

A designer's guide to VHDL synthesis

ETRI ASIC design center

The ASIC technology explosion

- I IC 기술의 급속한 발전은 그것에 걸맞는 설계 툴의 개발을 요구
 - â 이것은 보다 복잡하고 커다란 IC설계를 가능케 함
- I ASIC설계를 위한 설계방법론 및 툴 개발의 지속적 발전
- I Two major advances
 - „ Hardware description language(HDL)
 - „ Powerful logic synthesis system

The Technology of IC

- I SSI (Small-scale integration)
 - Several logic gates in a single package
- I MSI (Medium-scale integration)
 - The IC have a complexity of 10 to 100 gates
- I LSI (Large-scale integration)
 - The device performs a logic function with more than 100 gates
- I VLSI (Very Large-scale integration)
 - The device that contains thousands of gates in a single chip

Logic Synthesis

- I General or “random” logic functions과 관련한 것에 국한
- I 광의에서 ASIC vendor에서 극히 특정적이고 규칙적 논리구조를 갖는 RAM, ROM과 같은 메모리나 FIFO등에 사용하는 silicon compiler, function generator를 포함

Logic Synthesis (II)

- I 전통적인 schematic-based design과 달리 VHDL과 같은 언어와 함께 사용하여 language로 기술된 회로를 gate level의 logic으로 만들어준다.
- I 따라서 synthesis는 논리설계자가 VHDL을 이용하여 보다 높은 level에서의 설계를 가능하도록 해주는 도구

VHDL (Very high speed IC Hardware Description Language)

- I 1980년대에 미국정부와 공군의 지원으로 개발
- I 1987년에 IEEE에 의해 standard hardware description language로 채택됨

Advantages

(gained by moving to a VHDL synthesis methodology)!

ž Shorter design cycles

- ˆ higher level VHDL simulation
- ˆ design error를 줄일 수 있다(기능상)
- ˆ 기능구현 위한 VHDL code의 size는 회로의 gate 수에 비례하지 않는다

Advantages (II)

ǯ Vendor and Technology Independence

- 대부분의 synthesis tool vendor들은 다양한 ASIC library 제공
- 설계자는 하나의 라이브러리에 제한되지 않는다

ǯ Lower Design Cost

- Design re-usability
- Shorter design cycles

Disadvantages

ž A change of culture

Schematic capture에서 language-base설계로 설계기법이 전환함에 따라 설계함에 있어서 새로운 사고를 요하며 많은 language 및 툴 교육이 요망됨

ž Cost of getting started

VHDL 및 합성기술을 익히고 설계 툴을 evaluation하고 구매하는데 많은 초기비용이 소요됨

Disadvantages (II)

Learning and Training

ASIC 설계를 위한 VHDL 및 합성 기술은 단순히 language의 syntax만의 학습으로는 안된다. 설계 방법 전체에 대한 훈련이 필요함

(합성에 사용되는 language 사용을 위하여 VHDL language 전체를 습득할 필요는 없다)

Disadvantages (III)

ž Debugging Design Problems

VHDL은 synthesis를 통하여 회로를 생성시킨다. 다시 말해 결과적으로 생긴 gate-level design은 설계자가 만든 것이 아니라 컴퓨터에 의해 생성된 것임

이러한 문제들은 design cycle 초기에 testability를 고려한 설계를 하거나, timing analysis 툴들을 이용함으로써 어느 정도 문제들을 최소화 할 수 있다

What can be synthesized?

- I 모든 디지털회로 설계들은 기본적인 논리기능을 구현하기위한 기본 소자들로 이루어진다. 이러한 기본적인 기능들은 gate나 flip-flop과 같은 기본 로직 셀들로 구현한다
- I Combinational logic, Sequential logic 모두 합성 가능

What can be synthesized? (continued)

- | Combinational Logic
- | Sequential Logic: Combinational Logic + clocked storage device

| Combinational functions

Multipliers, Decoders, Encoders, Comparators, Adder, Subtractors, ALUs, Multipliers, PLA structures

What can be synthesized? (continued)

I Sequential Logic(기능별로 3개의 그룹으로 나눔)

Counters:

Binary, BCD, Johnson, Gray, Up/Down Counters

Memory address counters, FIFO memory pointers

Register and latch functions:

