Introduction to Development and V&V of FPGA-based Digital I&Cs

김의섭

목차

01

[FPGA]

FPGA

- 프로그램이 가능한 비메모리 반도체의 일종.
- 회로 변경이 불가능한 일반 반도체와 달리 용도에 맞게 회로를 다시 새겨 넣을 수 있다.
- 따라서 사용자는 자신의 용도에 맞게 반도체의 기능을 소프트웨어 프로그램 하듯이 변형시킬 수 있다.
- 일반 반도체에 비해 가격이 수십~수백 배 비싸며 항공, 자동차, 통신 등의 분야에 주로 쓰인다

- 네이버 (한경 경제용어사전)

2014년 프로그래머블 반도체 기업 시장 점유율

자료:IHS

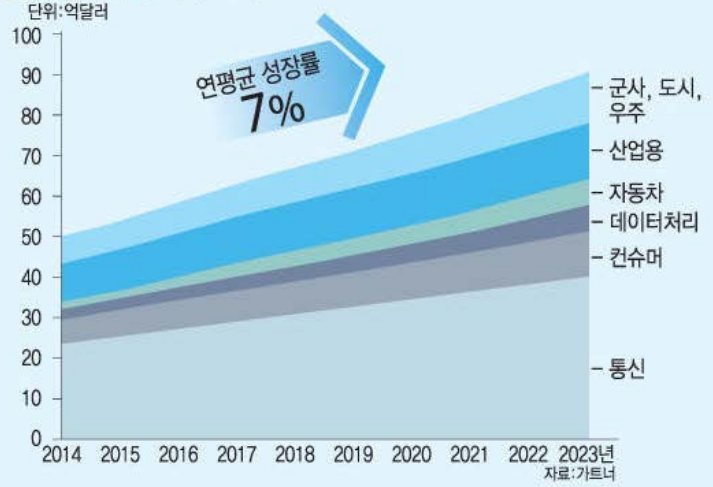
새로운 합병법인으로 본 작년 세계 반도체 기업 매출 순위

단위:백만달러

자료:IC인사이트

FPGA 시장 성장 전망

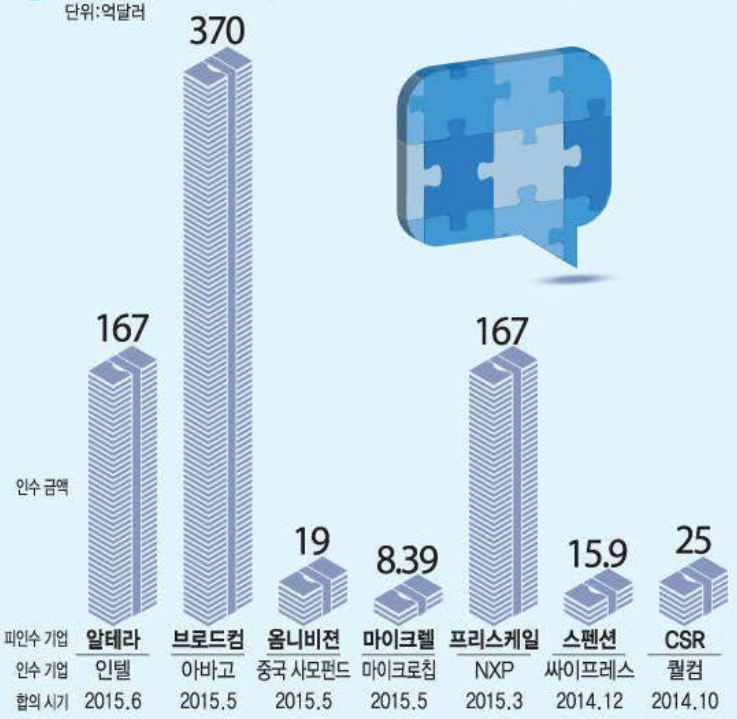
단위:억달러



자료:가트너

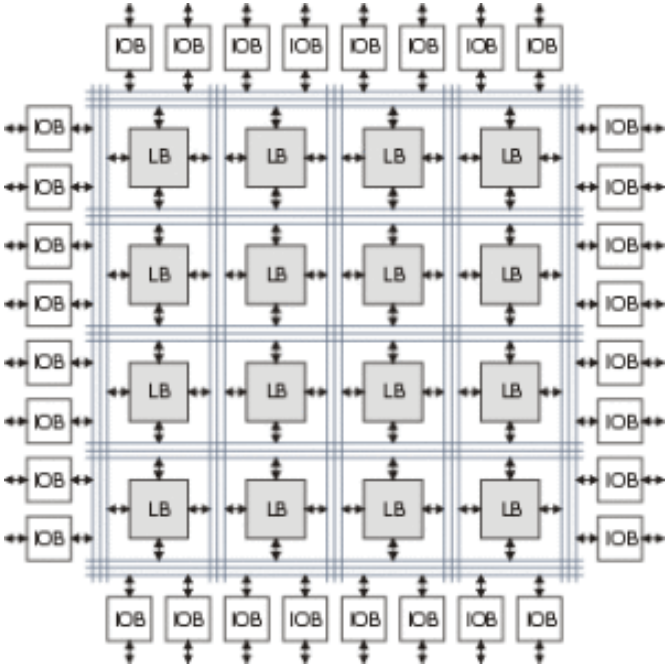
최근 발생한 주요 세계 반도체 시장 인수합병 사례

단위:억달러



FPGA

- FPGA (Field-Programmable Gate Array)
  - 프로그래머블 논리 요소와 프로그래밍가능 내부선이 포함된 반도체 소자
  - 넓은 평야(field)의 바둑판처럼 규칙적인 구획을 가진 배열 (Array)을 프로그래밍
  - Filed(사용자)에서 프로그래밍이 가능한 gate array(디지털 회로 반도체)

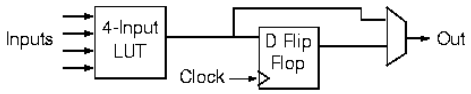


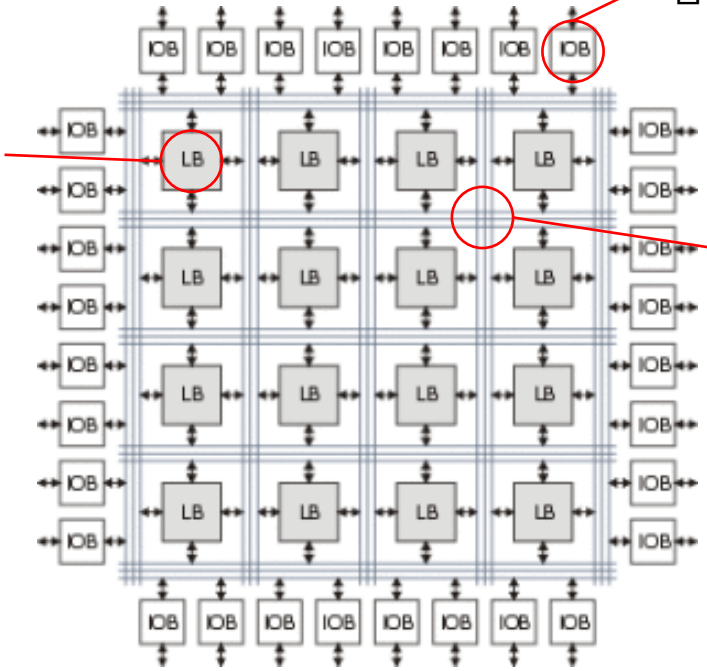
FPGA

- FPGA (Field-Programmable Gate Array)
  - 프로그래머블 논리 요소와 프로그래밍가능 내부선이 포함된 반도체 소자
  - 넓은 평야(field)의 바둑판처럼 규칙적인 구획을 가진 배열 (Array)을 프로그래밍
  - Filed(사용자)에서 프로그래밍이 가능한 gate array(디지털 회로 반도체)

입력 및 출력 회로 배치

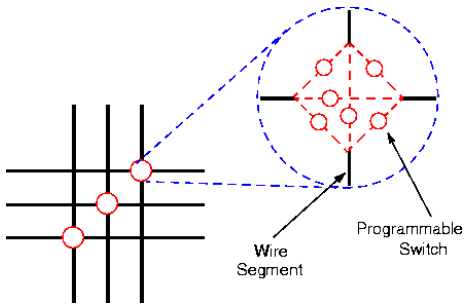
디지털 회로를 구현할 수 있는 게이트 (AND, OR 등), 플립플롭, 멀티플렉서, 메모리



