디지털 논리회로
(Digital Logic Circuit)

수업 소개

- 교수: 이 채 은
  - E-mail: chae.rhee@inha.ac.kr
  - homepage: http://syd.inha.ac.kr
  - Office: 하이테크 518호
  - Office hour : e-mail로 약속

수업 소개

- I-class
  - http://learn.inha.ac.kr
  - 강의 자료
  - 공지 사항
  - 성적 게시
  - 과제 및 프로젝트 제출
  - 질의 응답: 교수님께/수강생에게
    - 이메일로 질문 받지 않습니다.

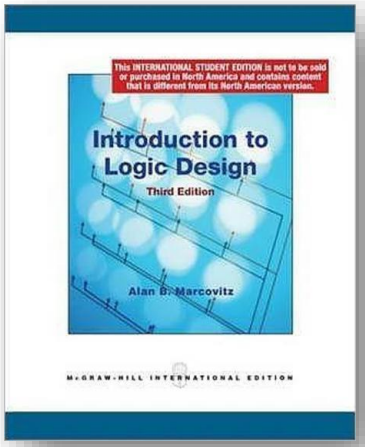
수업 관련 '개인적인' 질문을 메일로 보낼 경우

- 제목
  - **디지털 논리회로 OO 분반** 수업 ~~~ 질문입니다.

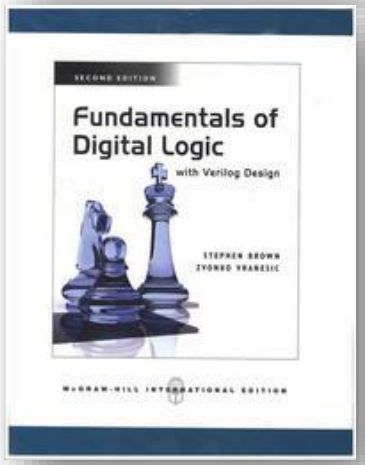
- 이메일 내용
  - 안녕하세요
  - **디지털 논리회로 OO 분반**
  - 수강하고 있는 **학번 XXXXXX 이름 ***** 입니다.
  - ~~~
  - 감사합니다.

수업 소개

- 교재



주교재:
Introduction to Logic Design (3rd edition), Marcovitz
McGraw-Hill



부교재:
Fundamentals of Digital Logic with Verilog Design, Brown
McGraw-Hill

이 수업에서는 무엇을 배우나?

디지털 논리

컴퓨터가 이해하는 논리(언어, 숫자)
하드웨어/소프트웨어에 모두 적용

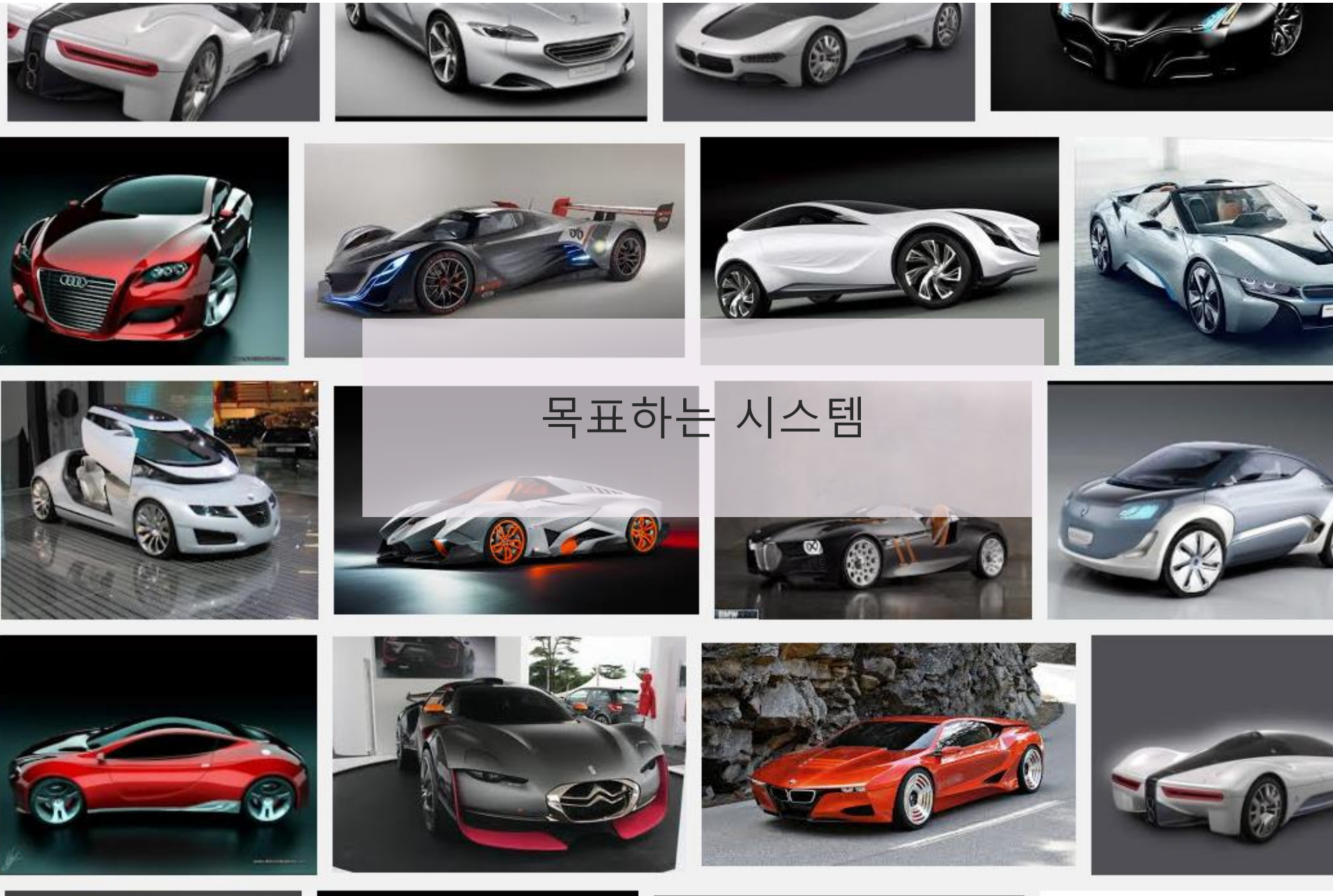
회로

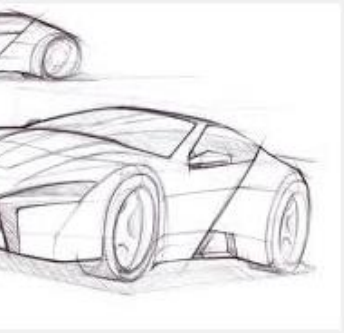
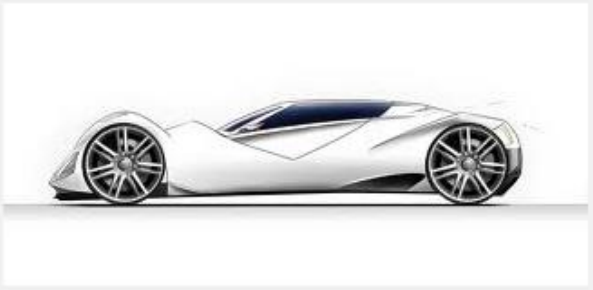
하드웨어에서 논리 구현을 위한 기본 단위들과 그 특성
Physical level이 아닌 Conceptual level에서의 회로

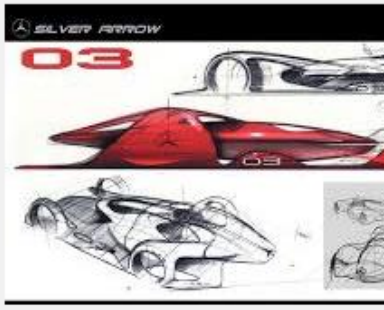
설계

원하는 입력과 출력이 주어졌을 때
논리와 회로로 만들 수 있는 '시스템' 설계
Conceptual 회로로 '크고 새로운' 시스템을 '빨리' 만들 수 있음

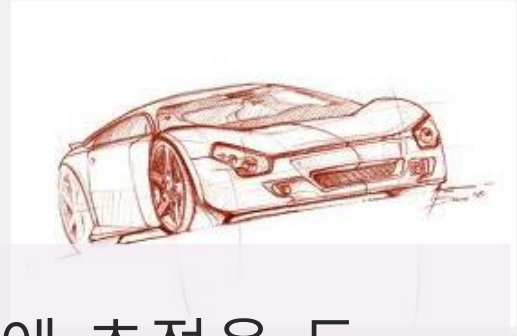
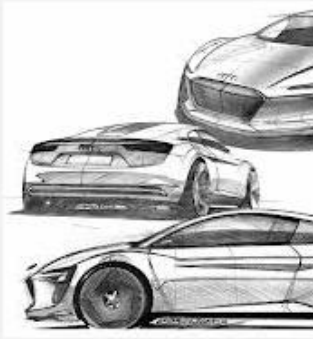
자동차 설계 과정을 비유로 들어보면..



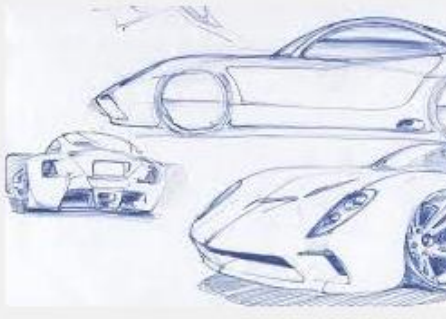




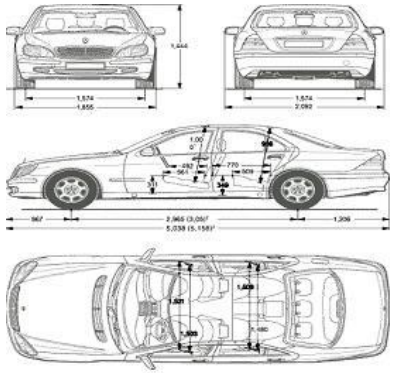


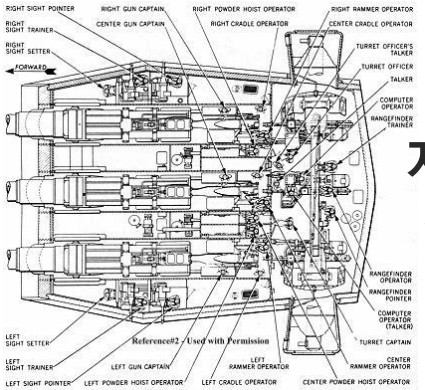


스펙과 기능에 초점을 둔
개발 스케치

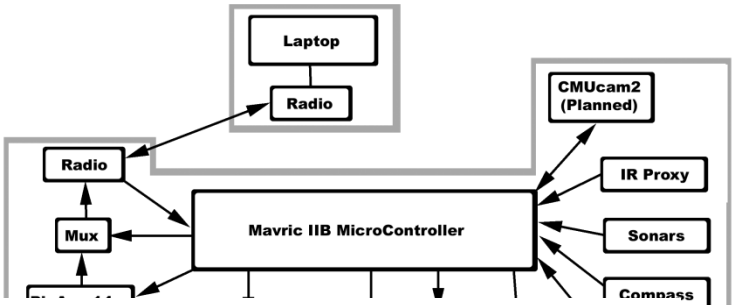
설계자의
아이디어





자동화된
설계 툴

하드웨어 설계 적용 예



설계자의 아이디어

