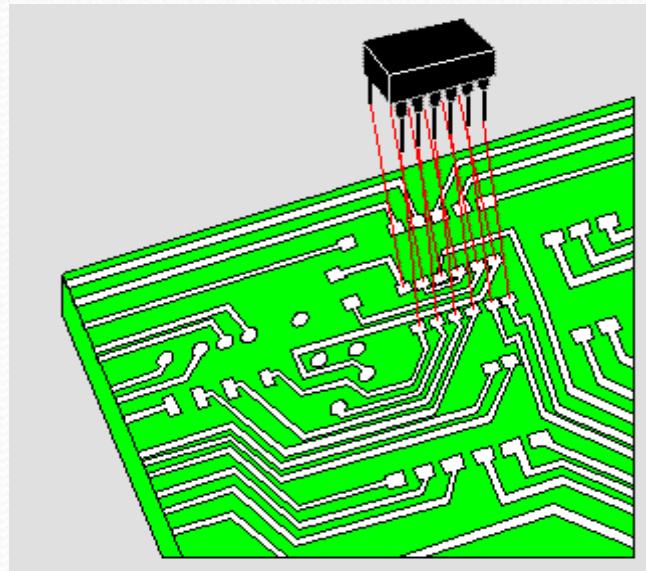
**What is Digital design
&
Why ?**

Introduction & Overview



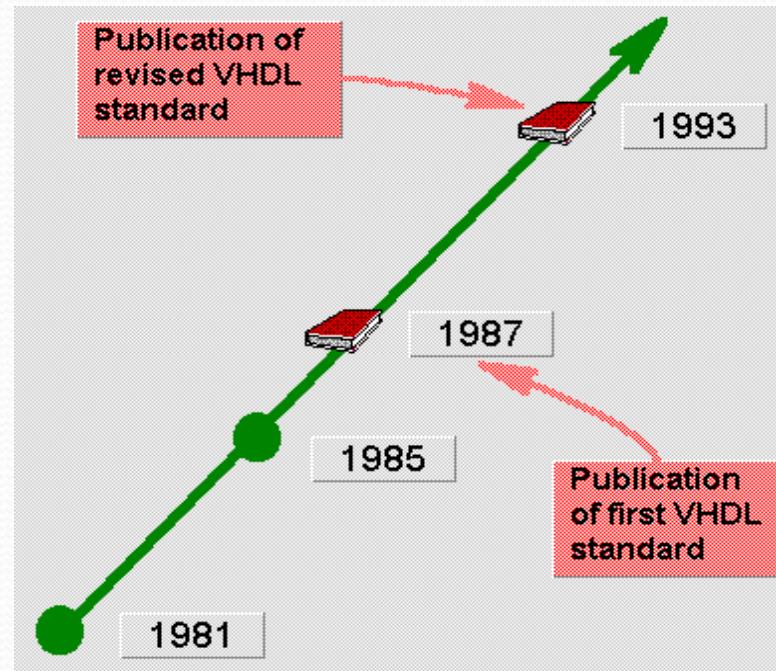
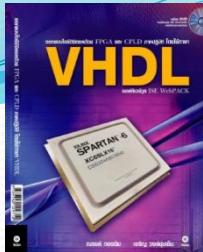
What is VHDL ?

- VHSIC (Very High Speed Integrated Circuit) Hardware Description Language
- Modeling DIGITAL Electronic Systems
- Both Concurrent and Sequential statements





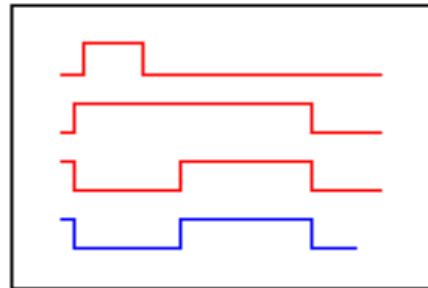
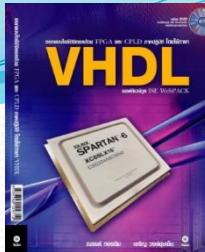
VHDL History



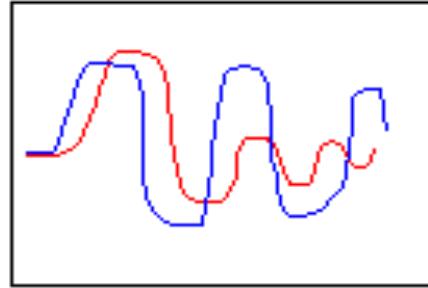
- Department of Defense (DoD) developed in 1981
- IEEE 1076-1987 Standard (VHDL'87)
- IEEE 1076-1993 Standard (VHDL'93)



VHDL Limitation



Digital



Analog

Now:

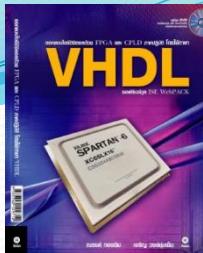


Soon:





Hardware Description Language (HDL)



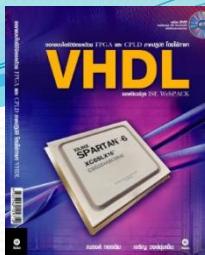
“The VHDL is a software programming language used to model the intended operation of a piece of hardware, similar to Verilog-HDL.”

Use of the VHDL Language

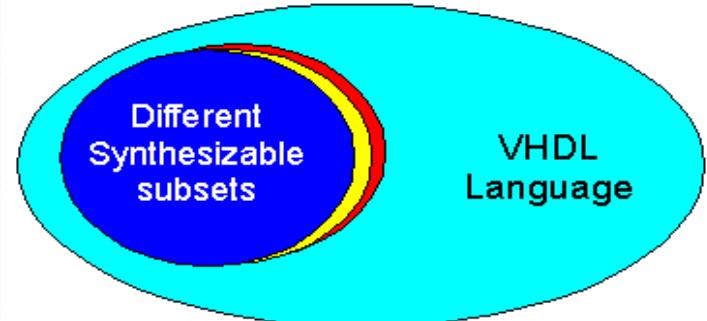
- 1. Documentation Language:** To provide human and machine readable documentation
- 2. Design Language:** To provide a structure reflecting hardware design and hierarchy and provide methods to handle complexity by partitioning the design.



Hardware Description Language (HDL)

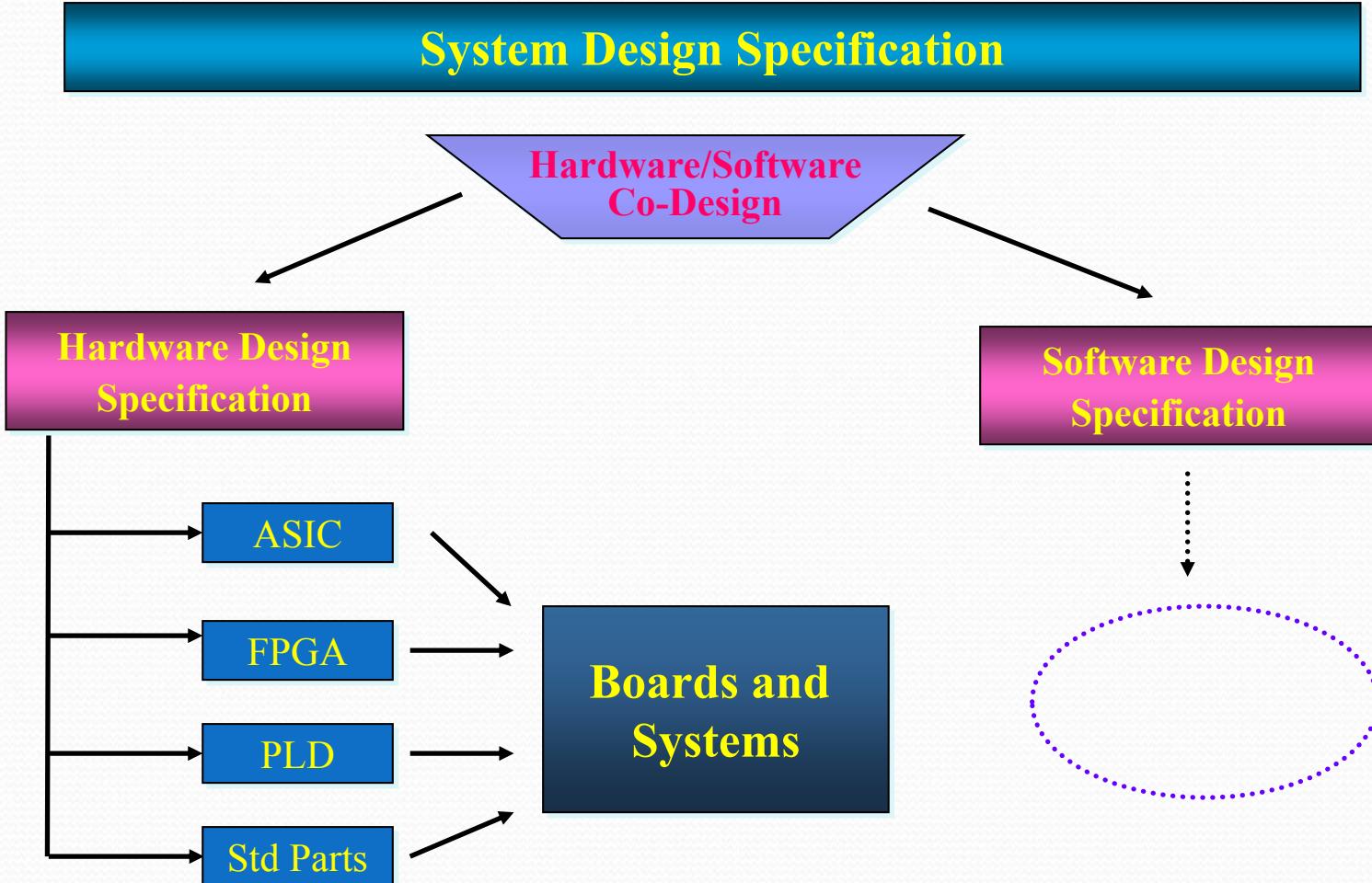
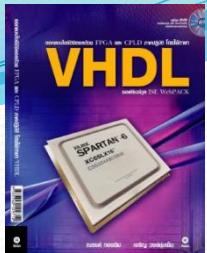


3. **Verification Language** : To provide a concurrent method verifying hardware interaction and provide constructs for stimulating a design.
4. **Test Language** : To provide ability to generate test vectors, multiple testbench strategies and method to write self checking code.
5. **Synthesis Language** : To provide high-level constructs that can be translated to Boolean equations and then translate them to gate as well as to provide constructs that can be optimized.