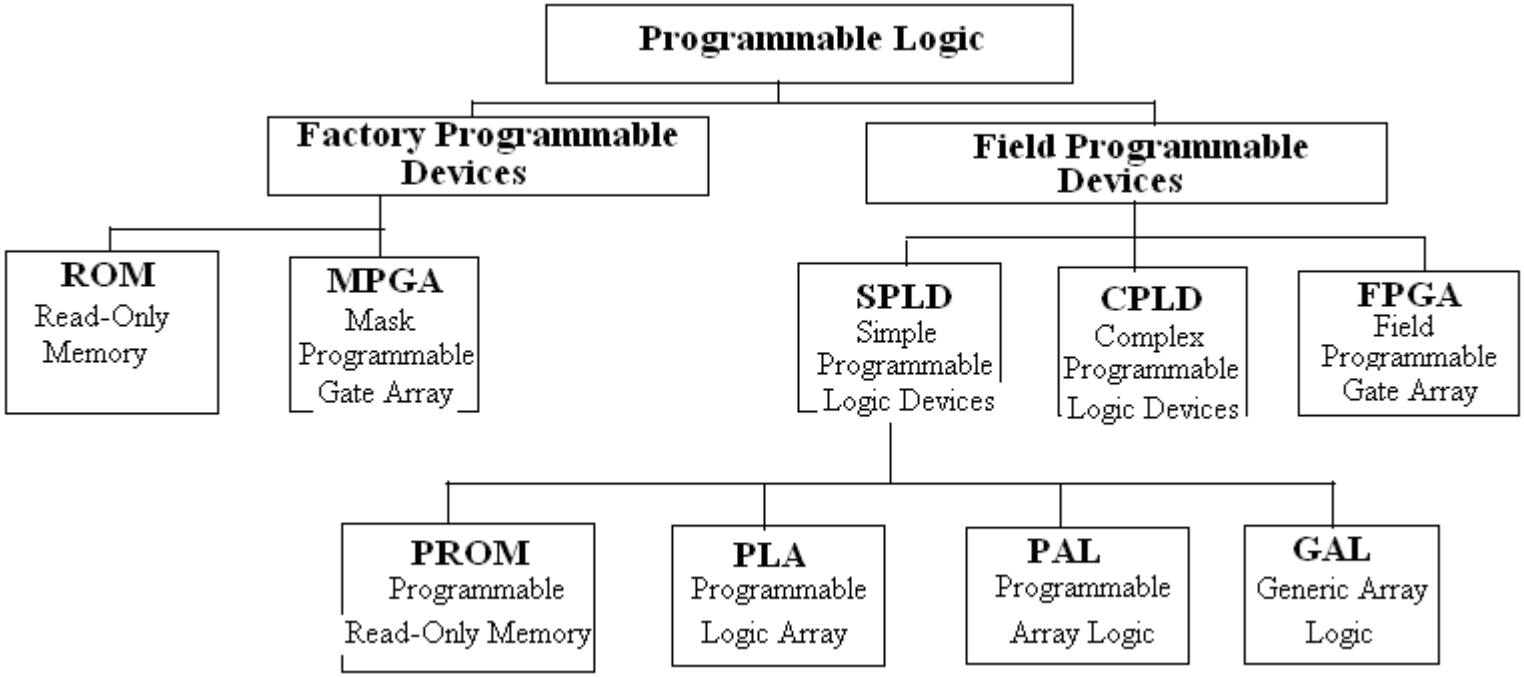


FPGA Design

Programmable Interconnect
로직과 로직 사이를 연결

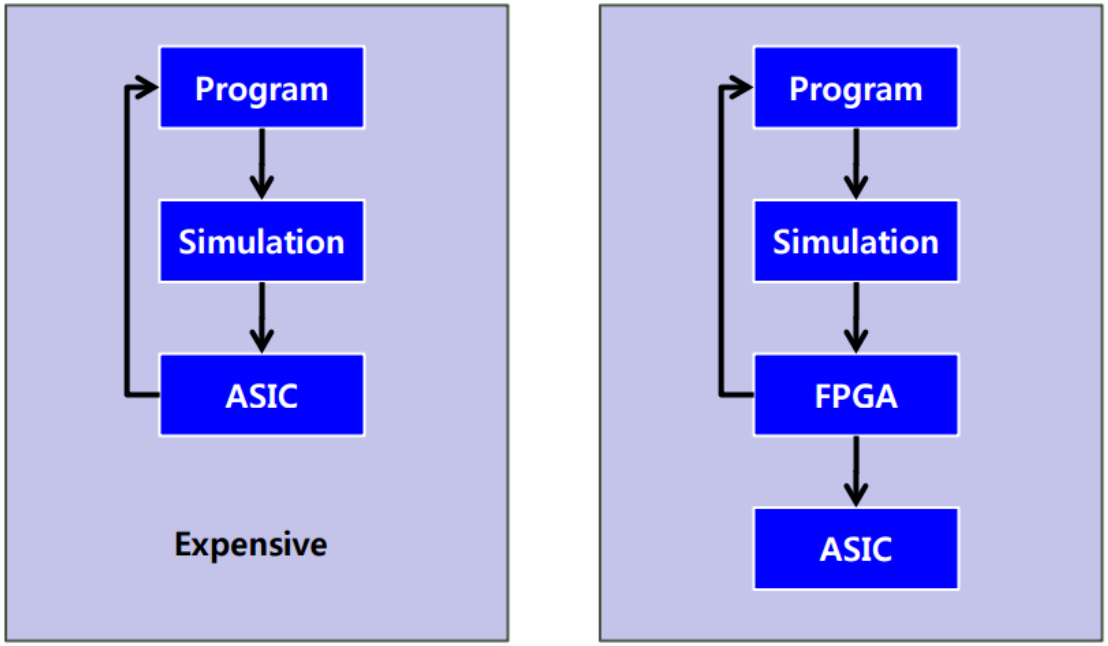


7


Programmable Logic
Factory Programmable Devices
Field Programmable Devices
ROM
Read-Only Memory
MPGA
Mask Programmable Gate Array
SPLD
Simple Programmable Logic Devices
CPLD
Complex Programmable Logic Devices
FPGA
Field Programmable Gate Array
PROM
Programmable Read-Only Memory
PLA
Programmable Logic Array
PAL
Programmable Array Logic
GAL
Generic Array Logic

FPGA

- 초기 FPGA는 ASIC 제품의 양산 전 단계의 시험용 제품으로 사용
  - FPGA는 ASIC으로 가지 전 성능과 동작을 테스트하기위해 중간 개발물 형태의 직접회로(IC)로 사용
  - 오류발생시 회로 변경이 가능한 특성(수정이 가능)
  - 개발시간이 짧으며 개발비용이 비교적 적음



FPGA 장/단점

장점

- **간편하게 설계한 로직을 반복적으로 이식할 수 있다**
- **Time to market이 좋다**
  - 빠르게 시장에 내다 팔 수 있다. (ASIC 대비)
  - ASIC은 한번 만드는데 대략 3~6개월 걸림 (설계상에 오류가 있다면 그만큼의 시간이 필요)
  - Non-recurring Engineering(NRE) charge를 내지 않아도 된다는 것이다.
    - - 이름그대로 NRE charge는 ASIC을 의뢰하여 샘플 받는데까 지 무조건 내야하는(공정에 따라 한번에 수천만원에서 수억원에 이름) 비용이고 칩 이 동작안한다고 하여 돌려받을 수 있는 돈이 아니다.
- **업데이트 가능**
  - FPGA는 SRAM타입의 경우 PROM파일만 바꿔주면 안에 내용을 바꿀 수 있음
  - 예를 들어
    - 해외에 수출했는데 오류가 발생 할 경우, ASIC은 전수 회수하여 칩을 교체해야 함
    - 반면, FPGA는 SW업데이트로 해결이 가능합니다.

단점

- **고비용**
  - 대량으로 생산하는 경우 (ASIC의 양산단가는 FPGA와 비교 안될 정도로 쌈)
- **Size문제**
  - FPGA에 비해 ASIC은 칩의 면적이 작음 (충분히 optimize 되기 때문)
  - 핸드폰 처럼 작은 사이즈가 필요한 제품은 FPGA로 설계하기 어려움 (발열이나 사이즈 문제)

원자력 도메인

- 원자로보호시스템(RPS)의 플랫폼 변경 요구 (**PLC → FPGA**)
  - 원전 디지털 계측제어시스템에서 공통원인고장(Common Cause Failure) 발생 가능성이 증가
    - 이를 방지하기 위해 다양성 및 심층방어(Diversity & Defense in Depth) 설계 개념 도입 필요
    - Security 측면
  - 기존 PLC 기반 소프트웨어의 유지보수 비용과 새로운 RPS 개발의 복잡성 증가

PLC (Programmable Logic Controller)

FPGA (Field Programmable Gate Array)

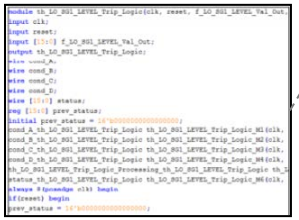
02

[Development Process / V&V]

Development Process

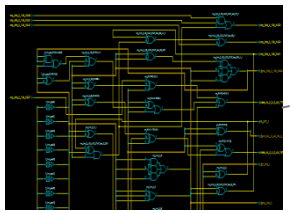
Requirement or Design Specification
- Functional Spec., I/O Signal Spec., Timing Const.