```
input in;
output out;
output [7:0] counter;

reg [7:0] counter;

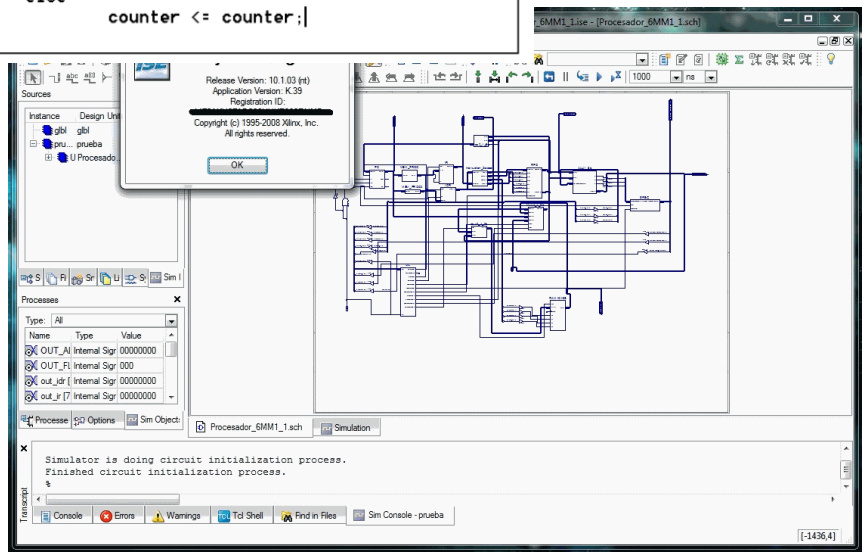
wire counter_overflow = (counter == 8'hff);
wire counter_underflow = (counter == 8'h00);

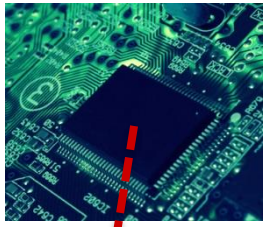
always @(posedge clk or negedge xreset)
begin
        if (xreset == 0)
                counter <= 0;
        else if (en && in && !counter_overflow)
                counter <= counter + 1;
        else if (en && !in && !counter_underflow)
                counter <= counter - 1;
        else
                counter <= counter;
end
```

컴퓨터가 이해하는 표현

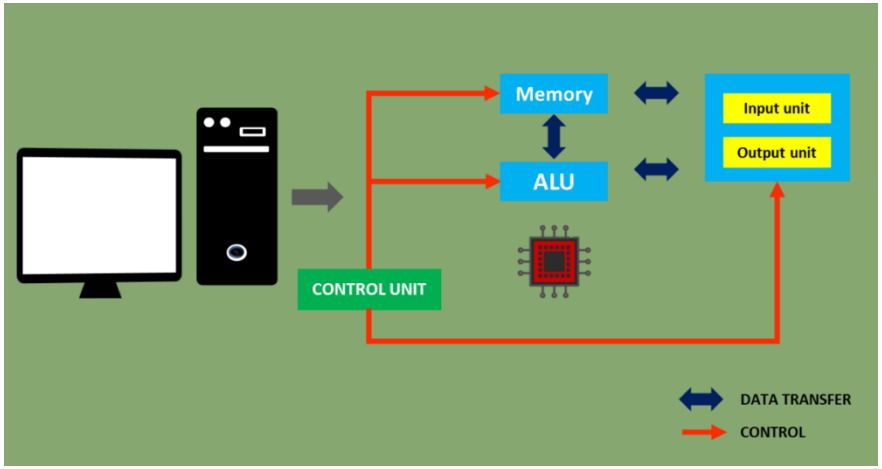
논리 회로 과목의 범위

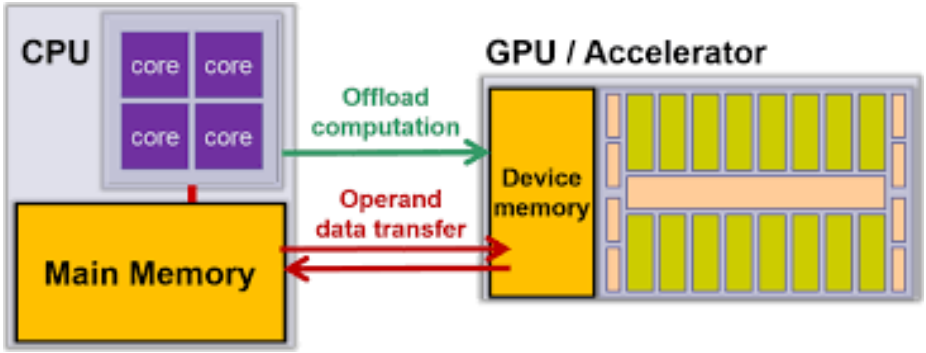
자동화된 설계 툴

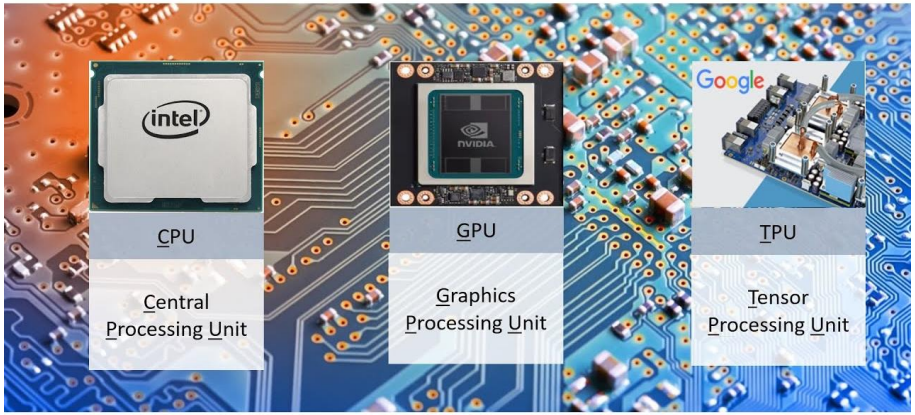




소프트웨어 설계 적용 예


Memory
ALU
Input unit
Output unit
CONTROL UNIT
DATA TRANSFER
CONTROL


CPU
core
core
core
core
Main Memory
Offload computation
Operand data transfer
GPU / Accelerator
Device memory


intel
CPU
Central Processing Unit
NVIDIA
GPU
Graphics Processing Unit
Google
TPU
Tensor Processing Unit

이 수업에서는 무엇을 배우나? – *교과서에서..*

Chapter 1 Introduction
컴퓨터에서의 수 체계 (이진수 + alpha)

Chapter 2 Combinational Systems – ex) 가위바위보
진리표로 구현하는 combinational systems
Bool 대수로 표현되는 진리표
게이트로 표현되는 진리표
입력→Combinational systems→출력

Chapter 3 The Karnaugh Map (K-map)
진리표를 map으로 표시 → 단순화, 최소화 → 최적의 시스템 만들기

Chapter 5 Designing Combinational Systems
디지털 시스템의 핵심 구성 요소가 되는 Combinational system
- adders, comparators, decoders, encoders, multiplexers..

이 수업에서는 무엇을 배우나? – *교과서에서..*

Chapter 6 Analysis of Sequential Systems – ex) 숫자세기
입력+현재 상태→Sequential systems systems→출력+다음 상태

Chapter 7 The design of Sequential Systems
디지털 시스템의 핵심 구성 요소인 Sequential system의 실례와 설계
- Counter, 패턴 인식

Chapter 8 Solving Larger Sequential Problems
Combinational + Sequential로 이루어진 통합 시스템 설계

이 수업에서는 무엇을 배우나? – *프로젝트에서..*

아직 '구현' 방법은
구체적으로
정해지지 않았음

Design concept

설계에 '방법'이 있는가?

구현 방법을 정하고
아이디어를
처음 구체화해 봄

Initial design

다시 고치는 건
늘 있는 일
Initial design보다 5배 이상의
시간과 노력이 들어감

구현 방법에 따라
시뮬레이션 방법이 달라짐
적절한 tool 선택

Simulation

Redesign

도대체 'correct'를 어떻게
검증할 것인가?
또는 어느 정도 확신으로?
80%? 90% 99%?

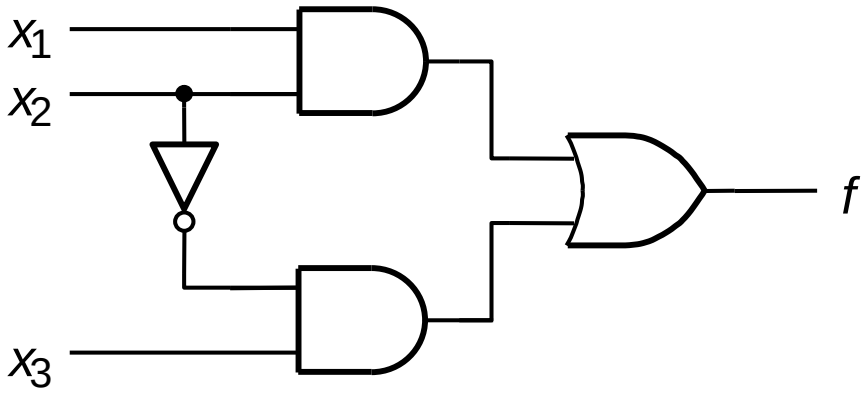
Design correct?

No

Yes

Successful design

이 수업에서는 무엇을 배우나? – *프로젝트에서..*



하드웨어 설계도
코딩할 수 있을까?