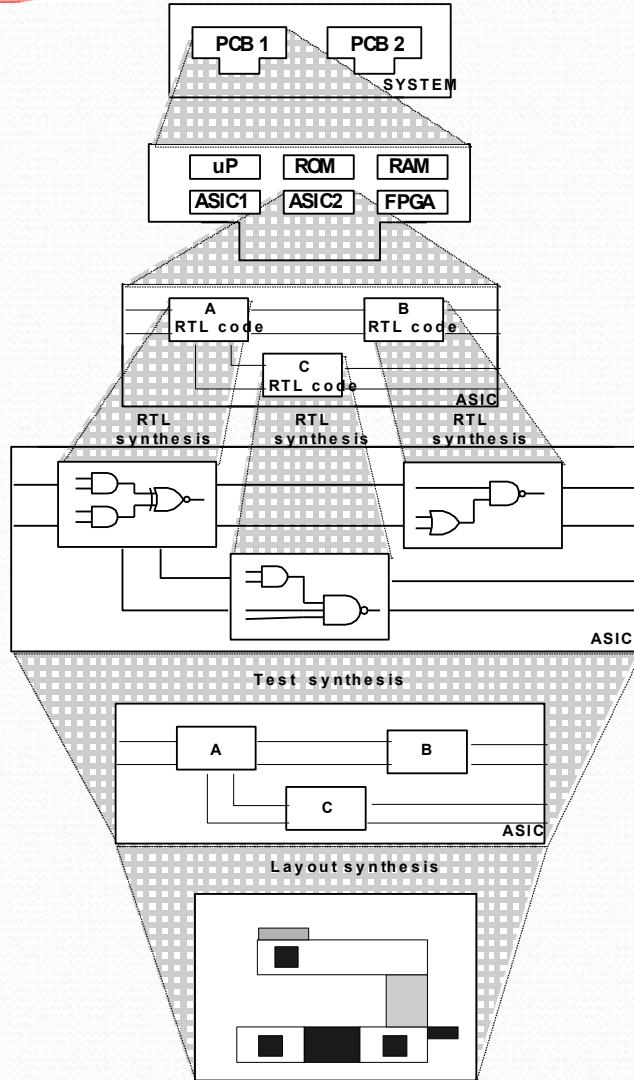
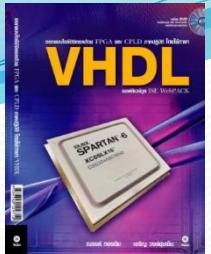


Electronic System Design





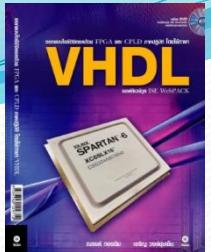
Top-Down Design



The top-level system is modeled for functionality and performance using a high-level behavioral description.

Each major component is modeled at the behavioral level and the design is simulated again for functionality and performance.

Each major component is modeled at the gate level and the design is simulated again for timing, functionality and performance.



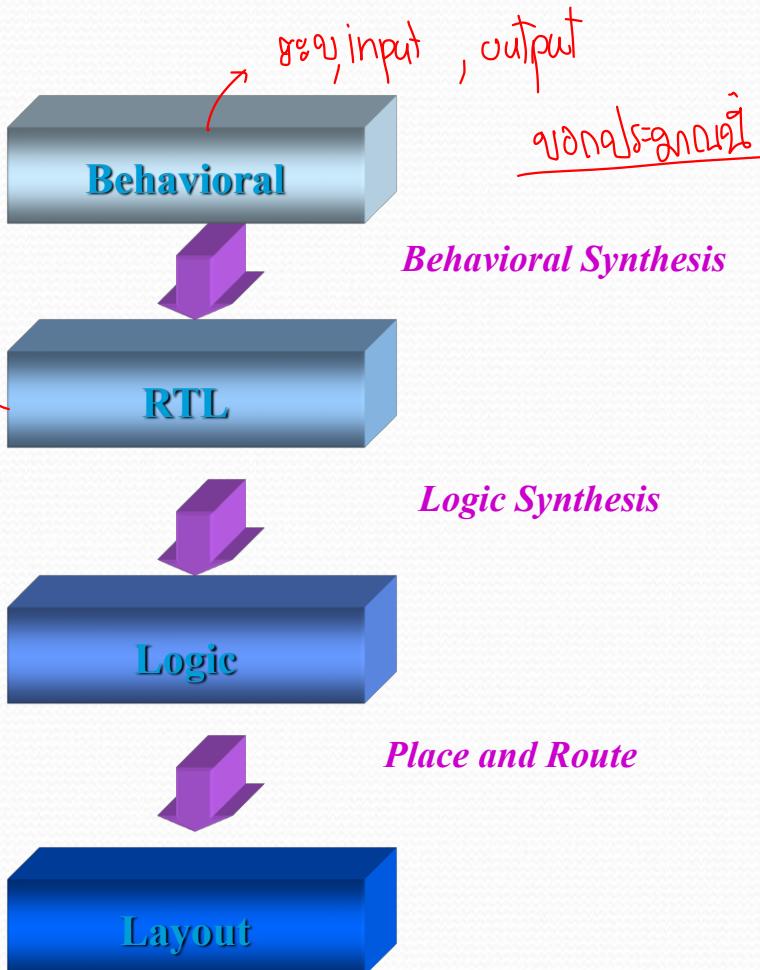
Levels of Abstraction : Capture

V
H
D
L

*Editing VHDL Code
Block Capture
System Level Tools*

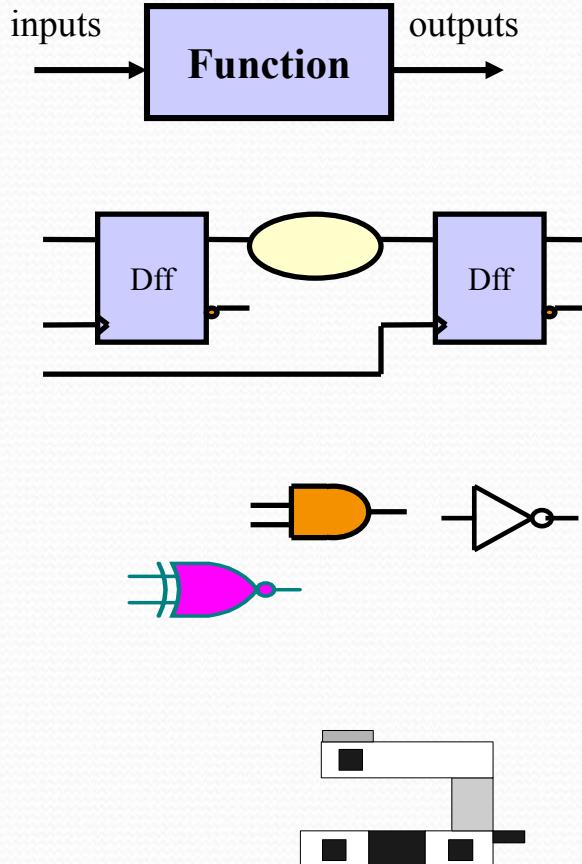
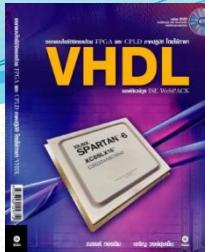
Schematic Capture