RTL Design
- HDL (Verilog or VHDL)

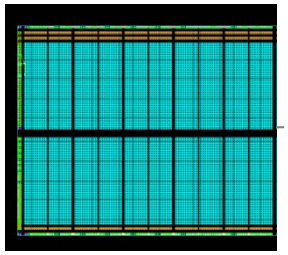


Synthesis
- 도구를 이용한 Compile (Optimization)

Gate-Level Design (Netlist)
- EDIF



Place & Route
- Chip spec 에 맞게 배치

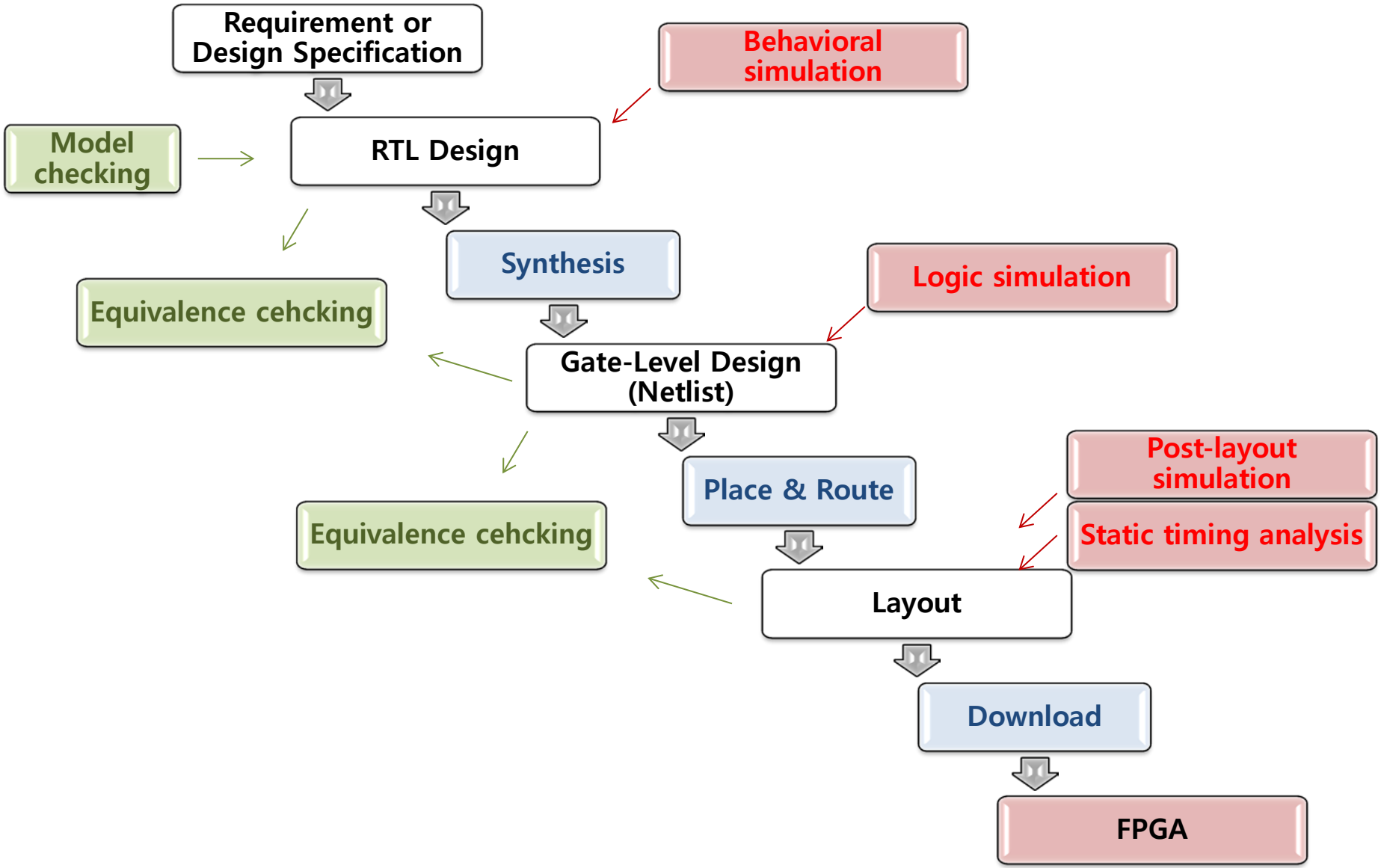


Layout
- JEDEC

Download
- bit stream

FPGA

Development Process + V&V



Requirement or Design Specification

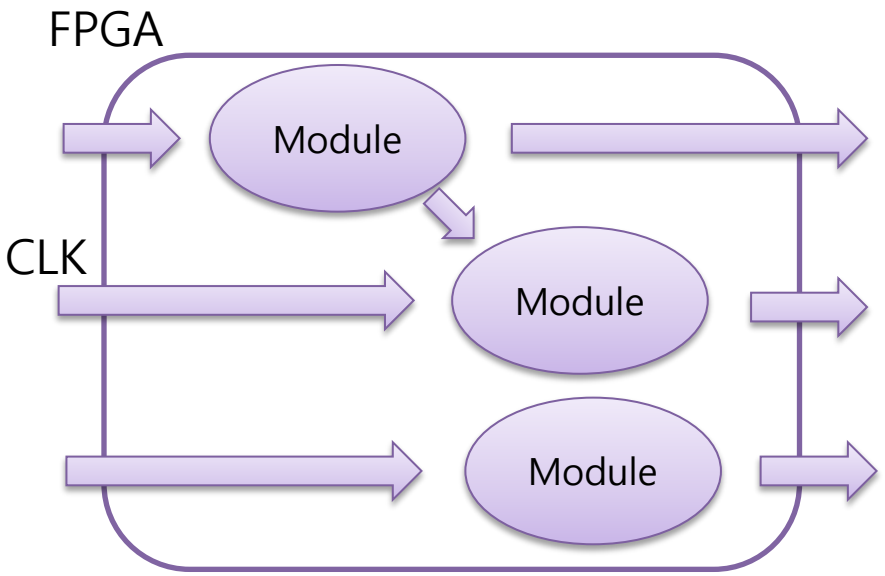
- 무엇을 어떻게 만들지 결정
  - 구현하고자 하는 function 분석
  - Verification plan
  - Project scheduling

- + chip과 system의 성능 분석
  - chip의 동작 clock 속도
  - Input / output 주기 분석

RTL Design

- **Register-transfer level**
  - 개발자는 Spec 을 **HDL (Hardware Description Language)** 로 구현 (coding)
  - 회로를 구현하는 세세한 게이트들과 그들간의 연결을 고려하지 않고 기능을 중심으로 구현 가능

- HDL
  - 반도체 및 전자회로를 기술하는 데 사용하는 컴퓨터 언어
  - 디지털 시스템을 모델링 (설계)
  - 기능
    - 시뮬레이션
    - 문서화
    - 합성
  - 언어특성
    - 동시성 (시간)
    - 병렬성 (공간)
    - 추상화
  - 대표적인 2가지 HDL
    - Verilog HDL
    - VHDL



Signal 또는 CLK의 천이에 따라
각각은 독립적(병렬적)으로 수행
(무순서 하게 명령어 실행: 동시성)

HDL

- **Verilog HDL**
  - 1983년 Gateway Design Automation 사에서 하드웨어 기술언어인 HiLo와 C언어의 특징을 기반으로 개발
  - EEE 1364로 표준화된 Verilog는 전자 회로 및 시스템에 사용되는 하드웨어 기술언어로, 회로 설계, 검증, 구현 등 여러 용도로 사용
  - **C 언어와 비슷한 문법**
    - if나 while, for loop 문 과 같은 제어 구조 / 연산자 : 논리 (&<^,|), 산술 (+,-,*,/)
    - C 언어와 달리, HDL의 특징인 시간에 대한 개념이 포함