```
module example1 (x1, x2, x3, f);
        input x1, x2, x3;
        output f;

        and (g, x1, x2);
        not (k, x2);
        and (h, k, x3);
        or (f, g, h);

endmodule
```

정보통신공학 주요 역량 별 교과목 구성

연번	역량 구분	관련 교과목
1	정보통신 기초의 이해	정보통신입문
2	정보통신 관련 수학의 이해	선형대수
		확률변수론
		공업수학, 일반수학, 물리학
3	정보통신 하드웨어의 이해	회로이론
		전자회로1, 전자회로2, 전자기학
		디지털논리회로, 디지털시스템설계, 반도체소자
		디지털집적회로설계, 집적회로공학설계
		광자공학기초, 광집적회로, 광전자소자, 디스플레이공학
4	정보통신 소프트웨어의 이해	객체지향프로그래밍
		임베디드시스템설계
		시스템프로그래밍
		모바일응용소프트웨어
5	유선통신에 대한 이해	인터넷공학기초, 광통신공학설계

정보통신공학 주요 역량 별 교과목 구성

연번	역량 구분	관련 교과목
6	무선통신에 대한 이해	전파공학, 이동통신, 인터넷 프로토콜
		디지털통신시스템설계
		통신이론설계, 컴퓨터 네트워크, 정보 및 부호화이론
7	신호처리 및 멀티미디어 시스템의 이해	신호 및 시스템
		디지털신호처리설계
		디지털영상처리설계, 멀티미디어시스템
		컴퓨터그래픽스 설계
8	보안의 이해	정보 및 부호화 이론, 정보보호론
9	시스템 설계 및 분석에 대한 이해	컴퓨터구조론, 시스템프로그래밍
		오퍼레이팅시스템, 모바일응용소프트웨어설계
		알고리즘 설계, 지능정보시스템
10	데이터 분석 및 시스템에 대한 이해	자료구조, 데이터베이스설계
		실시간운영체제

주 별 수업 계획

주차	내용	교과서
1	논리회로 소개, HDL 소개	Ch.1
2	2진 숫자, 부울 대수	Ch.1-2
3	논리, 게이트	Ch.3
4	논리 최적화	Ch.4
5	combinational 회로1 - ALU	Ch.5
6	combinational 회로2 – decoder, mux 등 / Verilog 소개	Ch.5
7	Combinational logic을 위한 Verilog **중간고사**	
8	**Project1: Verilog를 이용한 combinational system 구현**	

*수업 일정은 바뀔 수 있음

주 별 수업 계획

주차	내용	교과서
9	Sequential 회로1 – latch, flip-flop	Ch.6
10	Sequential 회로2 – counter, state diagram	Ch.7
11	Sequential 회로3 – design examples, FSM	
12	Sequential logic을 위한 Verilog	
13	기말고사 **Project2: Verilog를 이용한 digital system구현**	
14	**Project2: Verilog를 이용한 digital system구현**	
15	**Project2: Verilog를 이용한 digital system구현** (팀별 중간 점검/상담)	