Layout Tools





Levels of Abstraction : Definition



Behavioral

*Hardware system specification
Standard parts models*

Requirement
RTL

*ASIC/FPGA design for synthesis
Synthesizable models*

Logic

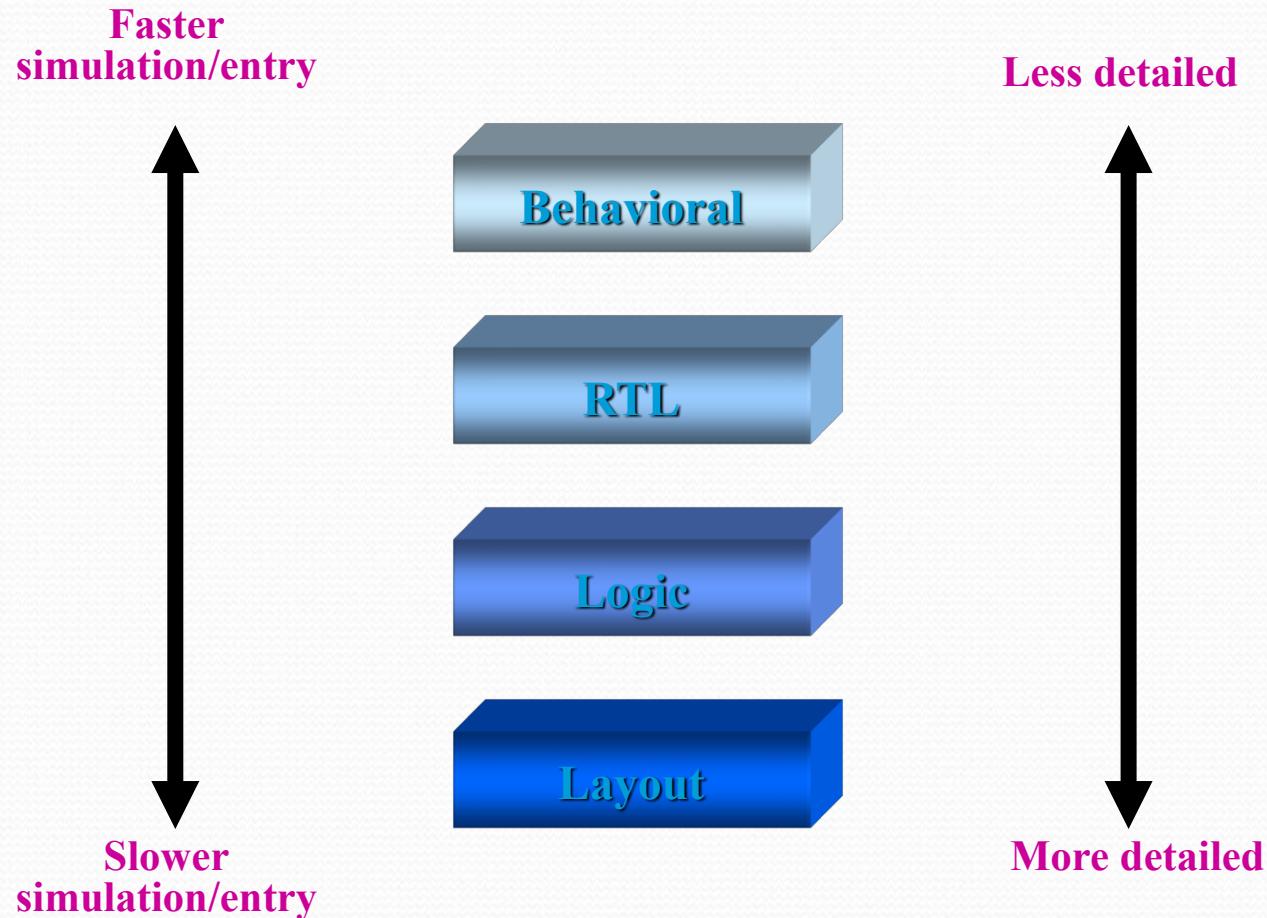
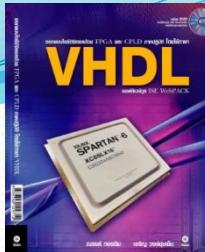
*Gate level design
PLD design*

Layout

Full custom design

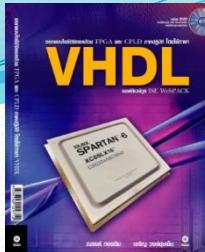


Tradeoffs Between Abstraction Levels





The Benefits of Using VHDL

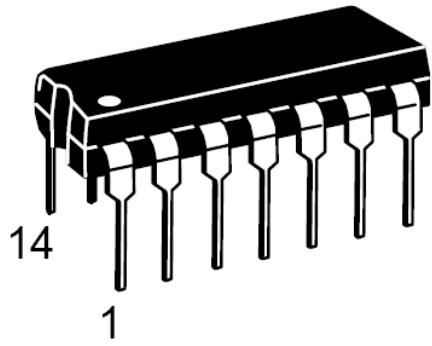
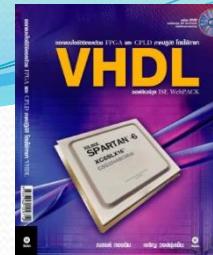


- **Design at a higher level**
 - Find problems earlier
- **Implementation independent**
 - Last minute changes
 - Delay implementation decisions
- **Flexibility**
 - Re-use
 - Choice of tools, vendors
- **Language based**
 - Faster design capture
 - Easier to manage