Data registers and latches, Shift registers,

Accumulators, Parallel/Serial converters

What can be synthesized? (continued)

Control logic:

Finite state machines

I ASIC구현 시 빠르고 효율적인 메모리등 구현할 때 합성하지 않는다(Flip-flop과 gate들을 이용하여 회로 합성하는 것은 효율적이 아님)

->대부분의 ASIC vendor들은 specialized compiler tool들을 사용

What can be synthesized? (continued)

- I 회로구현 시 “Random Logic”부분의 합성을 효율적으로 함
- I Timing critical한 설계를 하고자 할 때 synthesis tool은 요구하는 timing limit를 guarantee 할 수 없는 경우도 있다
 - Chip layout시 발생하는 delay문제 등으로
 - 기본설계 문제

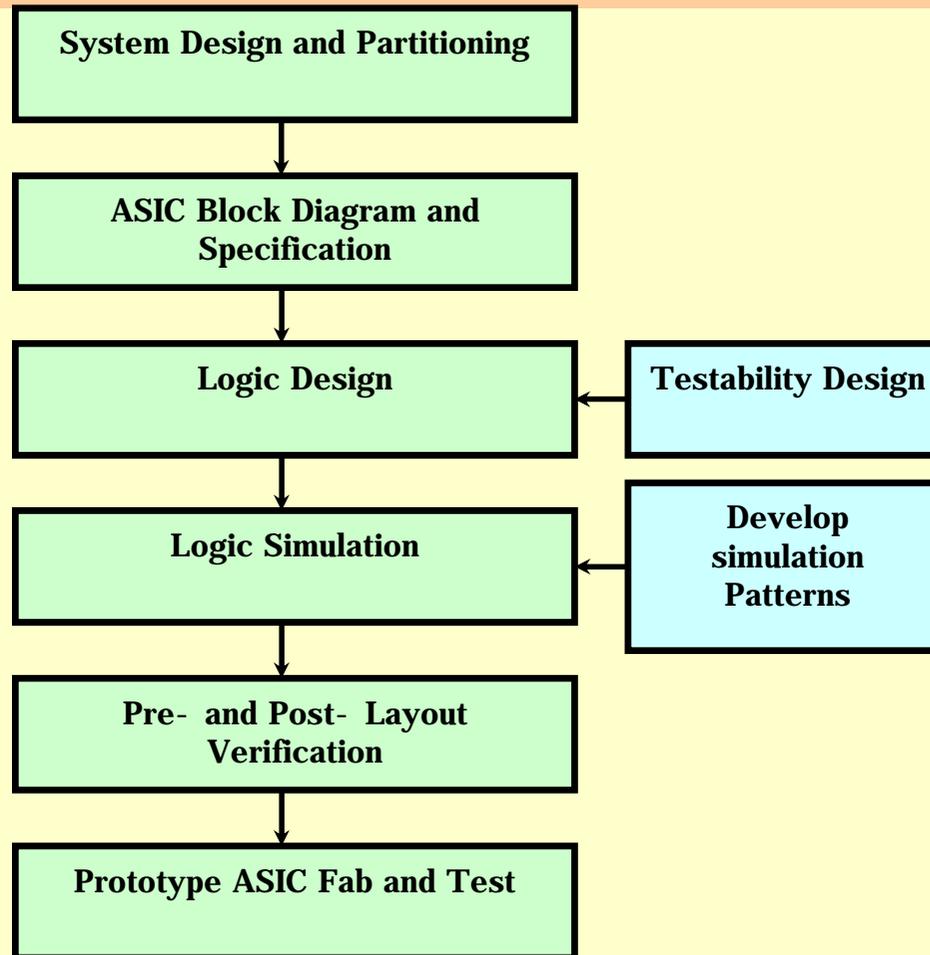
Some Basic Synthesis Premises

- I Synthesizer의 성능은 어떻게 VHDL을 사용하느냐에도 달려있다
- I VHDL로 기술만 하면 synthesizer가 모든 것을 알아서 해주는가?
- I 효율적인 설계와 합성 후 같은 시뮬레이션 결과를 얻기 위해서는 어떤 rule이 필요하다(=>합성가능 syntax)

Some Basic Synthesis Premises (II)