16	Project 완료 및 제출	

*수업 일정은 바뀔 수 있음

평가

- 중간고사 35%
- 기말고사 35%
- 퀴즈 10%

- 설계 과제 20%

- 중간고사는 기말고사보다 항상 쉽습니다.
- 퀴즈
  - Chapter 마친 그 다음 시간 수업 시작 전. 오픈 북.
- 설계 과제1
  - 1인 1과제 제출, 모두 동일한 문제
- 설계 과제2
  - 3~4인 1팀, 무작위로 서로 다른 문제 뽑기
  - 과제 진행 상담 잘 활용하기

제 수업에서는..

- 수업 시작: 지각/결석 하지 맙시다.
  - 1/4 이상 결석 시 무조건 F
  - 단, 특별한 일이 있을 경우 하루 전에 이메일 보낼 것

- 성적
  - 시험/프로젝트 채점 결과를 지정된 기간에 반드시 확인
  - 최종 성적이 결정된 이후에 정정 요구 절대 불가
  - (본인의 상담을 마치지 못하고) 성적 확인을 못한다고 개인적으로 성적 문의하지 말 것

- 시험 및 프로젝트 부정 행위.
  - 시험에서 부정 행위 발견 (시험 중/시험 후) – **무조건 F** 학점
  - 설계 과제 부정 행위– 베낀 사람/보여준 사람 모두 0점 처리, 온라인 코드 참고 포함

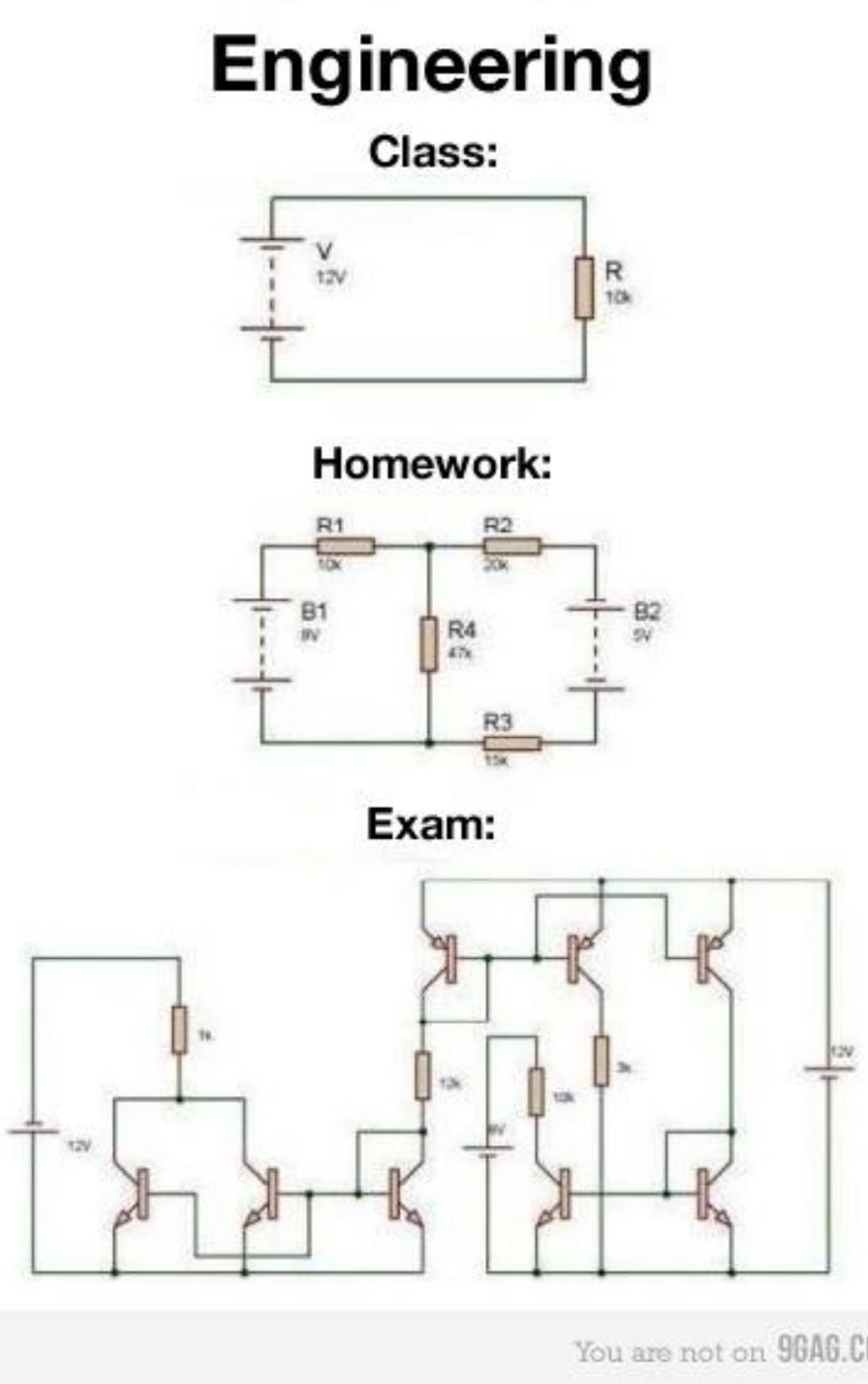
교수님이 대학교 2학년이었을 때..

- 기본적으로 논리 회로/설계는 아주 쉽고 재미있습니다!
- 내가 2학년 때 왜 그렇게 어려웠나?
  - 정답: 수업을 빼먹어서, 교과서를 안 읽어서
- 수업만 나오면 된다!
- 그럼에도 불구하고 수업 시간에,
  - "아주 쉬워요", "어려운 거 하나도 없어요"라고 말하면

웰컴 투 공대 시험

완전 사실임 →

어떡하죠?
연습문제를 푸세요!



표절 방지

- "**표절(plagiarism)은 라틴어로 '어린아이 납치범'**을 의미한다. 남의 견해를 무단 인용하는 것은 다른 사람의 '정신적인 아이'를 훔치는 것이라고 볼 수 있다." - Writing with Sources- A guide for Students, Harvard University
- "이 책에 따르면 **'표절은 절박함에서 비롯되는 경우'가 많다**. 처음엔 '참조'만 하려는 좋은 의도로 남의 저작물을 보다가 시간에 쫓기면 절박한 심정이 되어 과제를 쉽게 해결하려고 도덕적 경계선을 넘는다는 것"
- **공동 프로젝트는 각자가 기여한 부분을 정확히 명시해야** '부적절한 공조' 위험에서 벗어날 수 있다. 한국 대학생들은 동료에게 자신의 과제물을 대수롭지 않게 빌려 주고 있지만 이러한 행동은 **표절 방조 행위**라고 지적하고 있다.

- **하버드대는 표절 예방 노하우**로
  - 마감 직전 급하게 논문 쓰는 일을 피하기
  - 메모할 때 내 생각과 다른 사람의 견해를 구분하기
  - 실제보다 더 잘 아는 것처럼 보이려 들지 말기
  - 시간 안에 숙제를 제출하기 힘들면 교수와 상의하기
  - 컴퓨터 작업 때는 백업 사본을 항상 만들기

동아일보 2007년 2월 20일자 보도 참조

반도체