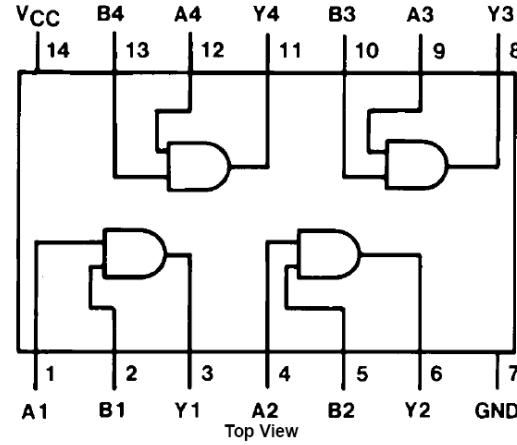
Digital design with **FPGA & CPLD**



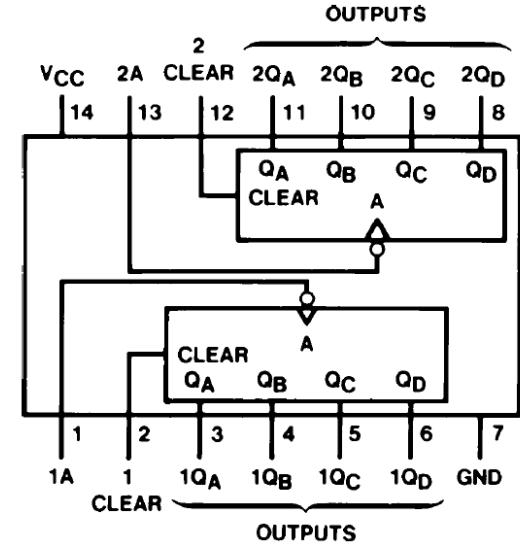
Standard IC



DIP Package



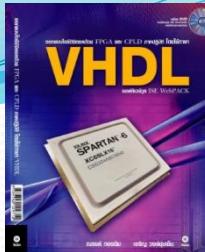
4 Nand gate



Dual binary counter



Evolution on digital design



TTL = Transistor Transistor Logic

CMOS = Complementary Metal Oxide Silicon

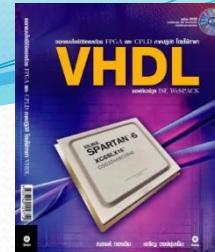
PLD = Programmable Logic Device

CPLD = Complex Programmable Logic Device

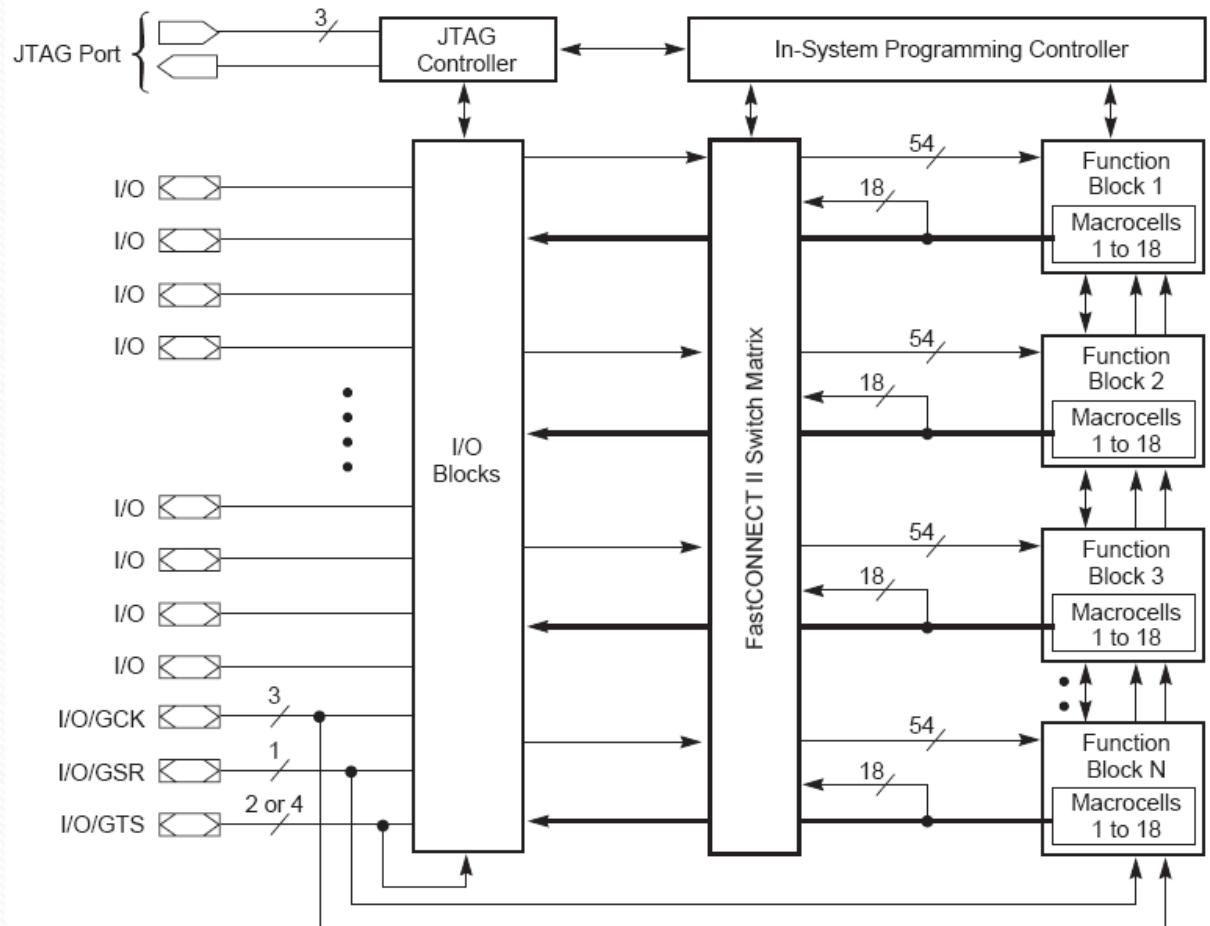
FPGA = Field Programmable Gate Array



CPLD

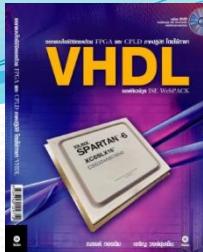


Xilinx
XC9536XL
800 Gates

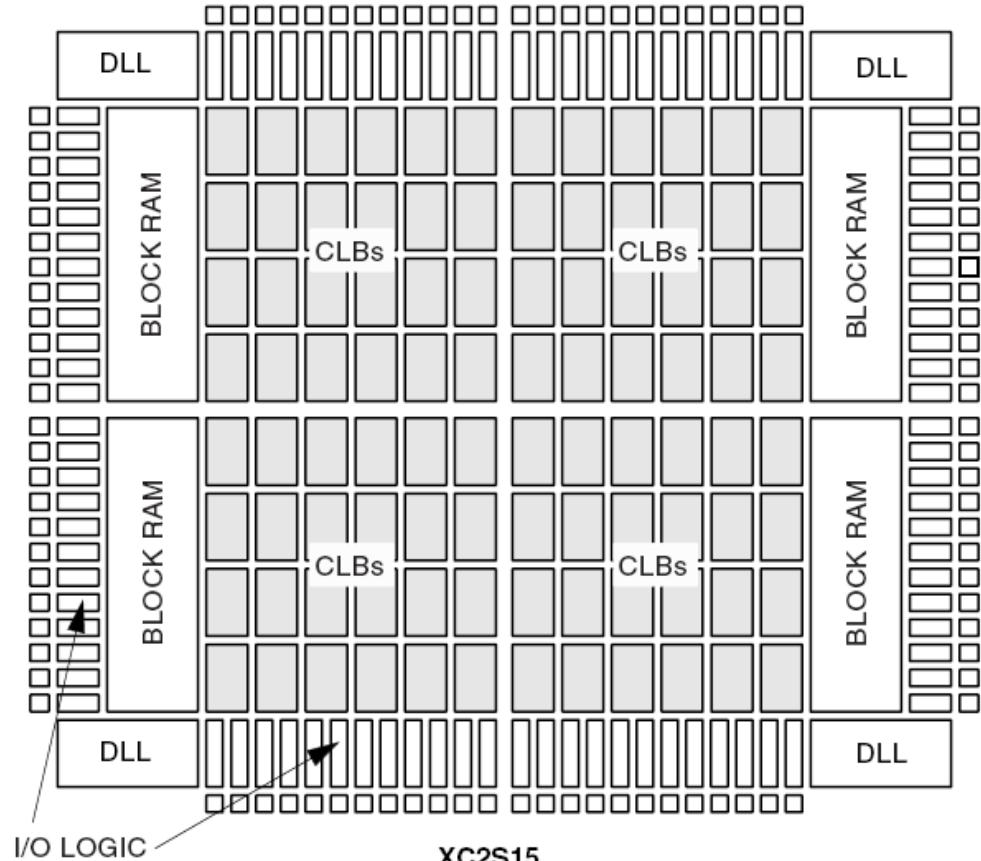




FPGA



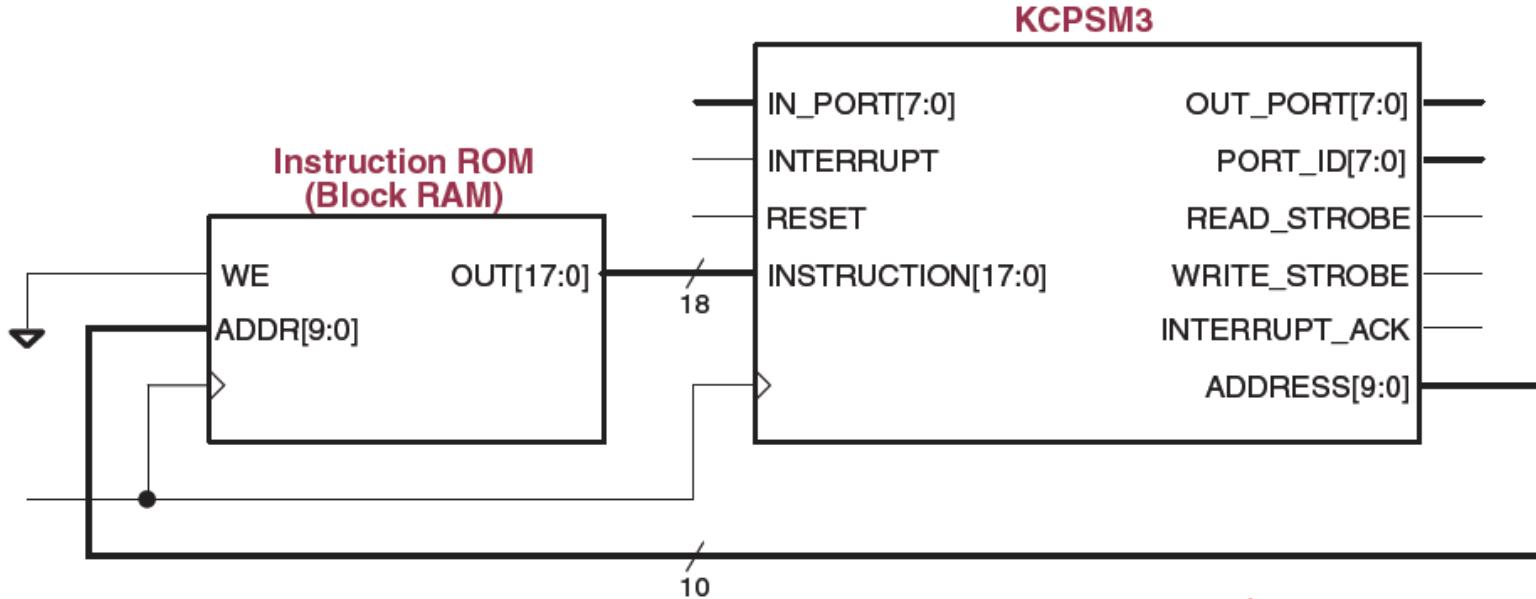
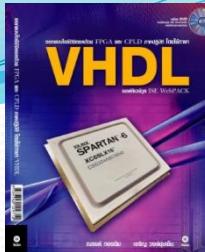
Xilinx
Spartan-II XC2S15
15,00 Gates



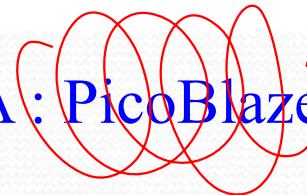
- Delay-Locked Loop (DLL)
- Configurable Logic Block (CLB)