- **VHDL**
  - 1981년에 제안, IEEE1076-1987,1993, 2000, 2002 표준
  - 미국 국방부에서 주문형 집적회로(ASIC)의 문서화에 사용하기 위해 만든 언어
    - 복잡한 매뉴얼로 회로의 동작 내용을 설명하는 대신, 회로의 동작 내용을 문서화하여 설명하기 위해 개발
  - 추가적으로 Simulation과 Synthesis 가 지원되면서 오늘날과 같은 형태의 문법으로 완성되어짐

RTL Verification

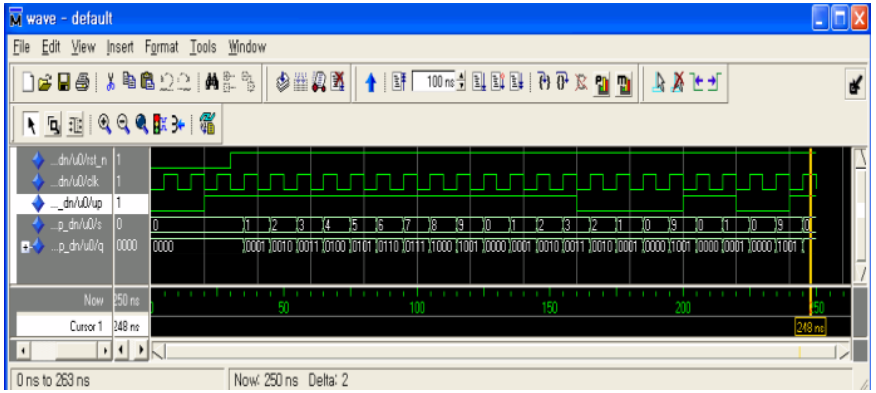
- Design이 끝났으면 design이 Spec 대로 올바로 구현 되었는지 검증을 해야 함
  - 개발한 Design이 specification을 만족하는지 확인
  - HDL은 hardware description 뿐 아니라 simulation을 위해서도 사용됨
- Behavioral simulation
  - Simulation은 일반적으로 가장 많이 사용되는 검증 방법
- Simulation 을 하기 위해서는 3 가지가 필요
  - 1. Test bench 작성
  - 2. Simulation
  - 3. Simulation 결과와 spec 비교

Simulation

- 1. Test bench 작성
  - Test Bench 란 테스트 입력을 인가하는 모듈
    - 매 clock cycle 마다 제대로 동작하는지 simulation vector라는 input data를 설계 모듈에 넣어 실행
  - **Issue - 모든 경우의 수를 확인하는 것은 불가능**
    - 따라서 충분한 Coverage 또는 Corner Case 를 커버하는 Test bench의 작성이 중요
    - 의미 있는 시뮬레이션 입력 벡터의 생성이 중요

- 2. Simulation
  - Simulation 도구를 이용하여 수행
  - **Issue - Simulation은 느리다**
    - Simulation tool은 HDL에서 기술된 모든 signal이 매 clock 어떻게 변하는지 monitoring을 하기 때문

- 3. Simulation 결과와 spec 비교
  - Simulation 결과는 Wave form 형태로 출력됨
  - **Issue – 결과 분석의 어려움**
    - 크기와 복잡도가 증가할 수록 분석에 시간과 노력이 필요

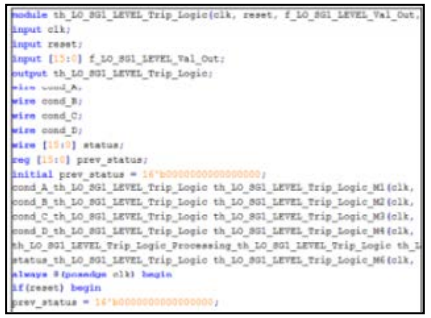


Synthesis 란?

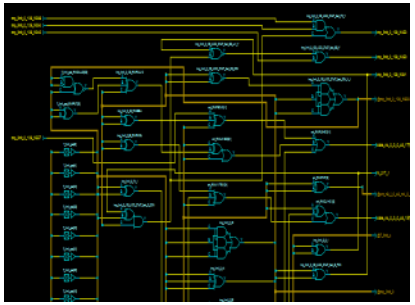
- 합성은 HDL을 이용한 설계에서 가장 중요한 과정

- 합성 = 변환 + 최적화
  (translation) (optimization)

- 변환 : 동작적 또는 RTL 기술 (HDL 설계)



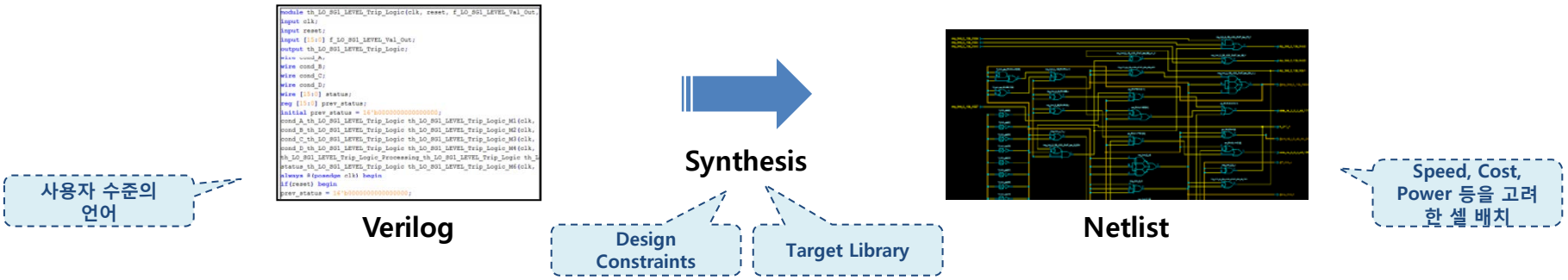
구조적 기술 (게이트 레벨 표현)



- 최적화 : 면적의 최소화, 동작의 고속화

Synthesis 란?

- **RTL 레벨의 HDL 로 작성된 Code를 Gate-Level의 optimization 된 Netlist 로 변환**
  - 원래는 사용자가 직접 Netlist를 작성해야 하지만 제품의 사이즈와 복잡도가 증가로 인해 직접 작성이 불가능 해짐
  - 사용자는 사용자 친화적 언어인 HDL로 제품을 설계하면 Synthesis 도구가 이를 지동으로 Netlist 형태로 변환 해줌
  - **사용자가 고려 해야 하는 복잡한 과정(Optimization)을 대신 수행**
    - **칩의 speed, cost, power 측면에서의 최적화를 자동으로 수행**
    - **따라서, 사용자는 HW의 고려 없이 디자인 및 설계에 집중**



21

Synthesis 의 장단점

- **설계 사이클 단축**
  - 설계 변경용이, 설계오류 가능성 저하, 간단한 기술 방식, 빠른 설계, 빠른 변경
- **설계의 질적 향상**
  - 여러 가지 architecture 시도 가능: 최적화
  - 면적 또는 속도 최적화 가능(논리 합성기의 기능)
    - Timing analyzer 기능 포함(critical path delay, setup, hold time 계산)
- **판매자, 기술(technology)에 무관한 설계 가능**
  - ASIC or FPGA 로 구현 가능, 어떤 vendor의 chip을 사용하든 무관
- 설계 비용 절감
  - 설계 기간 단축, 스키메틱 설계 과정 생략, 설계 오류 저하
  - Design reusability – IP, Library 그대로 또는 수정해서 사용가능

- **단점**
  - **논리 합성 결과, 자동 생성된 회로 도면을 이해하기 어렵다.**
  - **합성 도구 성능에 크게 좌우된다.**

Synthesis Tools

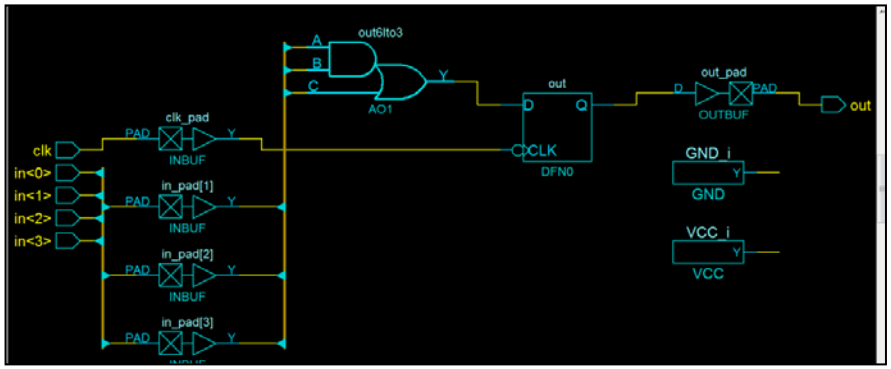
- **Commercial tool for logic synthesis (3rd party synthesis tool)**
  - **XST** (delivered within ISE) by Xilinx
  - **Quartus II** integrated Synthesis by Altera
  - **IspLever** by Lattice Semiconductor
  - **Encounter RTL Compiler** by Cadence Design Systems
  - **LeonardoSpectrum and Precision** (RTL / Physical) by Mentor Graphics
  - **Synplify** (PRO / Premier) by Synopsys
  - **BlastFPGA** by Magma Design Automation