(circuits & systems)
종류들

다양한 반도체 칩들

- 메모리: 데이터 기억
  - DRAM, SRAM, Flash 등
- MCU (Micro Controller Unit)
- MPU (Micro Processor Unit)
- DSP (Digital Signal Processor)

경계가 모호해지고 있음

- DAC(Digital to Analog Converter) and ADC

- Mp3, mpeg (mpeg2, mpeg4, h.264..) decoder and encoder

다양한 반도체 칩들의 분류

아날로그 반도체	디지털 반도체
	메모리 반도체
ADC	MCU
DAC	DSP
ASIC 칩	MP3 코덱
	MPEG 코덱
	ASIC 칩

표 5.2 아날로그 반도체와 디지털 반도체의 예

다양한 반도체 칩들의 분류

범용 반도체	ASSP/ASIC 반도체
메모리 반도체	
MCU	MP3 코덱
DSP	MPEG 코덱
ADC	ASIC 칩
DAC	

표 5.3 범용 반도체와 ASSP/ASIC 반도체의 예

Application Specific Integrated Circuit(ASIC)

Application Specific Standard Product(ASSP)

다양한 반도체 칩들의 분류

메모리 반도체	시스템 IC
	MCU
	DSP
	ADC
메모리 반도체	DAC
	MP3 코덱
	MPEG 코덱
Dram, sram, flash, rom, eprom eeprom..	ASIC 칩

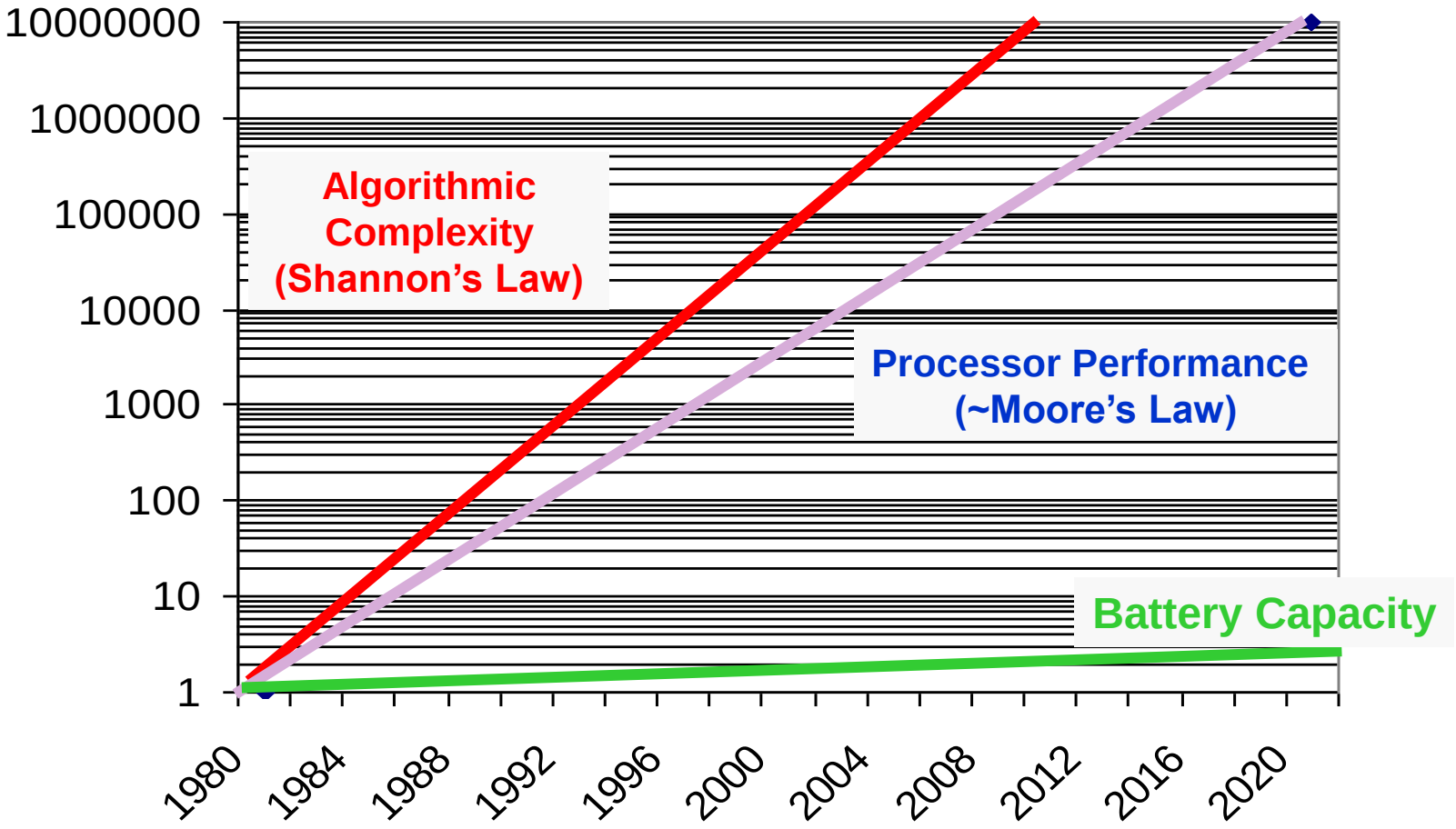
표 5.4 메모리 반도체와 시스템 IC의 예

우리나라는 반도체 최강국?

→우리나라는 메모리 반도체의 최강국

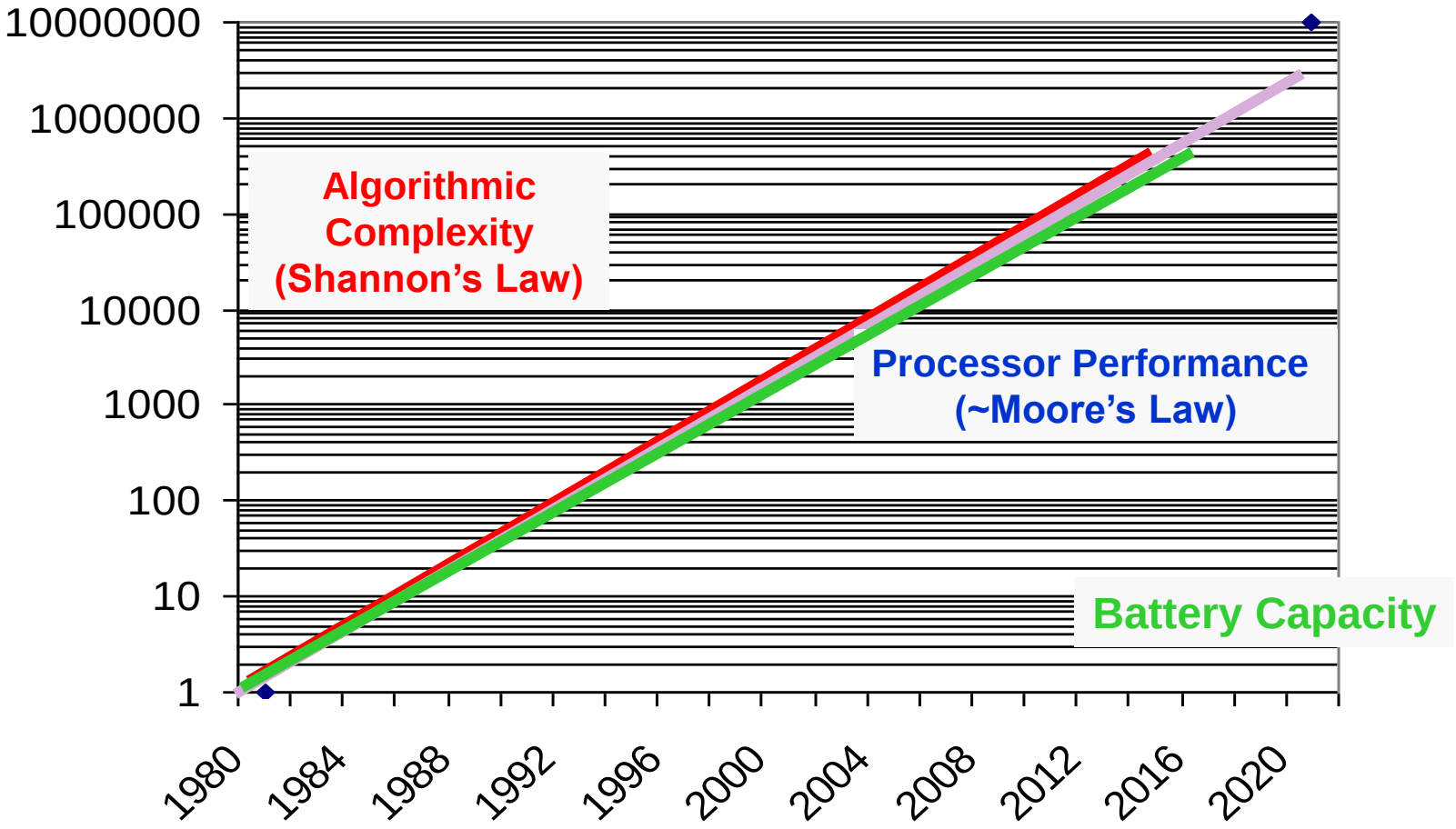
좋은 디지털 시스템 설계 의미

Technology Trend



Source: Data compiled from multiple sources (BWRC)

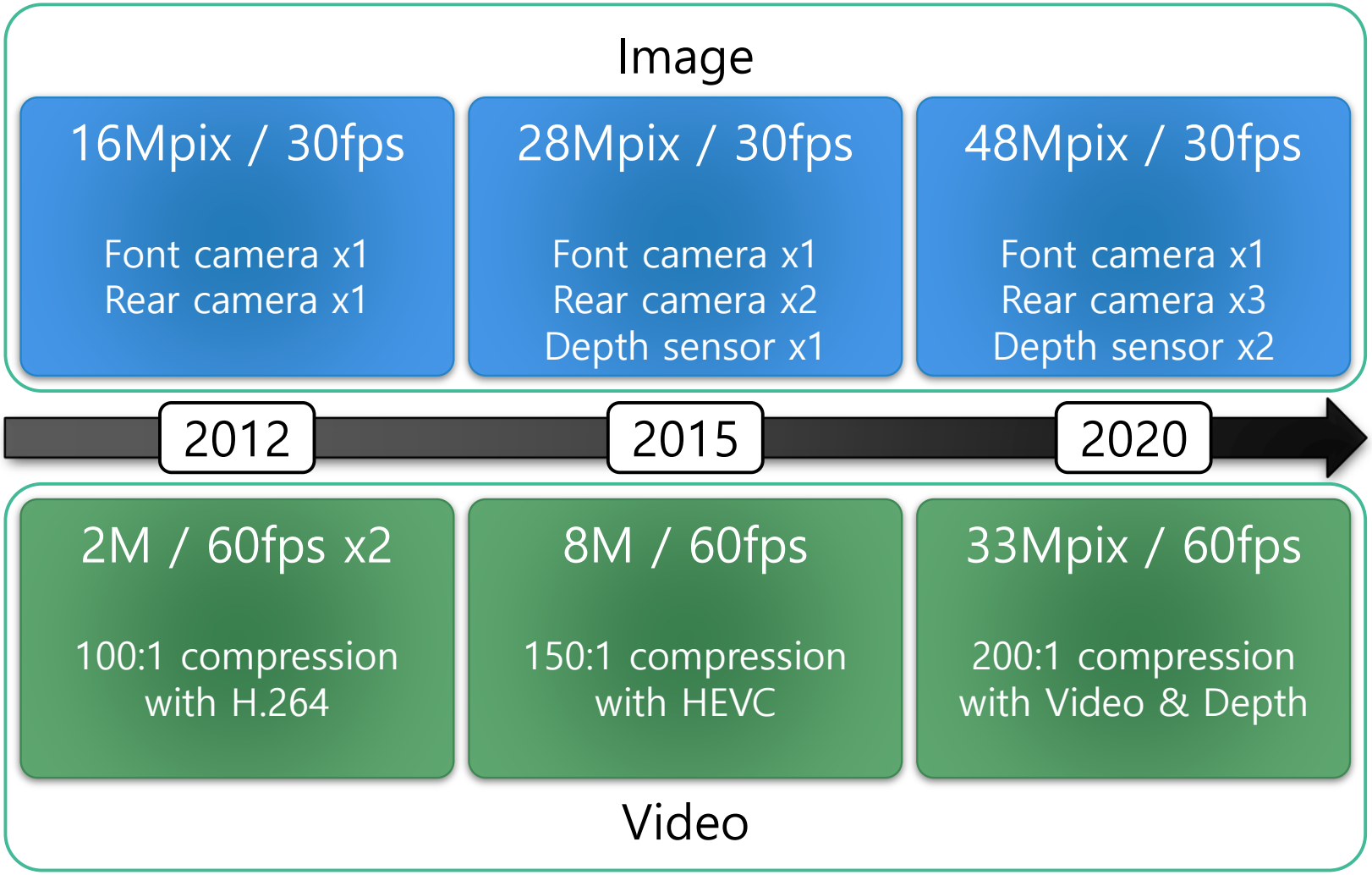
마법과 같은 일이 가능할까?



Source: Data compiled from multiple sources (BWRC)

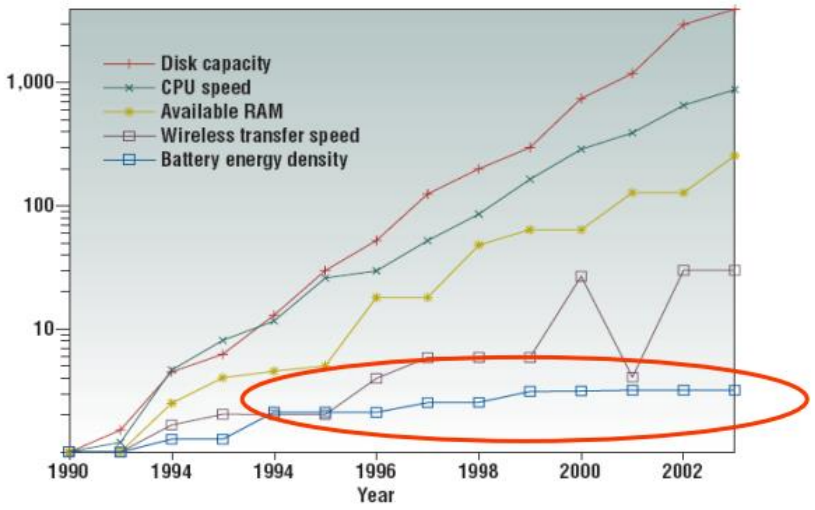
성능

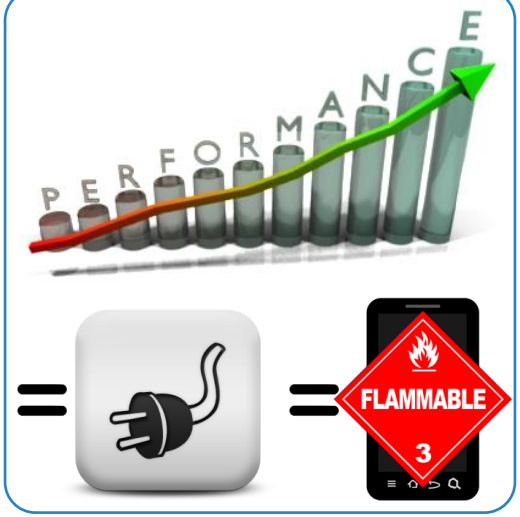
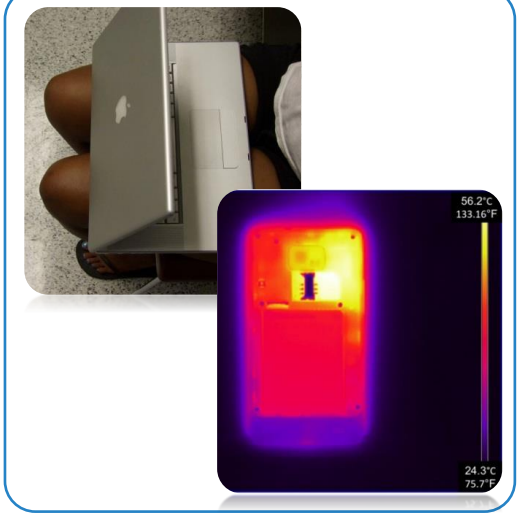
더 크게 크게 & 더 빠르게



에너지

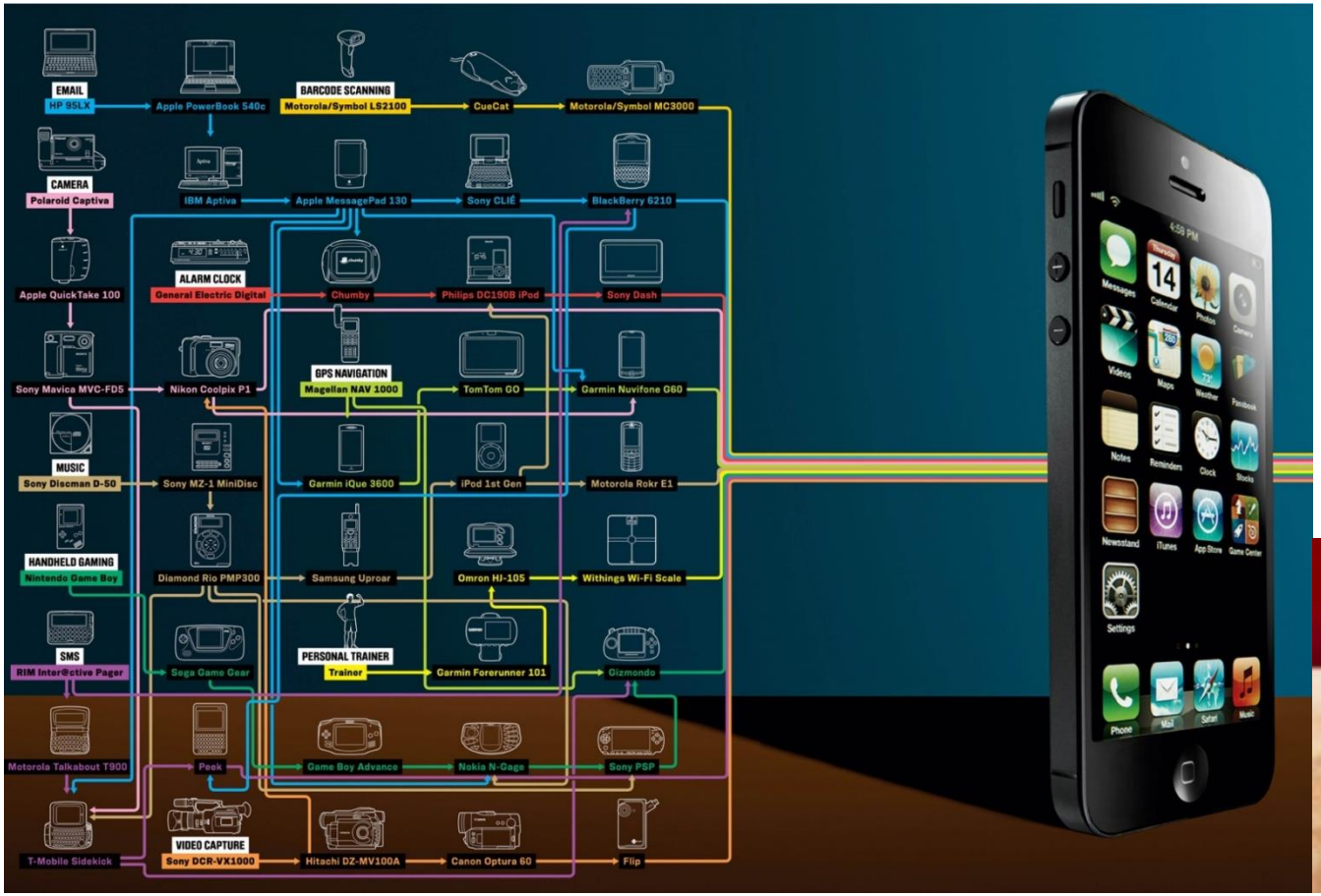
더 오래 & 덜 뜨겁게



High power consumption	Limited cooling capacity	Human sensitivity
		

크기

더 작게 작게 작게

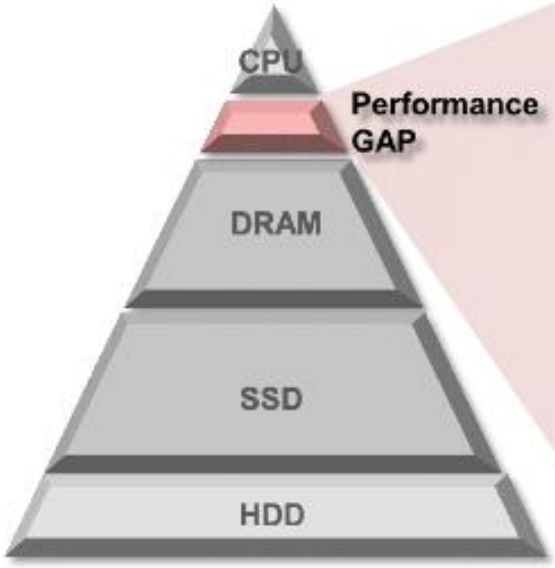

EMAIL
HP 95LX
Apple PowerBook 540c
BARCODE SCANNING
Motorola/Symbol LS2100
CueCat