- I 부적절한 VHDL표현은 합성불가 또는 비효율적인 논리회로를 생성하게 만든다
- I 일반적으로, 성공적인 논리합성을 위해서는 논리설계기술경험 및 ASIC설계경험이 상당히 중요한 요소가 된다
- I 또한 설계하고자 하는 바를 synthesizer가 곧바로 해석할 수 있도록 VHDL로 명확하게 기술한다

The ASIC Design Process



System Planning and Definition

- I 시스템 요구사항을 분석, 이용하고자 하는 알고리즘의 trade-off(경제성 등 고려), 하드웨어로 구현할 것인지 소프트웨어로 구현할 것인지 결정
- I 결과적으로 세부 spec. 도출 (high level block diagram 만들기)

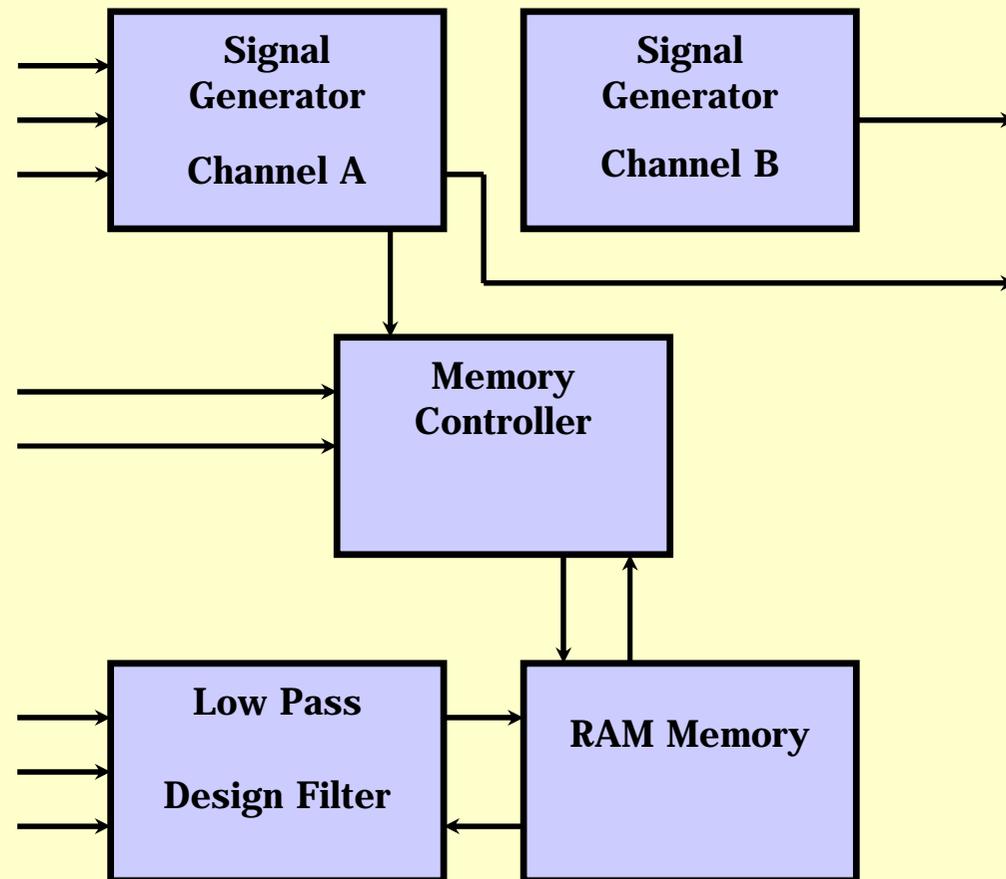
Block diagram and Specification

- I 시스템 설계에서 도출된 high level block diagram(digital logic)을 실현할 세부 블록으로 나눈다(Boards, ASICs, discrete ICs)
- I 이와 같은 결정은 weight, size, cost, performance 등을 고려하여 결정한다
- I 이 단계에서는 시스템에서의 ASIC의 구체적 spec. 그밖에 사용되는 logic function, 시스템에서의 다른 블록과의 interface, clock speed등이 구체적으로 나타나야 한다

Block diagram and Specification(continued)