Gate – Level Design (Netlist)

- Synthesis 후 나오는 결과
- 게이트와 그들간의 연결이라는 관점에서의 회로

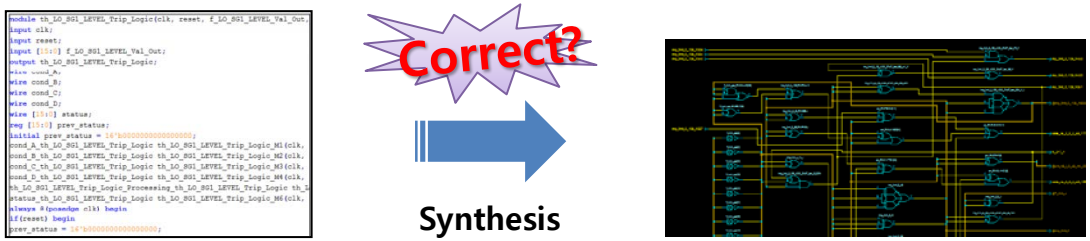
```
1
2  module Example_1 (clk, in, out)
3
4      input clk;
5      input [3:0] in;
6      output out;
7      reg out;
8
9      parameter a = 4 ;
10     parameter b = 1 ;
11
12     initial begin
13         out = 0;
14     end
15
16     always @(negedge clk)
17     begin
18         if (in > a + b)
19             out = 1;
20         else
21             out = 0;
22     end
23 endmodule
24
25
26
27
28
```

Synthesis →



Logic simulation

- Synthesis Tool 이 합성을 잘했는지 보장할 수 있을까?
  - Synthesis Tool도 software이므로 버그가 항상 존재할 수 있음
  - Vendor마다 최적화 또는 변환 룰이 각각 다름
  - Optimization 으로 인해 design의 정형성이 깨지게 됨
  - 일반적으로 좋은 결과를 제공하지만, 신뢰성 ↓
  - **따라서 합성 결과로 나온 netlist가 올바른지 검증 할 필요가 있음**
  - 특히, 원자력 분야에 아직 사용된 이력이 없어 신뢰성 및 안정성을 위해 검증 필요 (COTS dedication)



Logic simulation

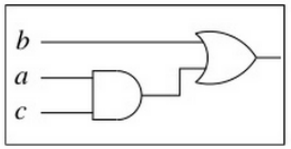
- Logic simulation
  - Netlist 가 의도한 대로 동작하는지 기능 확인
  - RTL level에서 사용한 test bench 를 사용할 수 있음

- **Issue -** 느린 simulation 속도 / RTL level의 simulation과 중복
  - Co-simulation 방법
  - 동일한 test bench 를 이용하여 simulation 을 수행 → 자동으로 결과 비교

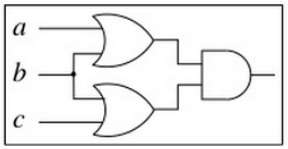
Equivalence checking

- Co-simulation 을 통해서 RTL 과 Netlist 간의 동일서 확인 가능
  - 하지만, 역시 모든 vector에 대해서 test를 진행할 수 없다는 단점이 있음

- 이를 검증 하기 위해 EC를 사용



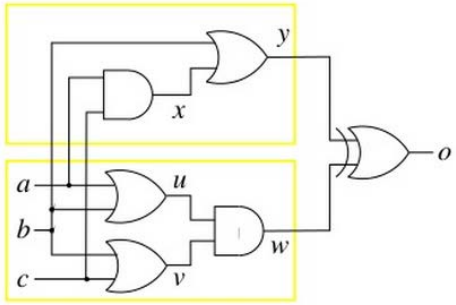
$b \vee a \wedge c$



$(a \vee b) \wedge (b \vee c)$

equivalent?

$b \vee a \wedge c \quad \Leftrightarrow \quad (a \vee b) \wedge (b \vee c)$



XOR gate

같으면 0
다르면 1

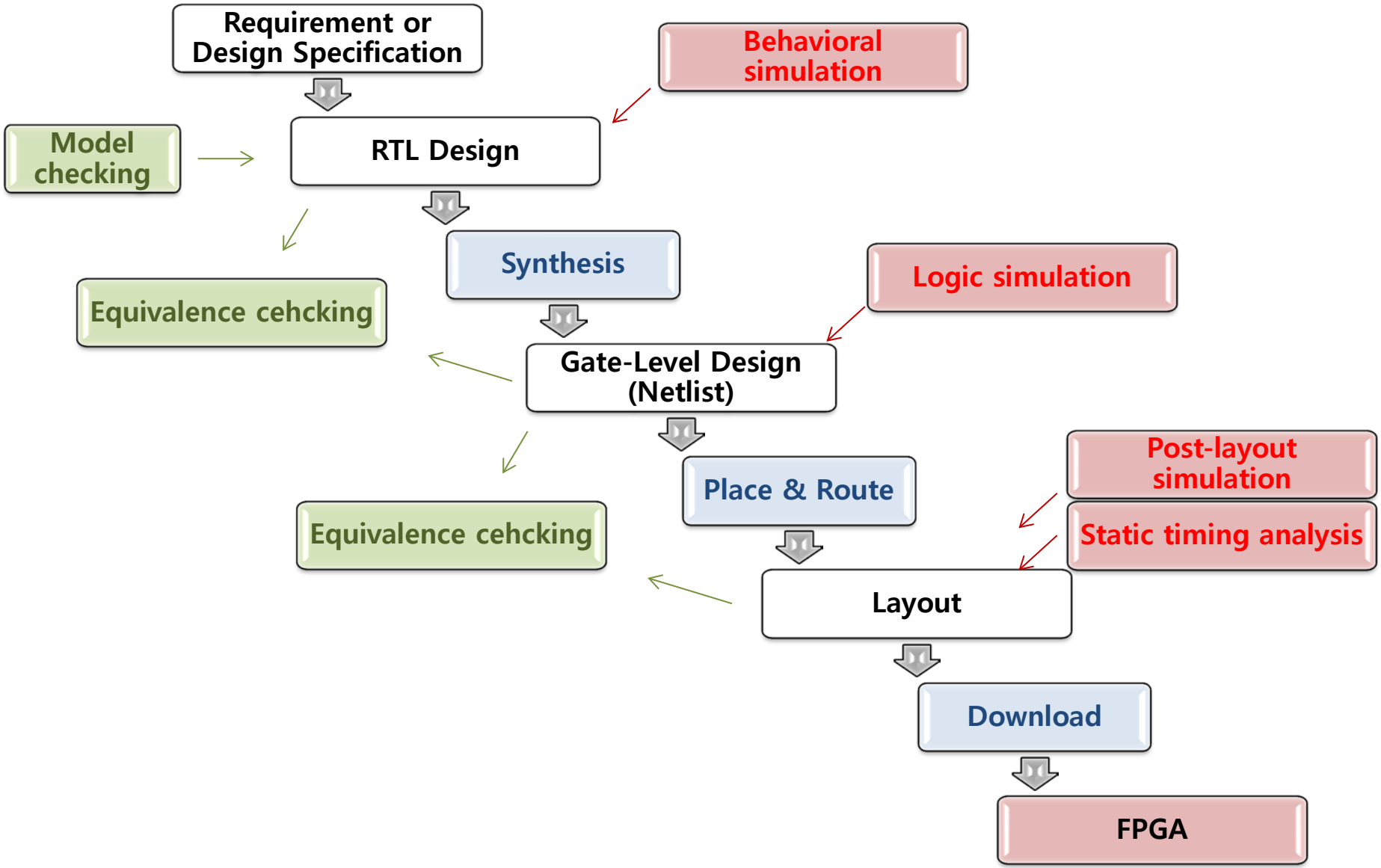
Equivalence checking

- EC 장점
  - 효율적인 검증 : 테스트 벤치를 생성할 필요가 없다.
  - 효과적인 검증 : 시뮬레이션에 비해 속도가 빠르다.
  - 정형 검증
    - 가능한 모든 입력에 대해 확인이 가능 하다.
    - 시뮬레이션 기반 방법은 불가능