Motorola/Symbol MC3000
CAMERA
Polaroid Captiva
IBM Aptiva
Apple MessagePad 130
Sony CLIÉ
BlackBerry 6210
Apple QuickTake 100
ALARM CLOCK
General Electric Digital
Chumby
Philips DC190B iPod
Sony Dash
Sony Mavica MVC-FD5
Nikon Coolpix P1
GPS NAVIGATION
Magellan NAV 1000
TomTom GO
Garmin Nuvifone G60
MUSIC
Sony Discman D-50
Sony MZ-1 MiniDisc
Garmin iQue 3600
iPod 1st Gen
Motorola Rokr E1
HANDHELD GAMING
Nintendo Game Boy
Diamond Rio PMP300
Samsung Uproar
Omron HJ-105
Withings Wi-Fi Scale
SMS
RIM Interactive Pager
Sega Game Gear
PERSONAL TRAINER
Trainer
Garmin Forerunner 101
Gizmondo
Motorola Talkabout T900
Peek
Game Boy Advance
Nokia N-Gage
Sony PSP
T-Mobile Sidekick
VIDEO CAPTURE
Sony DCR-VX1000
Hitachi DZ-MV100A
Canon Optura 60
Flip

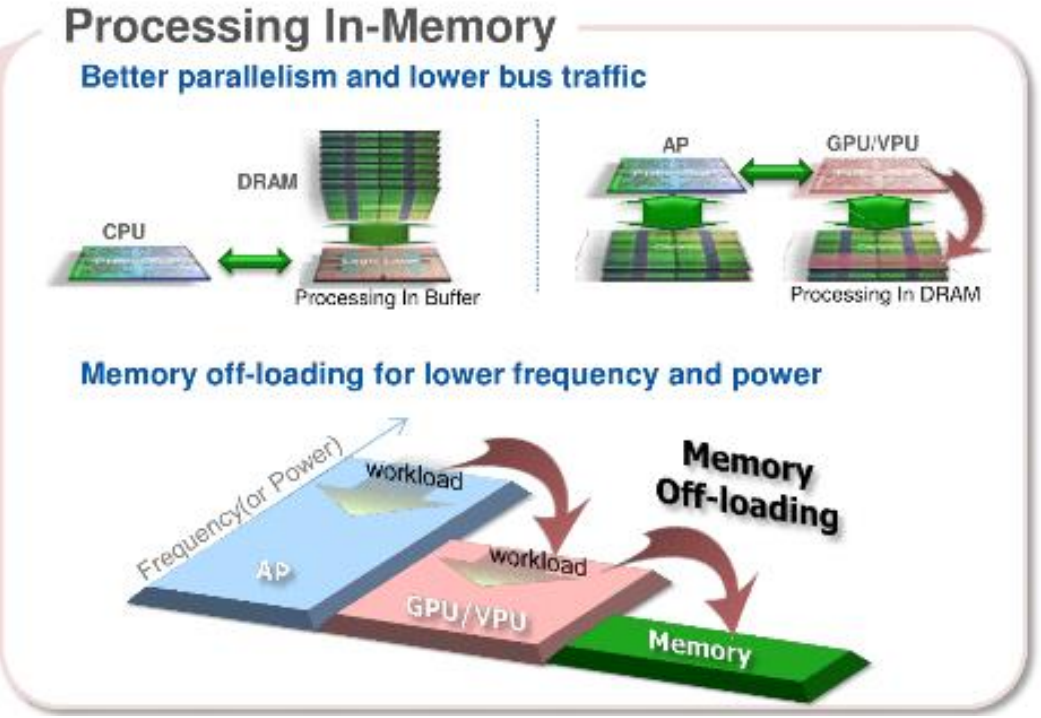
빠르게, 에너지 효율적으로

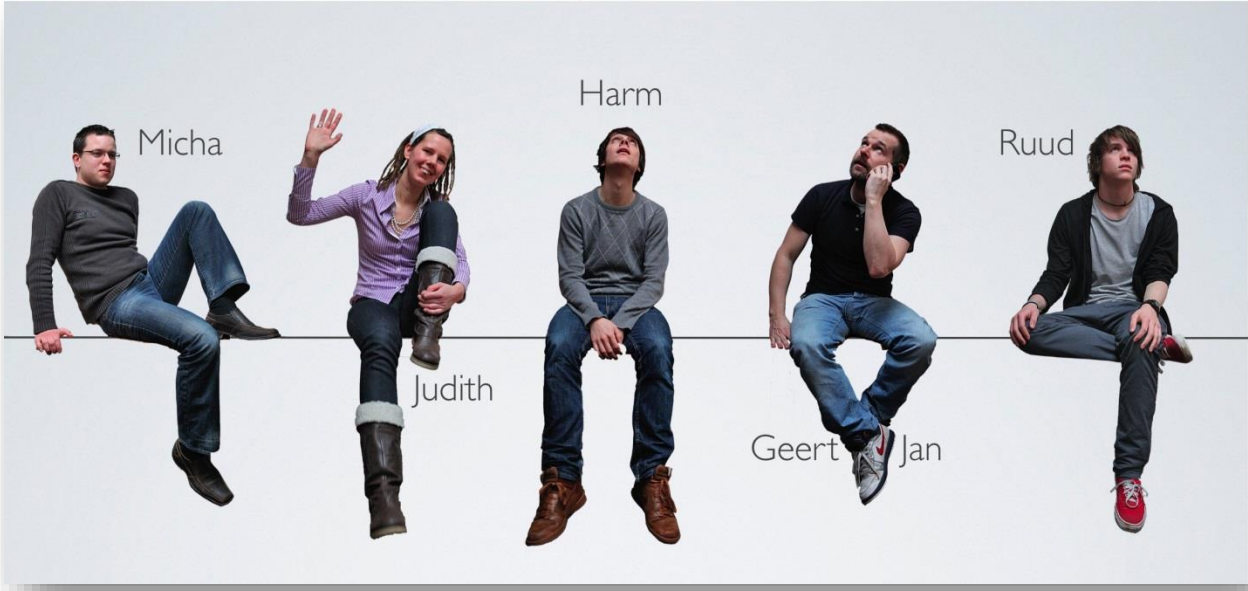
메모리 가까이!

Processing In Memory: PIM

- Fill the performance gap and deliver energy-efficient solutions



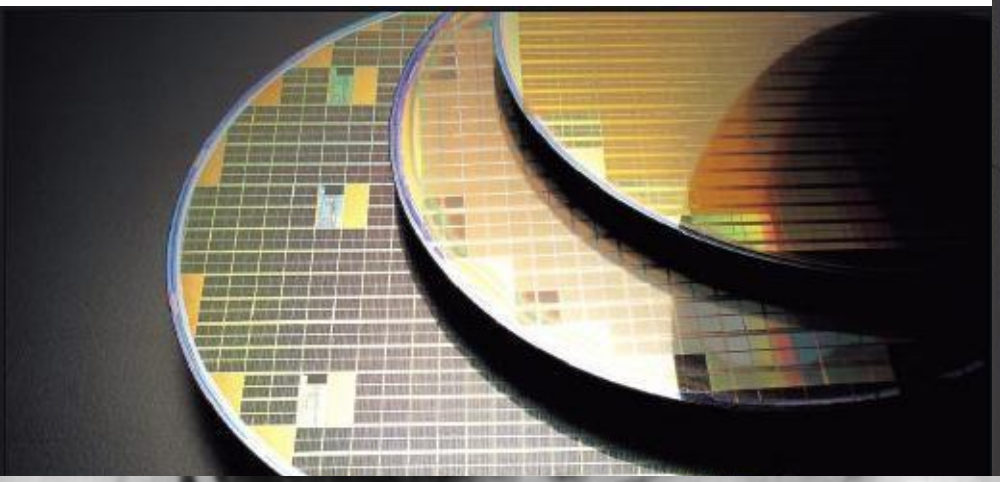


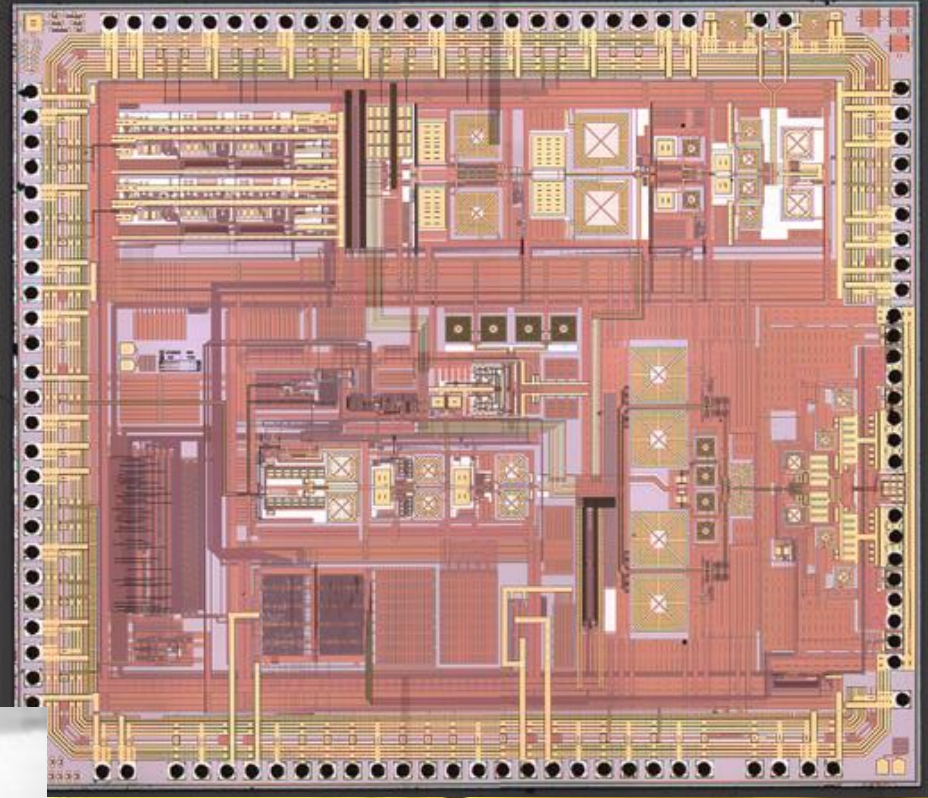


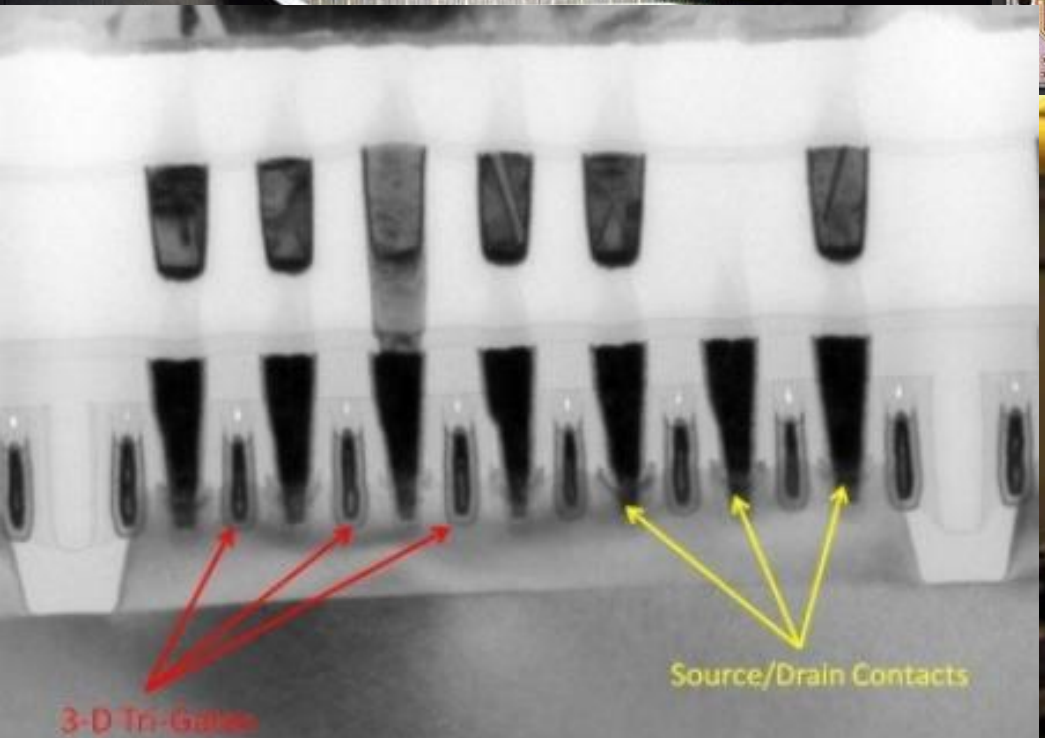
그리고, 사람에 대한 이해

설계 및 제조 단계

반도체 설계?

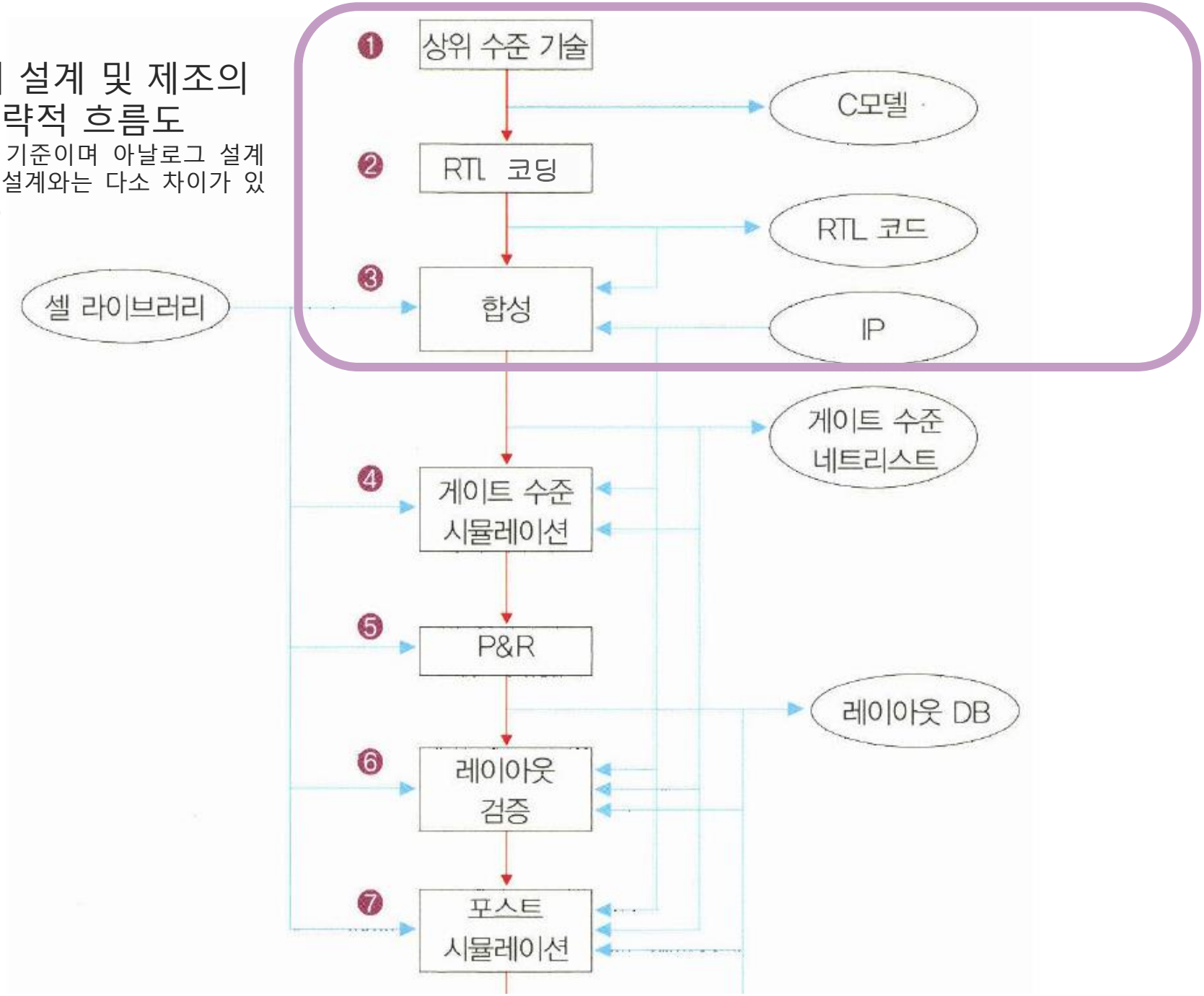






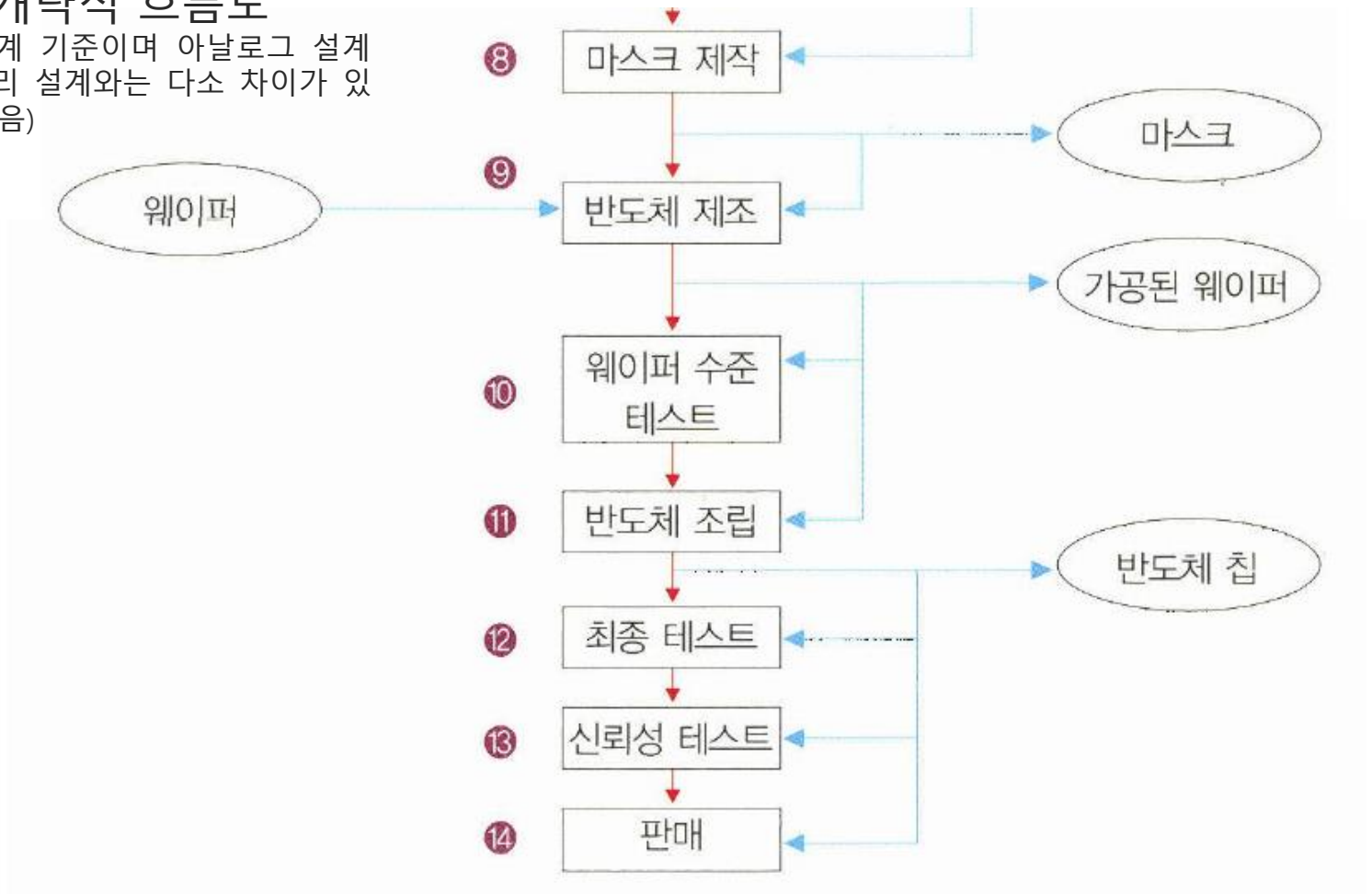
반도체 설계 및 제조의 개략적 흐름도

(SoC 설계 기준이며 아날로그 설계나 메모리 설계와는 다소 차이가 있을 수 있음)



반도체 설계 및 제조의 개략적 흐름도

(SoC 설계 기준이며 아날로그 설계나 메모리 설계와는 다소 차이가 있을 수 있음)



1. 상위수준 기술 (High level description)

- 컴퓨터 언어상의 상위언어(high level language)로 기술
  ex) C 언어

- C Model
  - C 언어로 시스템 설계의 'function'에 초점을 맞추어 작성된 프로그램

예 1.1.1