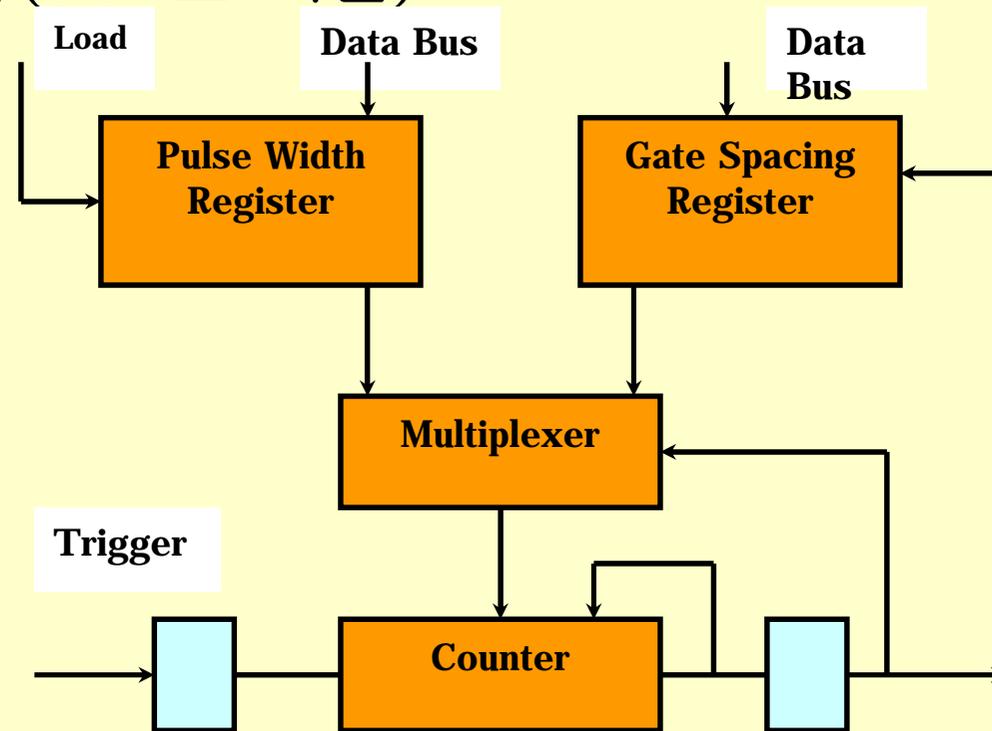
- I 중요사항: spec. 이 정확히 확정되지않은 부분은 절대로 ASIC에 포함시켜서는 안된다. (spec. 이 확정 될 때까지는 FPGA들로 구현해야 함)
- I ASIC의 복잡도에 따라 Block diagram은 몇 개의 hierarchy를 가질 수 있다
- I 또한 top level에서 중요기능별 블록 다이어그램을 만들 수 있다.

High level Block Diagram



Block diagram and Specification(continued)

설계를 위하여 각 주요 블록들은 세부 로직들로 만들 수 있다(VHDL로 기술)



Testability and Simulation Planning

I Testability Planning

P 시스템의 논리블럭을 정의함에 있어서 testability도 아울러 고려해야 한다

P Testability를 위한 additional logic을 만든다
이것으로 IC에 있는 모든 게이트들 각자가 올바르게 기능을 수행하는지 확인한다. 이를 위하여 많은 test pattern들이 필요하다

Testability and Simulation Planning(continued)

P 부적절한 testability는 칩 테스트 시 기능이 pass되었으나 실제 시스템에 적용하여 최종 테스트 시 기존에 발견되지 않았던 문제들이 나타날 수 있다

P 설계초기부터 testability를 고려한 설계가 중요
많은 장비 회사들도 이와 관련된 툴을 판매

Testability and Simulation Planning(continued)

I Simulation Planning

P 시스템 적용 시 ASIC의 정확한 기능검증을 위해 어떻게 시뮬레이션 해야 할 것인가

P 어떻게 ASIC에 입력신호를 넣을 것인가?