- **Issue** – State explosion or 성능

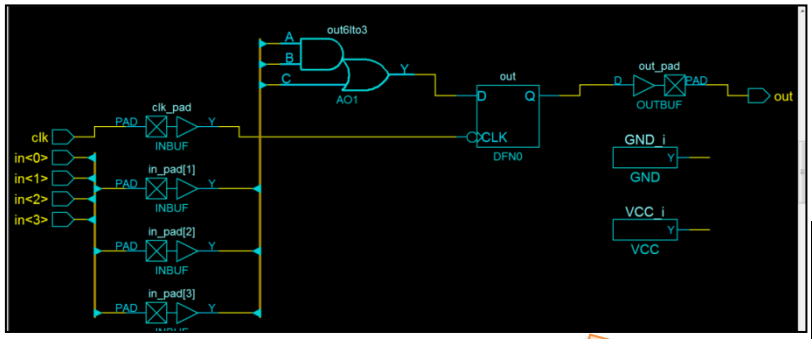
- **다양한 지원 Tools 존재 – 외산 / 상당한 로얄티**
  - **FormalPro** by Mentor Graphics / **Conformal** by Cadence / Jasper Gold by Cadence / **Formality** by Synopsys / Conformal LEC by Cadence Design Systems / Quartz Formal by Magma Design Automation / 360 EC by OneSpin Solutions

Development Process + V&V

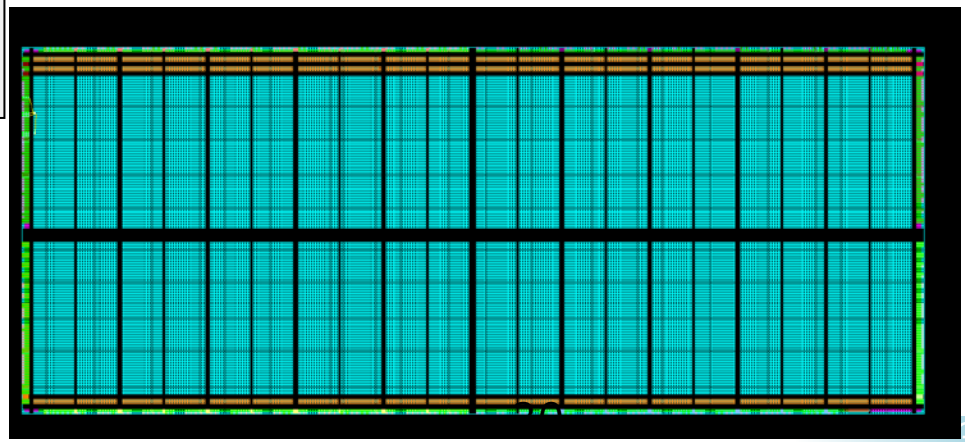


Place & Route 란?

- **Place & Route란**
  - Map 정보를 가지고 Logic Element를 FPGA내에 배치하고, Wire를 연결해주는 작업
  - Gate 수준의 Netlist 정보를 받아 Cell을 배치하고 Cell 사이를 route 함

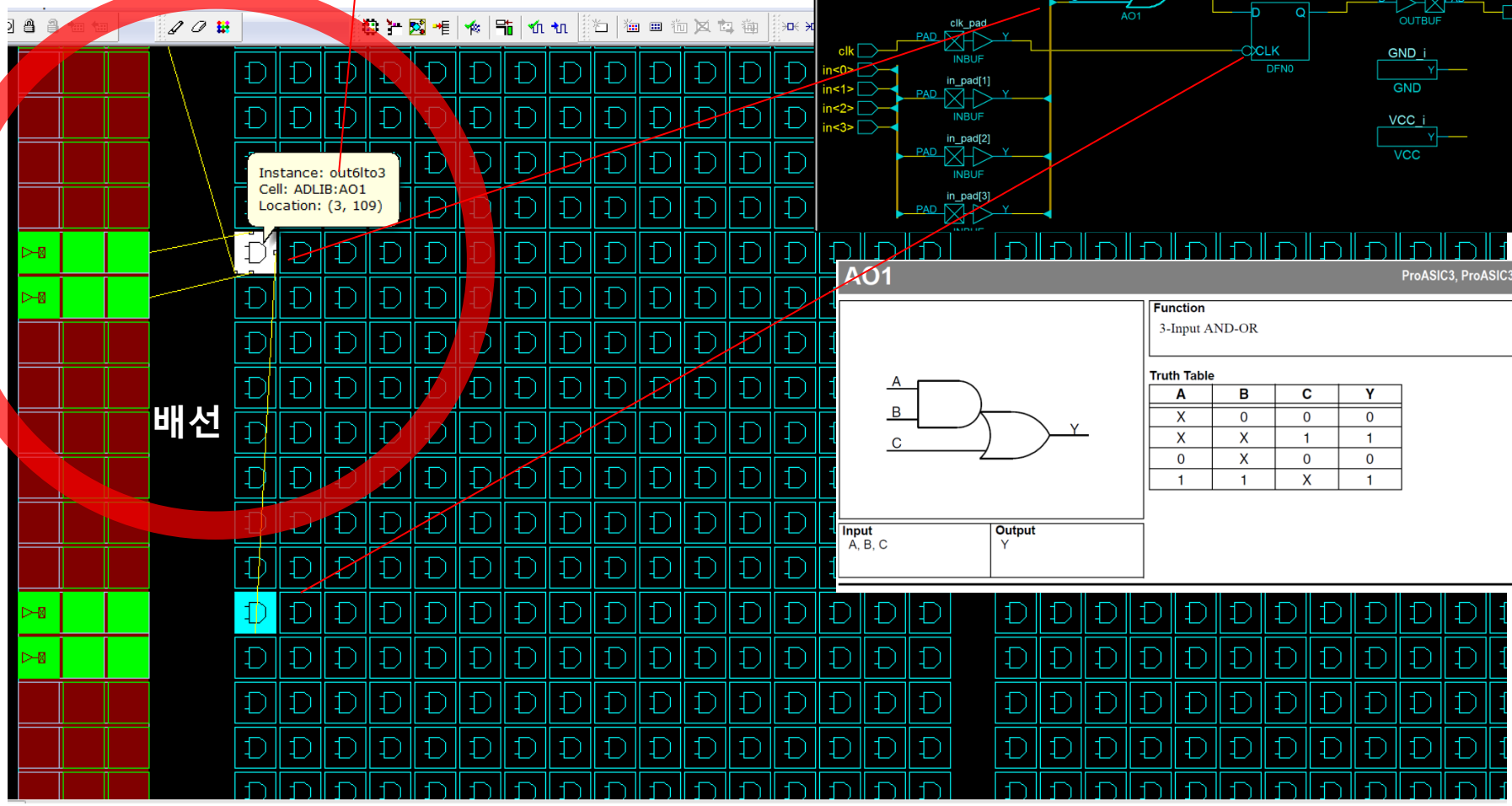


?