```
S=(A+B);
```

- C model이 왜 필요??

2. RTL coding & simulation (High level description)

- RTL Coding(Register Transfer Level) coding
  - HDL(Hardware Description Language)로 기술
  - Ex) VHDL, Verilog
  - 설계=코딩&Simulation

예 1.1.1

S=(A+B);

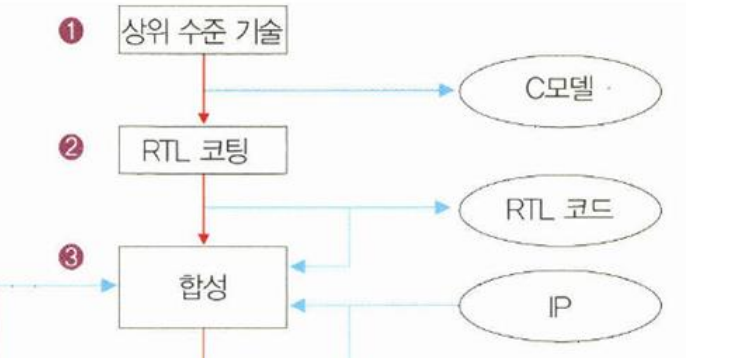
C → RTL

```
1  module  sum (S, A, B, clk);
2
3  output[5:0]          S;
4  input[4:0]           A, B;
5  input                clk;
6
7  reg[5:0]             S;
8
9        always @ (posedge clk) S <= (A+B);
10
11 endmodule
```

입력과 출력의 정확한 bit width

덧셈이 1cycle안에 끝남

예 1.2.1 verilog HDL로 기술한 RTL 코드의 예



3. 합성 (Synthesis)

- 합성 툴(Synthesis tool)을 이용하여 RTL 코드를 gate 수준의 netlist (gate level netlist)로 바꾸어 주는 과정