P 어떻게 multiple ASIC/board level simulation을 다룰 것인가

Testability and Simulation Planning (continued)

I Detailed Logic Design

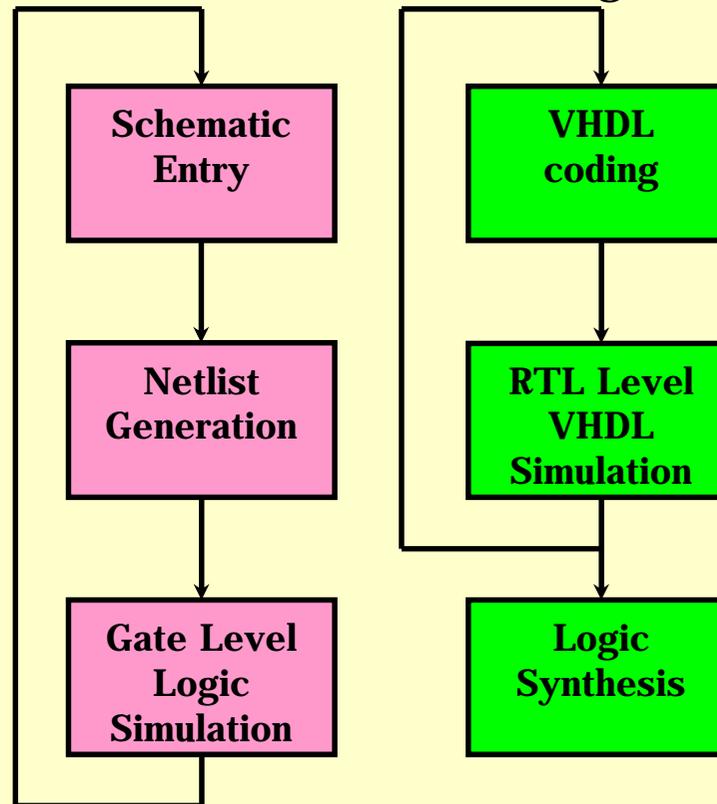
P 설계자 입장에서는 schematic으로 설계된 것이 훨씬 더 회로분석이나 시뮬레이션 결과를 검증하는데 있어서 편리하다

P 합성되어 나온 schematic회로는 사람이 그린 것이 아니기 때문에 규칙성도 없고 정렬되어 있지 않기 때문에 분석에 다소 어려움 있음

P VHDL에 익숙하다면 schematic은 단지 critical circuit을 체크 할 때 또는 timing 문제가 발생 되었을 때만 보게 됨

Testability and Simulation Planning (continued)

Comparison of detailed design flows



Simulation

- I 블록 다이어그램이 완성되고 회로에 대한 VHDL coding이 완료되고 나면 다음 단계는 VHDL simulator를 이용하여 설계된 회로를 검증하는 단계가 필요함
- I 이 단계에서는 gate level에 대한 것이 아니라 RTL(Register Transfer Level)이라고 불리는 behavioral level의 검증이다
- I VHDL simulator는 source file을 컴파일하고, 에러를 체크하고 설계회로의 각 블록들을 link시켜주는 의미에서 기존의 simulator와 같다

Simulation (continued)

- I 회로가 정상 동작하는지의 검증을 위하여 waveform이나 tabular listings등을 이용하여 원하는 입력 및 출력신호를 모니터 한다는 점에 있어서도 기존의 simulator와 같다
- I 기능상에 문제가 발생되었을 때 schematic방식으로 설계된 회로는 schematic상에서 gate by gate로 잘못된 부분을 찾아 수정
- I VHDL설계방식에서는 VHDL code를 체크하여 오류를 수정한다

Simulation (continued)

- I 위 두 경우 모두 회로상의 오류를 검증하기 위하여 additional internal signal들을 사용하여 이들을 모니터 함으로써 어느 부분에 오류가 있는지를 찾아낸다
- I 이러한 과정이 모든 설계 블록에서 기능이 검증될 때까지 계속된다

Simulation (continued)

I simulation 방식

P 설계의 각 서브 블록들을 하나씩 simulation하는 경우는

=>설계가 매우 복잡하거나, 각 블록의 세부 블록을 일일이 검증해야 할 필요가 있는 경우, 그리고 VHDL초보자인경우에 이 방법이 좋다 단, 이 방법을 이용할 경우 많은 검증 시간이 요구되며, 하나의 sub-block의 검증을 위해 사용된 test-bench들은 설계전체의 기능 검증 시 사용 할 수 없고 최종기능 검증을 위하여 추가적인 test-bench가 필요함

Logic Synthesis

- I Simulation결과는 단순한 기능검증이며 회로의 “delay”등 하드웨어적인 요소는 포함되지 않는다
- I Synthesis시
 - 첫째, 각 library들은 개개의 gate delay들을 가지기 때문에 Target library를 (ASIC vendor) 선정해야 한다
 - 둘째, 설계자의 요구에 맞는 design constraint를 준다 (예, clock rate, operating temperature, voltage, permissible propagation delays through critical paths 등)