배치

AO1



AO1

ProASIC3, ProASIC3E

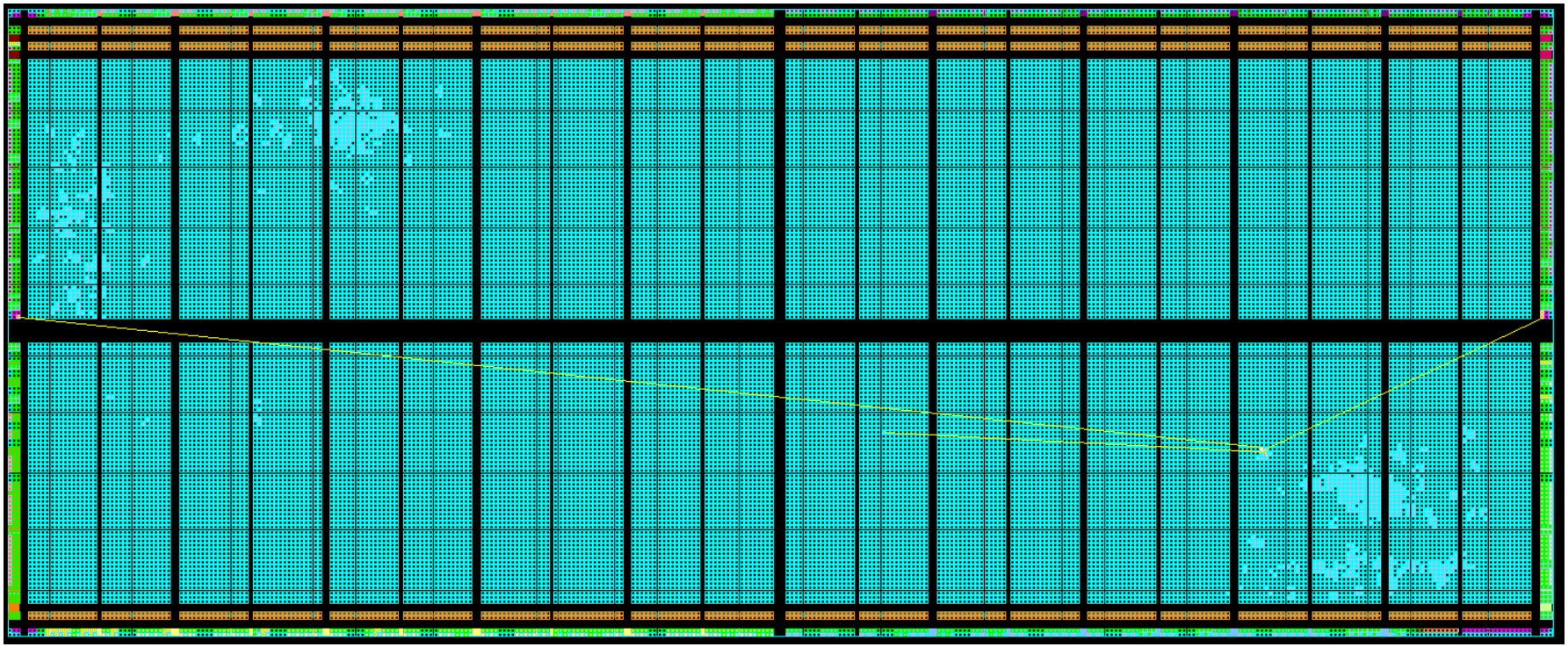
Function

3-Input AND-OR

Truth Table

A	B	C	Y
X	0	0	0
X	X	1	1
0	X	0	0
1	1	X	1

Input	Output
A, B, C	Y

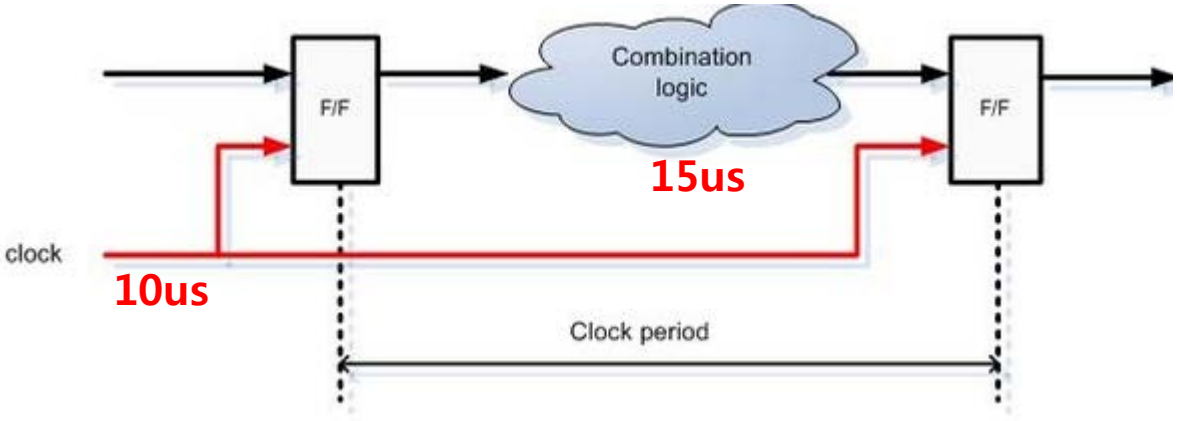


Post – layout simulation

- P&R 후 back-annotation 정보 (gate 들의 delay 정보 등) 을 이용해 시뮬레이션 수행
- Delay가 삽입되어도 동일한 기능을 (동일한 출력) 수행하는지 확인하는 과정
- **Issue -** 느린 simulation 속도 / behavior & logic simulation과 중복
  - Co-simulation 방법
  - 동일한 test bench 를 이용하여 simulation 을 수행 → 자동으로 결과 비교

Static Timing Analysis

- **FPGA 개발에 있어 타이밍과 동작기능은 항상 같이 고려해야 할 문제**
  - STA는 발생할 수 있는 **timing에 대한 문제를 사전에 처리**해주는 기능
  - 실제 수행(simulation) 없이 회로의 delay 시간을 계산하여 Violation을 확인
  - 설계자가 원하는 timing(clock 주파수나, delay 등)이 제대로 맞는 지 검증

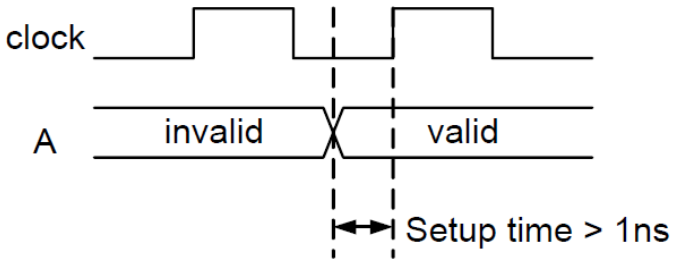


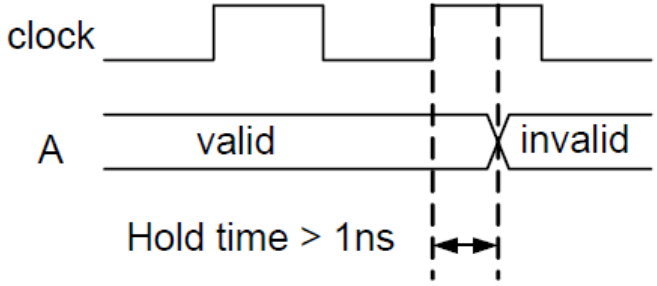
Combinational Logic이 상당히 복잡할 경우
두 번째 F/F에 원하는 Signal 이 도착 하지 못할 수 있다

35

Timing violation의 예

- **간단한 타이밍 위반(timing violation) 예로 두 가지가 있음**
  - **1. Setup timing violation**
    - 합성 Constraint가 너무 타이트 한 경우 발생
    - 합성된 결과가 클럭 주기보다도 큰 딜레이를 갖는 Combinational logic을 가질 때
    - **어떤 rising edge에서 data 값을 인식하기 위해서는 data는 clock rising edge보다 이전에 유효한 값을 일정 시간 동안 가지고 있어야 함**
  - **2. Hold timing violation**
    - hold time 동안은 유효한 데이터가 계속 변하지 않고 유지되어야 함
    - 예를 들어 상승 에지에 동작하는 플립 플롭이 입력 값을 인지하기도 전에 입력이 바뀌면 hold time violation이 발생
    - **아래는 홀드타인요구조건이 1ns인 경우이며, 이러한 경우에는 클럭상승 에지를 기준으로 최소 1ns동안은 값을 유지하고 있어야 함**

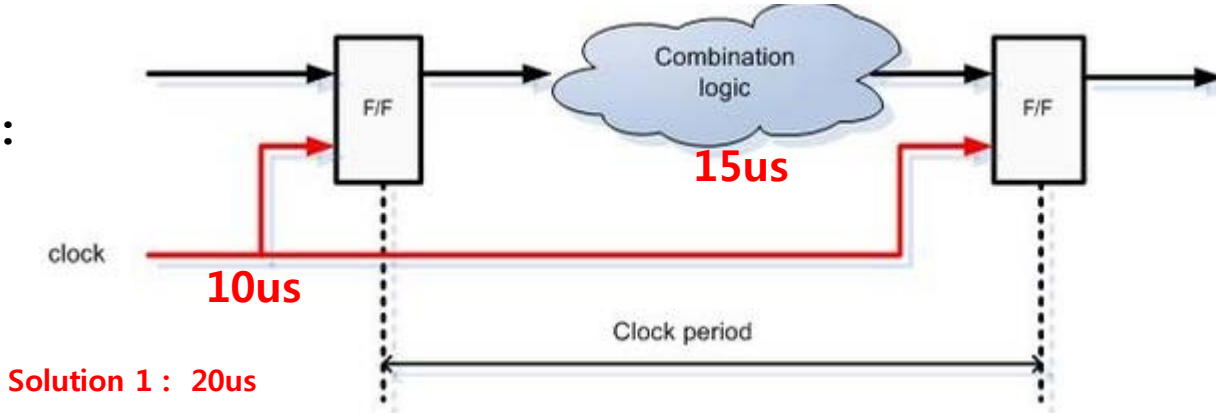




36

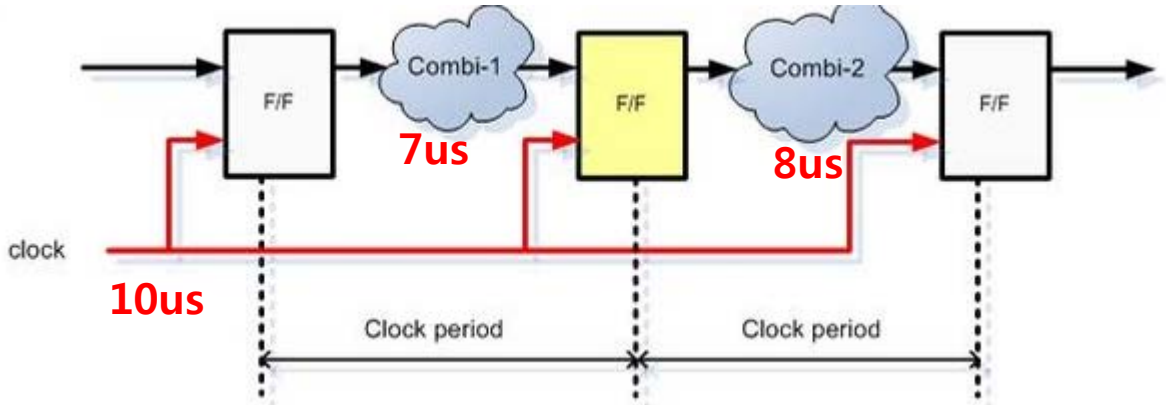
Ex) Setup timing violation

Original :