FPGA, CPLD & MCU



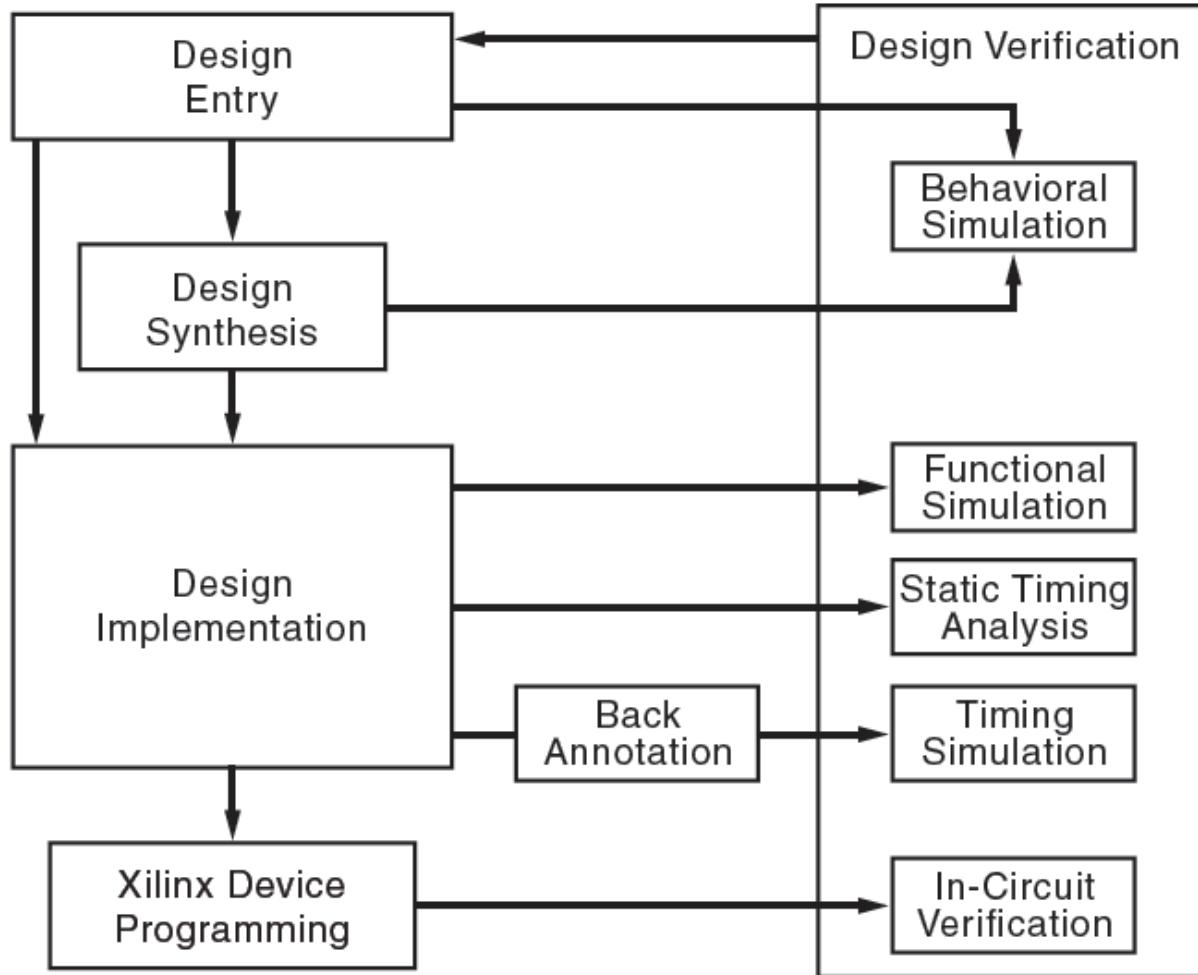
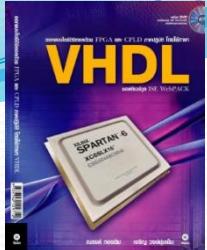
Embedded MCU in FPGA : PicoBlaze

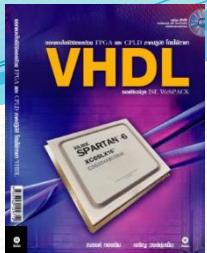


System On a Chip(SOC)



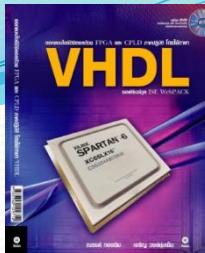
FPGA & CPLD Design flow





Design process

- Design entry
 - Schematic
 - HDL : VHDL, Verilog
 - State diagram
- Design synthesis
 - Netlist files
 - XST
 - EDIF

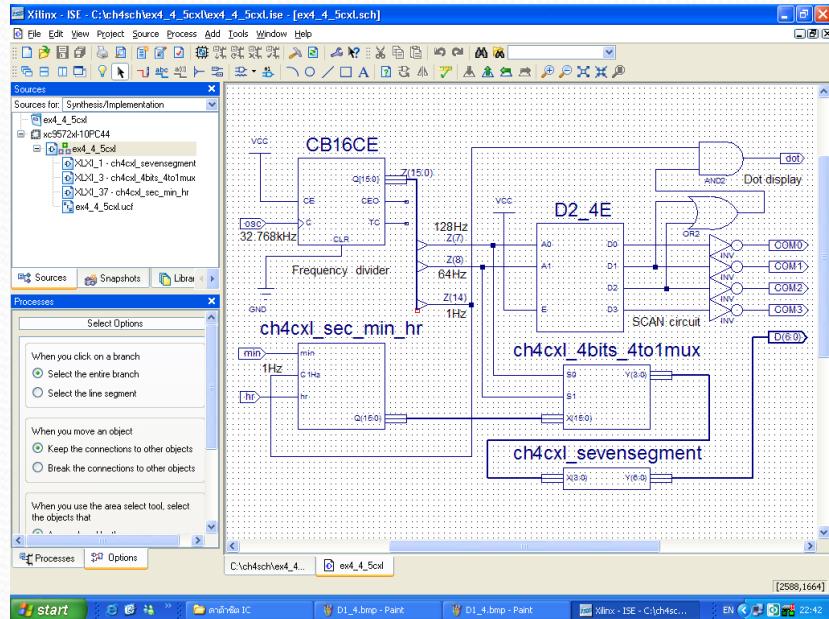
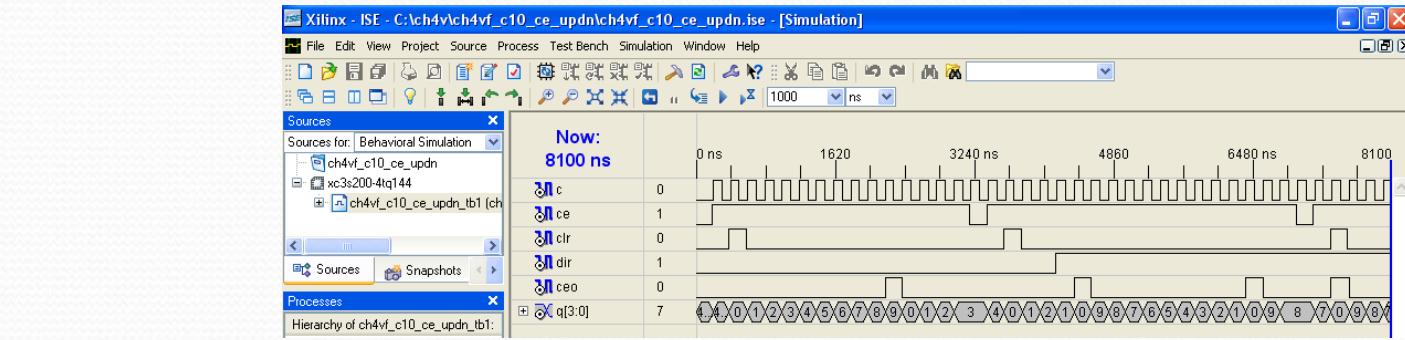
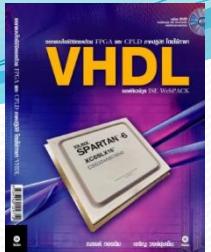


Design process

- Design testing & Verification
 - Behavioral simulation
 - Functional simulation
 - Timing simulation
- Design implementation
 - Translate (Netlist + Constrains)
 - Mapping
 - Place & Route
 - Programming file generation
 - Fitting (CPLD)
- Programming (.bit, .jed)



ISE WebPACK™



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD.TEXT_UNSIGNED.ALL;

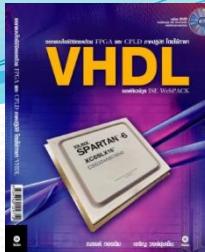
entity C100UP is
Port ( OSC : in STD_LOGIC;
PBI : in STD_LOGIC;
CLR : in STD_LOGIC;
DOT : out STD_LOGIC;
T : out STD_LOGIC_VECTOR (6 downto 0);
COM : out STD_LOGIC_VECTOR (1 downto 0));
end C100UP;

architecture Behavioral of C100UP is
signal C,S,DB,CLR_DB : STD_LOGIC;
signal Q_temp : STD_LOGIC_VECTOR (14 downto 0);
signal Q,Q1,A : STD_LOGIC_VECTOR (3 downto 0);
begin
DB : process(OSC,PBI,CLR_DB)
begin
if CLR_DB='0' then
C <= '0';
elsif C_DB'event and C_DB='1' then
if PBI='0' then
C <= '1';
end if;
end if;
end process;
C_DB <= not C and OSC;
F_DIV : process(OSC)
begin
if OSC'event and OSC='1' then