Logic Synthesis (continued)

- I 처음에 synthesizer는 VHDL code를 multiplexer, ALU, decoder, register 등과 같은 low level logic building block들로 바꾸면서 효율적 이용을 위하여 이중 어떤 로직 블록들을 공유 할 수 있는지 판단한다
- I 다음에는 generic function들을 vendor specific library cell들로 바꿔주고 speed constraint들을 만족 하도록 optimization을 하며 speed가 critical 하지 않은 경우에는 logic minimization을 수행한다

Logic Synthesis (continued)

I Synthesis tool의 출력

P Vendor specific net-list

P Timing, gate count, critical path 등에 대한 report문

P Schematic diagrams

I ASIC vendor는 이 net-list와 schematic들을 chip bonding and layout, gate level logic simulation, routing전후의 최종 timing check를 위하여 ASIC vendor design verification tool에 이용한다

ASIC verification

- I 이 과정은 (pre-route signoff라고도 함) ASIC의 layout 및 routing이전에 최종적으로 gate level에서 검증하는 단계이다
- I 이 과정에서는 gate level logic simulation, timing analysis, design rule checking, electrical rule checking, I/O pin assignment등을 수행한다. 이러한 업무는 특정 ASIC vendor의 소프트웨어 시스템을 이용하거나 vendor가 인정하는 general purpose CAE system을 이용한다

ASIC verification(continued)

- I 어쨌든 이들의 목적은 layout이전에 에러가 없는 회로임을 검증하기 위한 것
- I 시뮬레이션 라이브러리와 synthesis시 적용하는 rule은 ASIC vendor와 synthesis vendor와 공동으로 개발되었기 때문에 synthesis와 timing optimization결과는 정확하다

ASIC verification(continued)

- I 문제는 layout이후에 발생하는 routing delay 이다 (Speed critical 하지않은 회로에서는 큰 문제가 안되나 propagation delay나 gate spike등이 회로에 영향을 줄 수 있다)
 - I Timing 또는 그 밖의 문제가 발생되었을 때 원인규명을 위하여 ASIC verification system의 netlist, block diagrams, schematics등과 함께 report 메시지등을 이용한다
- 경우에 따라 재합성 또는 VHDL code를 수정한다

A Few Observations

- I VHDL이 하나의 language이자 software이지만 ASIC 설계를 위해 VHDL을 이용하고 합성하는 것은 논리 회로설계를 하는 과정이다. 따라서 VHDL coding은 단순한 software programming이 아니라는 사실을 명심하라!
- I The most successful approach to synthesis is to code the VHDL to produce what you want, rather than having the synthesizer try to figure out what you want! (18쪽 참조)

A Few Observations (continued)

- I 설계하는 동안 simulation은 수없이 반복되어 수행된다. 이때 논리회로 simulation시 사용한 test-bench들을 final design verification을 위해 사용하는 ASIC vendor's certified simulator 또는 “golden” simulator에도 똑같이 적용하여 같은 결과를 얻어야 한다

VHDL for Synthesis

- I VHDL은 abstract behavior의 가장 높은 레벨에서부터 가장 낮은 gate level structure까지 표현할 수 있는 아주 광범위한 능력을 보유한 언어이다
- I 따라서 VHDL은 회로설계 시 발생하는 문제들에 대해 아주 융통성 있게 대처 할 수 있으며 완벽한 top down system design methodology를 이용한 회로 구현이 가능하다

VHDL for Synthesis (continued)

- I VHDL의 기능은 logic synthesis에 필요한 것 이상의 능력을 보유하나 많은 abstract modeling feature들을 digital logic으로 대치 할 수 없다
- I 따라서 logic synthesis를 위해서 VHDL 의 subset 을 사용=> synthesis가능한 code만 사용

VHDL for Synthesis (continued)

- I Synthesis 가능한 VHDL은 표준화 되어있지 않고 각 vendor 마다 조금씩 다르다
- I Subset of VHDL만이 합성가능하나 합성 불가능한 syntax도 익혀야 함=>예를 들어 test benches 나 test patterns등과 같이 설계검증을 위하여 반드시 필요함