Solution 1 : 20us

Solution :



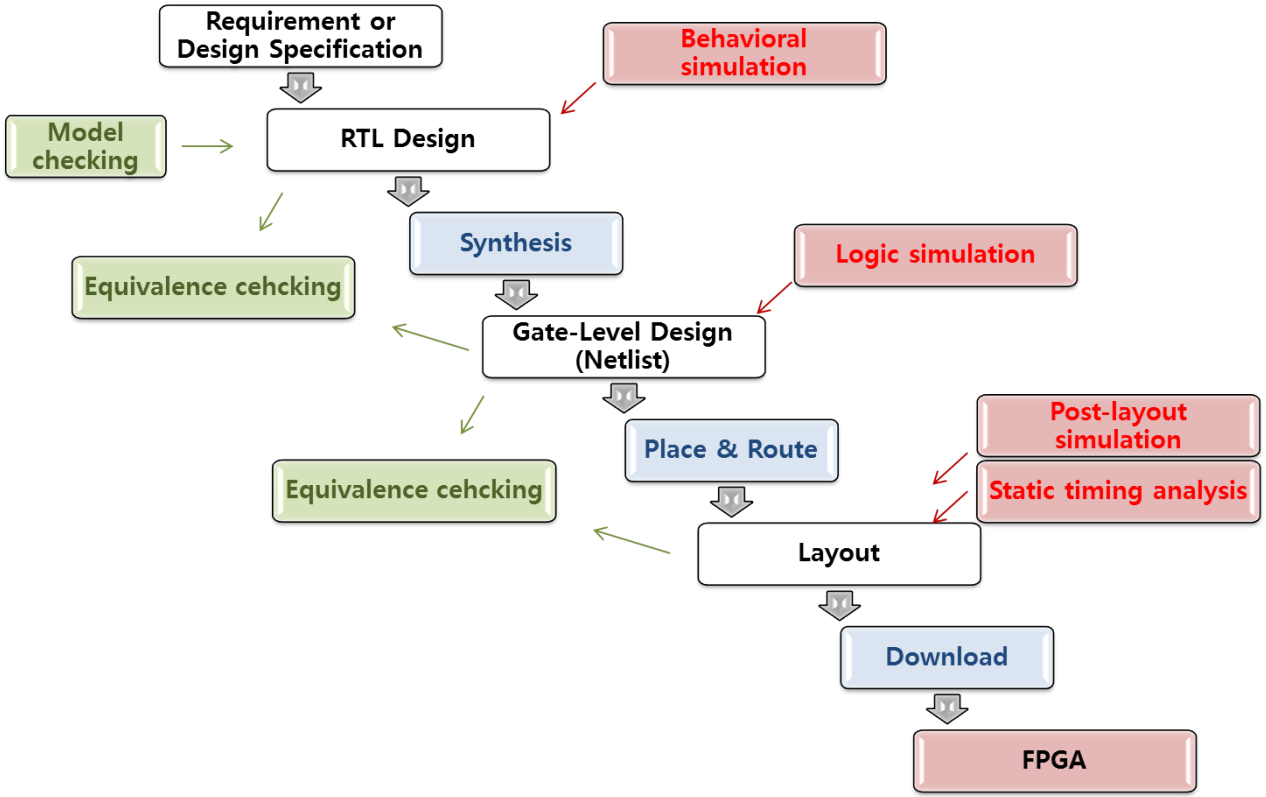
클럭 주기보다 Comb의 Signal 인가 시간이 짧도록 유지

Issue : 적절한 timing constraints?

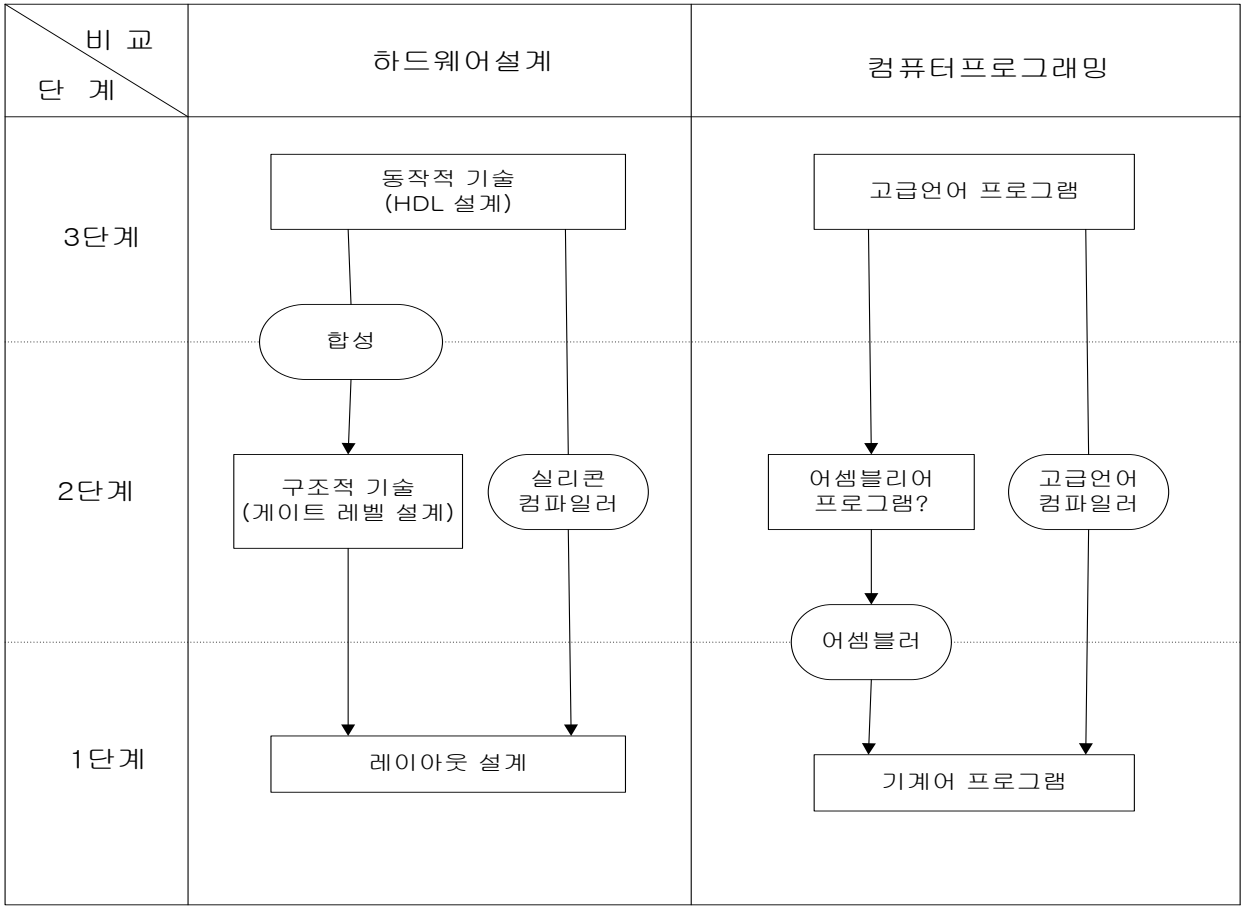
system 과 chip spec 에 대한 분석 필요

Summary

- **FPGA**
  - 개발자가 프로그래밍 가능한 반도체
  - FPGA 개발 + 검증 Process



- 행위 수준 합성 도구들도 등장 하고 있다고 함
  - RTL을 Layout 단계의 회로 행위로 변환



감사합니다.