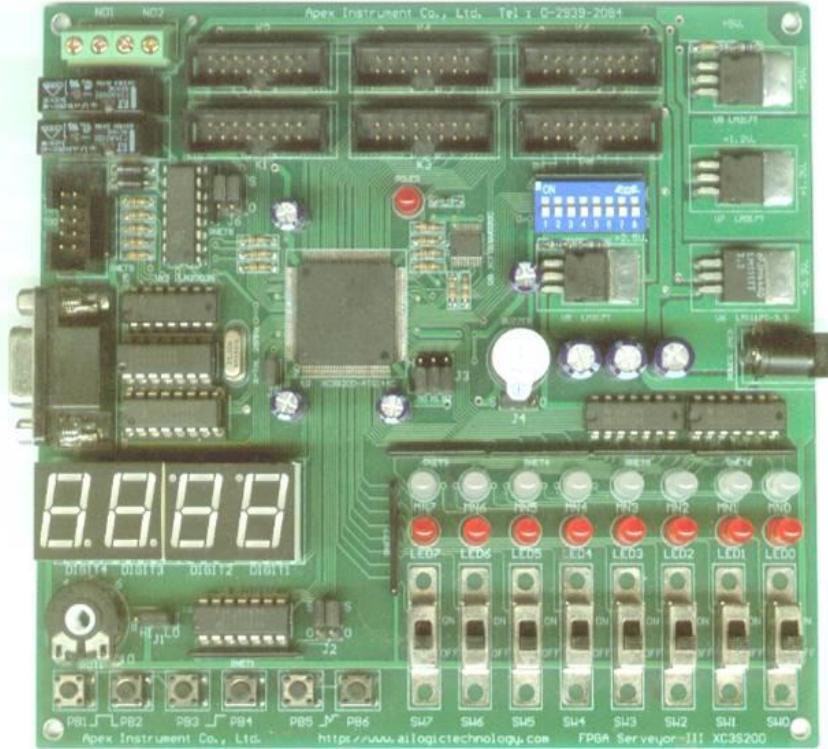
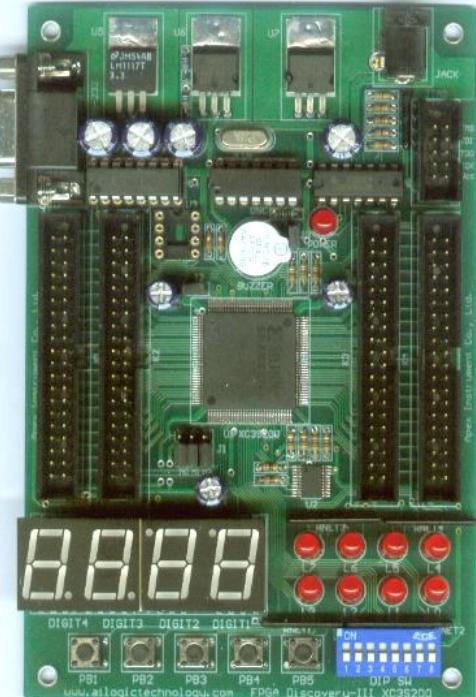
```



Development board



- Performance
- Reliability
- Functionality
- Reasonable price



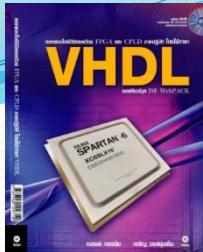
www.ailogictechnology.com

VHDL

Basic & introduction



VHDL Structural elements



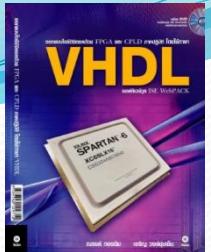
- **Entity declaration** → մտքայի մուլտիպլիքացիոն բլոգում
 - Interface definition
 - Input and output signal (Ports) → input , output
- **Architecture**
 - Architecture body input & output → what is relation between
 - Relation of input and output
 - Circuit behavior
- **Package**
 - Library : Constants, Procedures, Data types, Components

Դեռ այսուհետու կարող է անդամակից աշխատանքը առաջանալ
- **Configuration**
 - Match entity declaration and architecture

1 entity 2 declarative architecture behavioral config los
VHDL : Tong +



VHDL Basic : Example



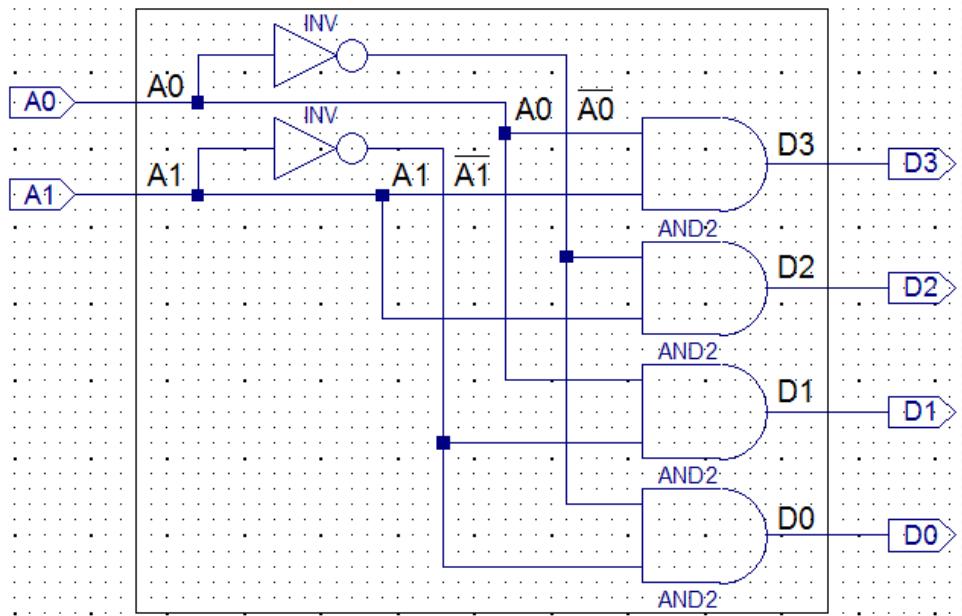
Input		Output			
A1	A0	D3	D2	D1	D0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$$D0 = \overline{A1} \cdot \overline{A0}$$

$$D1 = \overline{A1} \cdot A0$$

$$D2 = A1 \cdot \overline{A0}$$

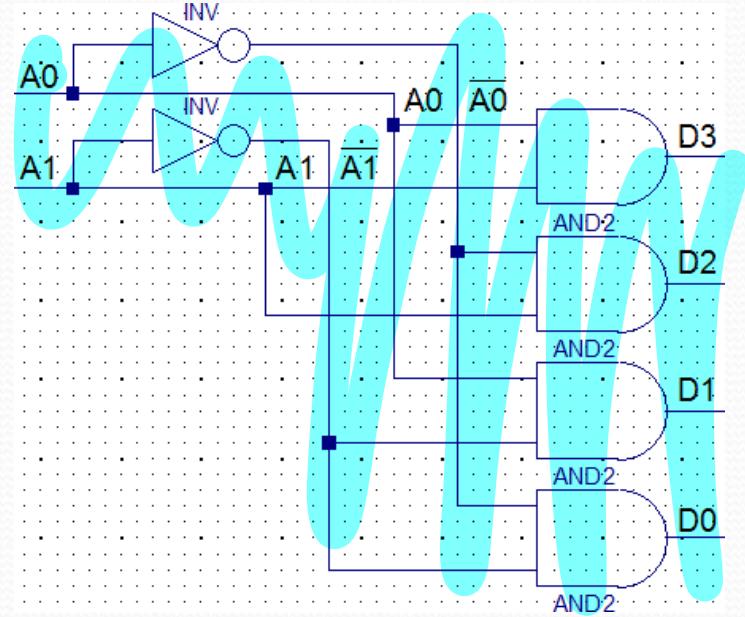
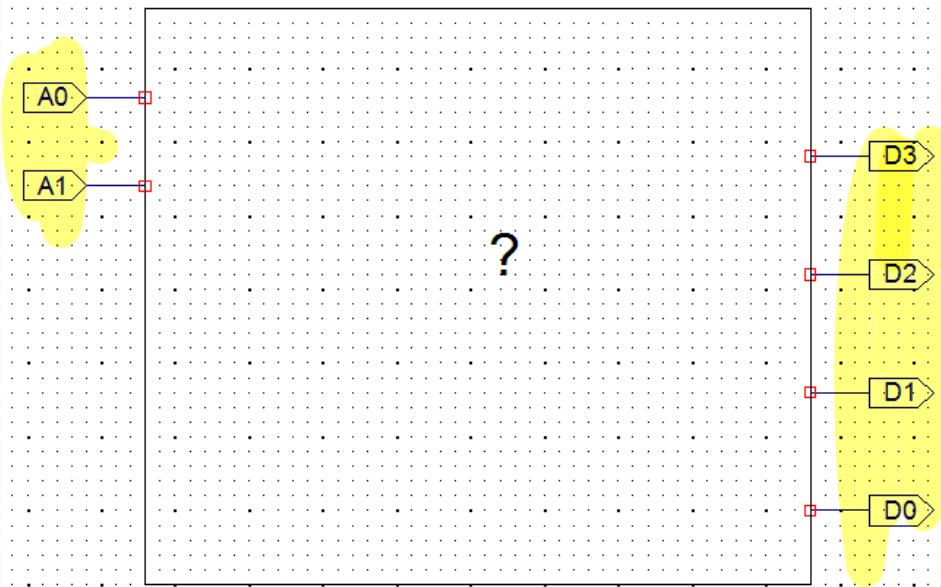
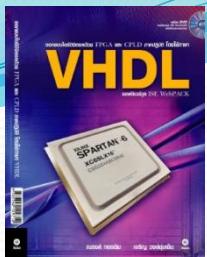
$$D3 = A1 \cdot A0$$



Example of 2 to 4 decoder



VHDL Basic : Example



Entity declaration

Architecture



VHDL Basic : Example



```
2  entity DECODER2TO4 is
3    port ( A0 : in bit;
4           A1 : in bit;
5           D0 : out bit;
6           D1 : out bit;
7           D2 : out bit;
8           D3 : out bit);
9  end DECODER2TO4;
10
11 architecture BEHAVIORAL of DECODER2TO4 is
12 begin
13   D0 <= (not A1) and (not A0);
14   D1 <= (not A1) and A0;
15   D2 <= A1 and (not A0);
16   D3 <= A1 and A0;
17 end BEHAVIORAL;
```

ชื่อエンティตี้

Mode

ชนิดข้อมูล bit

จบค้วย ";"

บรรทัดที่ 2-9 เป็นประกาศใช้อเอนติตี้ ซึ่งส่วนนี้จะบอกว่าจะรอมินพุตและเอาต์พุตอะไรบ้าง

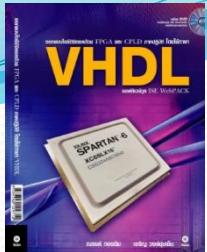
ชื่ออาชิเทกเจอร์

ชื่อエンติตี้

บรรทัดที่ 11-17 เป็นอาชิเทกเจอร์บอดี้ ซึ่งส่วนนี้จะบอกว่าจะทำงานอย่างไร

Operator เช่น and และ not

VHDL 87 code

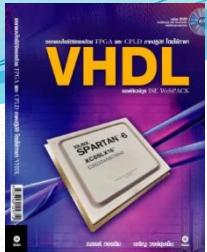


Signal Concurrency

VHDL → Compile → gos

```
11 architecture BEHAVIORAL of DECODER2TO4 is
12 begin
13     D3 <= A1 and AO;
14     D2 <= A1 and(not AO);
15     D1 <= (not A1)and AO;
16     D0 <= (not A1)and(not AO);
17 end architecture BEHAVIORAL;
```

Concurrent statement



Compact style



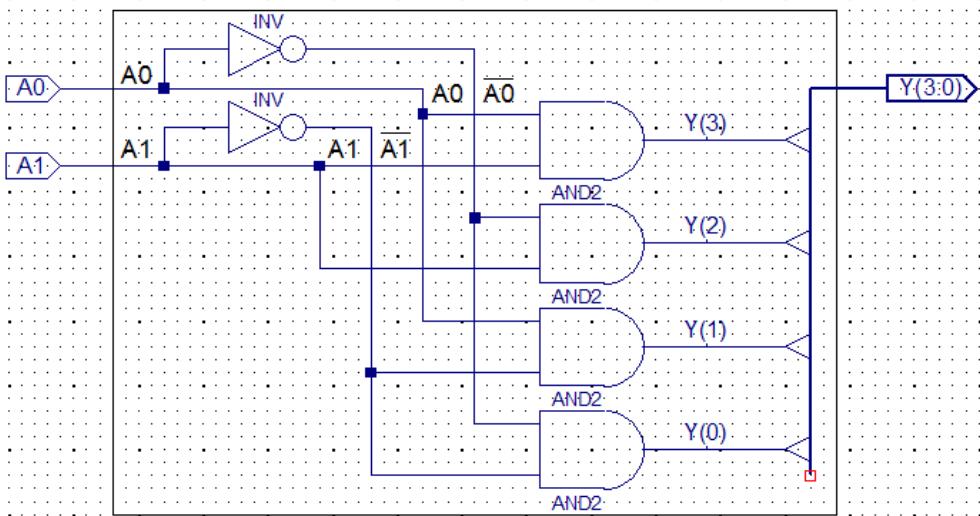
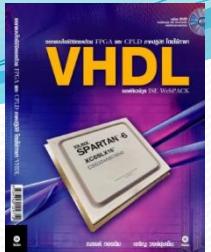
```
2 entity DECODER2TO4 is
3     port ( A0,A1 : in bit;
4             D0,D1,D2,D3 : out bit);
5 end DECODER2TO4;
6
7 architecture BEHAVIORAL of DECODER2TO4 is
8 begin
9     D0 <= (not A1)and(not A0);
10    D1 <= (not A1)and A0;
11    D2 <= A1 and(not A0);
12    D3 <= A1 and A0;
13 end BEHAVIORAL;
```

การเขียนอินพุตหรือเอาต์พุตหลายตัวจะต้อง
มี “,” กันระหว่างอินพุตหรือเอาต์พุตทุกตัว

} สืบฯ มาก็จะใช้ “,” ไม่ใช่ “ ”



Bus style

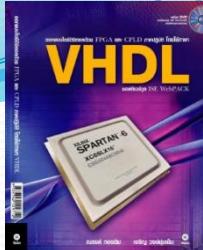


```
2 entity DECODER2TO4 is
3     port ( A0,A1 : in bit;
4             Y : out bit_vector(3 downto 0));
5 end DECODER2TO4;
6
7 architecture BEHAVIORAL of DECODER2TO4 is
8 begin
9     Y(0) <= (not A1)and(not A0);
10    Y(1) <= (not A1)and A0;
11    Y(2) <= A1 and(not A0);
12    Y(3) <= A1 and A0;
13 end BEHAVIORAL;
```

MSB LSb



General Coding in VHDL



```
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4
5 entity DECODER2TO4 is
6     port ( A0,A1 : in STD_LOGIC;
7             Y : out STD_LOGIC_VECTOR(3 downto 0));
8 end DECODER2TO4;
9
10 architecture BEHAVIORAL of DECODER2TO4 is
11 begin
12     Y(0) <= (not A1)and(not A0);
13     Y(1) <= (not A1)and A0;
14     Y(2) <= A1 and(not A0);
15     Y(3) <= A1 and A0;
16 end BEHAVIORAL;
```

Using the package

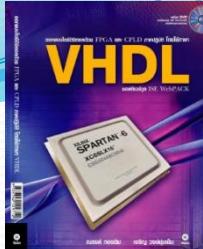
Entity declaration

Architecture body

2 to 4 Decoder

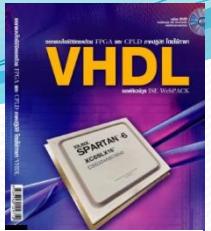


MUX 4 : 1



Control		Output O	Remark
S1	S0		
0	0	A	O = A
0	1	B	O = B
1	0	C	O = C
1	1	D	O = D

Truth table



MUX 4 : 1

```
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4
5 entity MUX4TO1 is
6     port ( A,B,C,D,S1,SO : in STD_LOGIC;
7             O : out STD_LOGIC);
8 end MUX4TO1;
9
10 architecture BEHAVIORAL of MUX4TO1 is
11     signal SEL : STD_LOGIC_VECTOR(1 downto 0);
12 begin
13     SEL <= S1&SO;           -- Concatenation
14     with SEL select
15         O <= A when "00", 00 (ເລືອດ A ດອກ
16                         B when "01",
17                         C when "10",
18                         D when others; -- SEL="11"
19 end BEHAVIORAL;
```

ຕ້ອງປະກາດໃຊ້ signal ເນື່ອຈາກ
SEL ໄນມີຊື່ໃນ port ຂອງ entity

ການຮັມໜົດຂໍ້ມູນເປັນແບບກາລຍ
ນິຕ ໂດຍໃຊ້ "&" (Concatenation)

Selected signal assignment, MUX 4 : 1



1 Bit full adder



Input			Output	
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

Truth table



1 Bit full adder



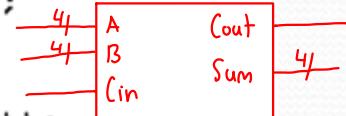
```
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4
5 entity FULL_ADDER_1BIT is
6     Port ( A,B,Cin : in STD_LOGIC;
7             Cout,Sum : out STD_LOGIC);
8 end FULL_ADDER_1BIT;
9
10 architecture Behavioral of FULL_ADDER_1BIT is
11     signal X : STD_LOGIC_VECTOR(2 downto 0);
12 begin
13     X <= A&B&Cin;
14     Sum <= '1' when (X="001" or X="010" or X="100" or X="111") else
15             '0';
16     Cout <= '1' when (X="011" or X="101" or X="110" or X="111") else
17             '0';
18 end Behavioral;
```



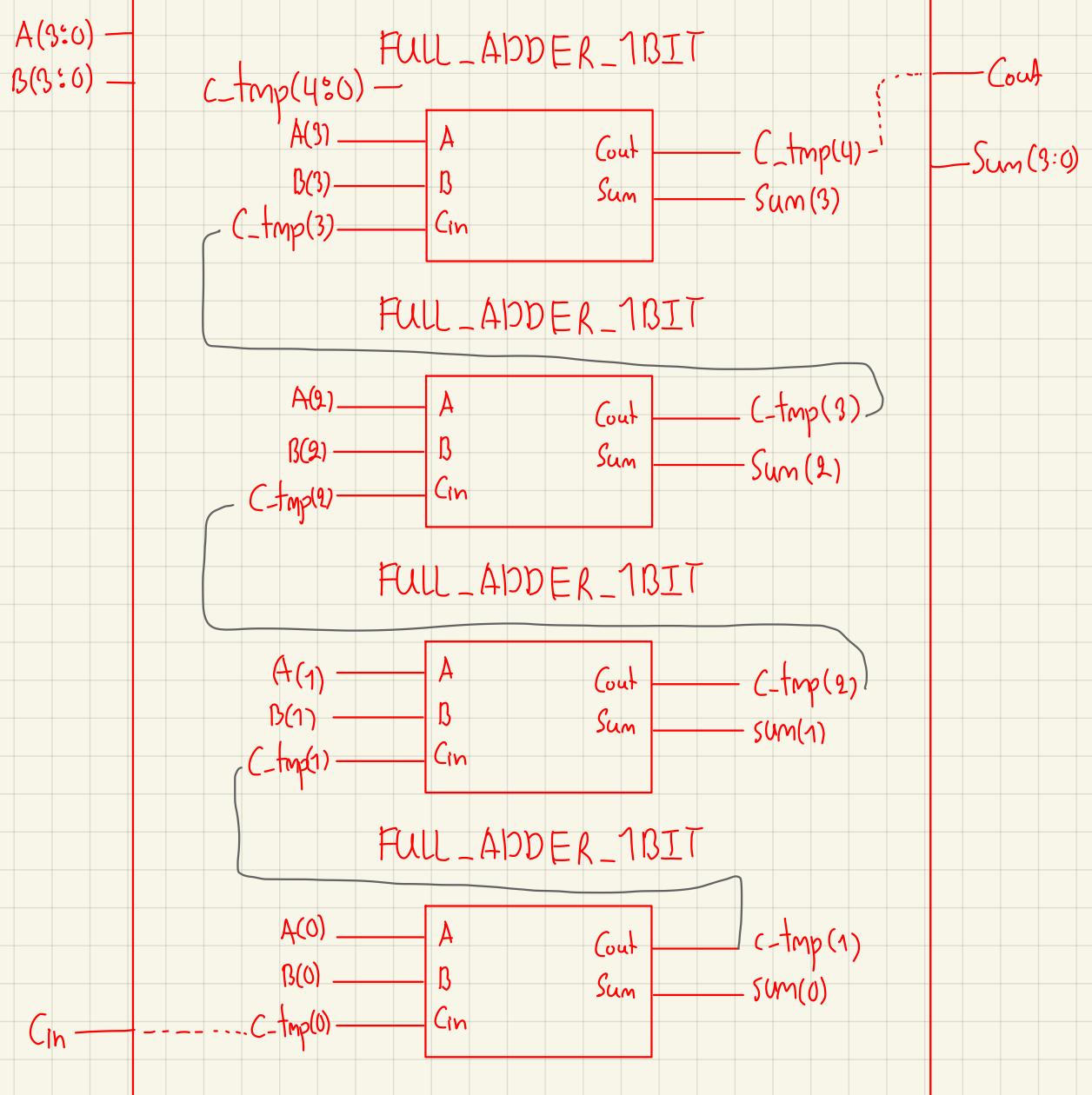
4 Bits adder using Generate



```
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4
5 entity ADDN_FOR_GEN is
6     Port ( A,B : in STD_LOGIC_VECTOR (3 downto 0);
7             Cin : in STD_LOGIC;
8             Cout : out STD_LOGIC;
9             Sum : out STD_LOGIC_VECTOR (3 downto 0));
10 end ADDN_FOR_GEN;
11
12 architecture Behavioral of ADDN_FOR_GEN is
13     component FULL_ADDER_1BIT -- ตัวเกณฑ์ส่วนๆ แล้ว
14         Port ( A,B,Cin : in STD_LOGIC; -- ห้ามซ้ำชื่อ port ให้
15                 Cout,Sum : out STD_LOGIC);
16     end component;
17     signal C_tmp : STD_LOGIC_VECTOR (4 downto 0);
18 begin
19     C_tmp(0) <= Cin; generate label ชี้ค่าวร์ตัชช์อิฟี่ด้วย
20
21 ADDN_GEN : for I in 3 downto 0 generate
22     ADDN_BIT : FULL_ADDER_1BIT -- อนิเมชัน map ณ
23         port map(A(I),B(I),C_tmp(I),C_tmp(I+1),Sum(I));
24     end generate;
25
26     Cout <= C_tmp(4); ↴ จึงต้อง pin ที่ 4 ถึง pin ที่ 0 จึงได้
27
28
29 end Behavioral;
```

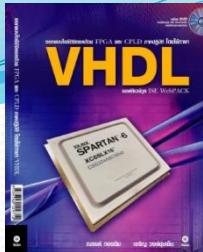


อกการ์ตูน
architecture





N Bits adder using Generate

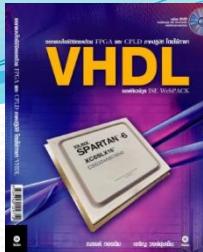


```
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4
5 entity ADDN_FOR_GEN is
6     generic (N : integer := 4);
7     Port ( A,B : in STD_LOGIC_VECTOR (N-1 downto 0);
8             Cin : in STD_LOGIC;
9             Cout : out STD_LOGIC;
10            Sum : out STD_LOGIC_VECTOR (N-1 downto 0));
11 end ADDN_FOR_GEN;
12
13 architecture Behavioral of ADDN_FOR_GEN is
14     component FULL_ADDER_1BIT
15         Port ( A,B,Cin : in STD_LOGIC;
16                 Cout,Sum : out STD_LOGIC);
17     end component;
18     signal C_tmp : STD_LOGIC_VECTOR (N downto 0);
19 begin
20     C_tmp(0) <= Cin;
21
22     ADDN_GEN : for I in A'range generate --A'range = N-1 downto 0
23         ADDN_BIT : FULL_ADDER_1BIT
24             port map(A(I),B(I),C_tmp(I),C_tmp(I+1),Sum(I));
25         end generate;
26
27         Cout <= C_tmp(C_tmp'high);           --C_tmp'high = N
28 end Behavioral;
```

for loop A



D Flip-Flop with Asynchronous Clear



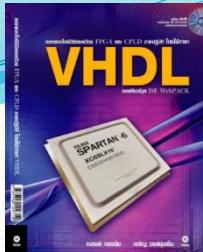
```
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4
5 entity DFF_CLR is
6     port ( D,C,CLR : in STD_LOGIC;
7             Q : out STD_LOGIC);
8 end DFF_CLR;
9
10 architecture BEHAVIORAL of DFF_CLR is
11 begin
12 process(C,CLR)
13 begin
14     if CLR='1' then Q <= '0';
15     elsif (C'event and C='1') then Q <= D;
16     end if;
17 end process;
18 end BEHAVIORAL;
```

การเปลี่ยนแปลงของอินพุต C และ CLR นั้นจะมีผลต่อเอาต์พุต Q โดยตรง (ทันที) จึงต้องเพิ่มน C และ CLR ไว้ใน Sensitivity list

C'event and C = '1' คือ C ทริกค์วัยของขาเข้าหรือขอบบวก ซึ่งจะไส่วนเล็บหรือไม่ไส่ก็ได้



4 bits Countup Binary Counter



```
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6 entity COUNTER4BIT is
7     port ( C,CLR : in STD_LOGIC;
8             Q : out STD_LOGIC_VECTOR (3 downto 0));
9 end COUNTER4BIT;
10
11 architecture Behavioral of COUNTER4BIT is
12     signal Q_temp : STD_LOGIC_VECTOR (3 downto 0);
13 begin
14     process(C,CLR)
15     begin
16         if CLR='1' then Q_temp <= "0000";
17         elsif C'event and C='1' then Q_temp <= Q_temp + 1;
18         end if;
19     end process;
20     Q <= Q_temp;
21 end Behavioral;
```

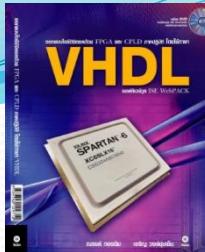
เรียกใช้ Package ชื่อ std_logic_unsigned เพื่อให้ชนิดข้อมูล std_logic_vector สามารถใช้กับ “+” ได้

signal Q_temp ไม่ใช่ Port จึงไม่มีความจำเป็นต้องใช้ Mode “buffer”

กำหนดค่า (Assign) Q_temp ซึ่งเป็น Signal ให้กับ Q นั้นต้องทำภายนอก Process เนื่องจาก การ Update ค่าของ Signal จะกระทำเมื่อจบ Process



Decade Countup Binary Counter



```
2 library IEEE;
3 use IEEE.STD_LOGIC_1164.ALL;
4 use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6 entity COUNTER10UP is
7     Port ( C,CLR : in STD_LOGIC;
8             Q : out STD_LOGIC_VECTOR(3 downto 0));
9 end COUNTER10UP;
10
11 architecture Behavioral of COUNTER10UP is
12     signal QT : STD_LOGIC_VECTOR (3 downto 0);
13 begin
14     process(C,CLR)
15     begin
16         if CLR='1' then QT <= "0000";
17         elsif C'event and C='1' then
18             if QT >= 9 then QT <= "0000";
19             else QT <= QT + 1;
20             end if;
21         end if;
22     end process;
23     Q <= QT;
24 end Behavioral;
```

เรียกใช้ Package ชื่อ std_logic_unsigned เพื่อให้ชนิด
ข้อมูล std_logic_vector สามารถใช้กับ “+” ได้

เมื่อกำ QT ≥ 9 (つまり $Q \geq "1001"$)
แล้วมีสัญญาณนาฬิกา C ทริกค์วัยขอบ
ขาขึ้นจะทำให้ QT มีค่าเป็น “0000”

VHDL

END, Introduction