```
1  module  sum (S, A, B, clk);
2
3  output[5:0]          S;
4  input[4:0]           A, B;
5  input                clk;
6
7  reg[5:0]             S;
8
9        always @ (posedge clk) S <= (A+B);
10
11 endmodule
```

예 1.2.1 verilog HDL로 기술한 RTL 코드의 예

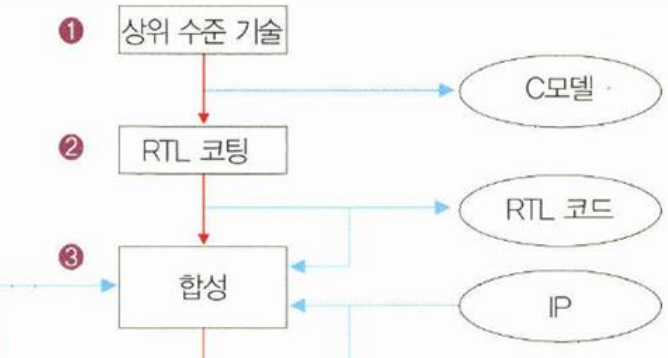
RTL → netlist

```
1  module  sum (S, A, B, clk);
2
3  output[5:0]     S;
4  input[4:0]      A, B;
5  input           clk;
6
7  wire[4:0]       pS, CO;
8
9  supply0 gnd;
10
11 FA d0 (pS[0], CO[0], A[0], B[0], gnd);
12 FA d1 (pS[1], CO[1], A[1], B[1], CO[0]);
13 FA d2 (pS[2], CO[2], A[2], B[2], CO[1]);
14 FA d3 (pS[3], CO[3], A[3], B[3], CO[2]);
15 FA d4 (pS[4], CO[4], A[4], B[4], CO[3]);
16
17 DFF i0 (S[0], pS[0], clk);
18 DFF i1 (S[1], pS[1], clk);
19 DFF i2 (S[2], pS[2], clk);
20 DFF i3 (S[3], pS[3], clk);
21 DFF i4 (S[4], pS[4], clk);
22 DFF i5 (S[5], CO[4], clk);
23
24 endmodule
```

Full adder

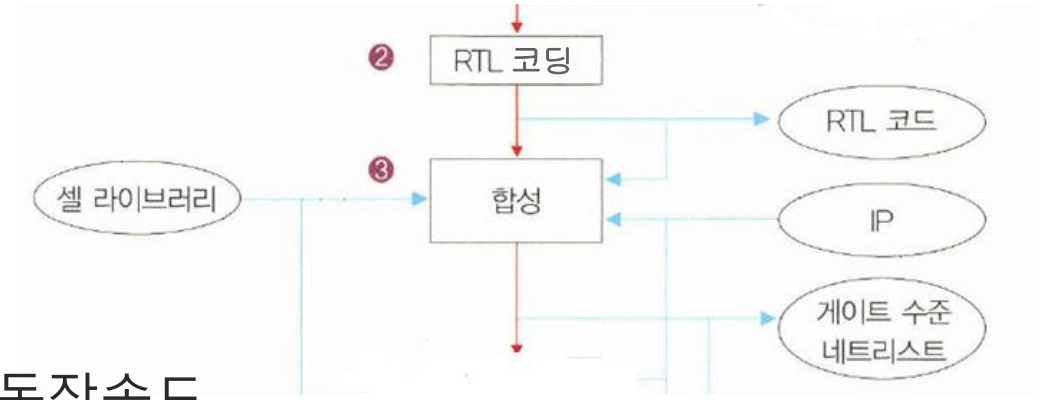
FlipFlop

Standard Cell



3. 합성 (Synthesis)

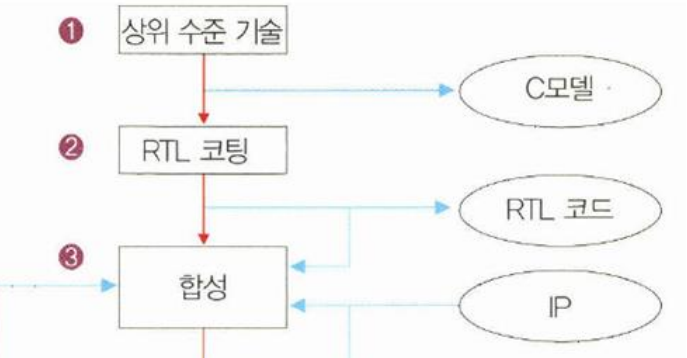
- 합성 툴(Synthesis tool)을 이용하여 RTL 코드를 gate 수준의 netlist (gate level netlist)로 바꾸어 주는 과정
  - RTL 코드를 이해해서
  - 동일한 기능을 하는 cell을 cell library에서 가져와서
  - Gate level netlist 작성



- Cell library
  - Cell의 기능, 크기(area), 동작속도,
    주어진 동작을 하는데 걸리는 시간(propagation delay time),
    동작에 소요되는 전력

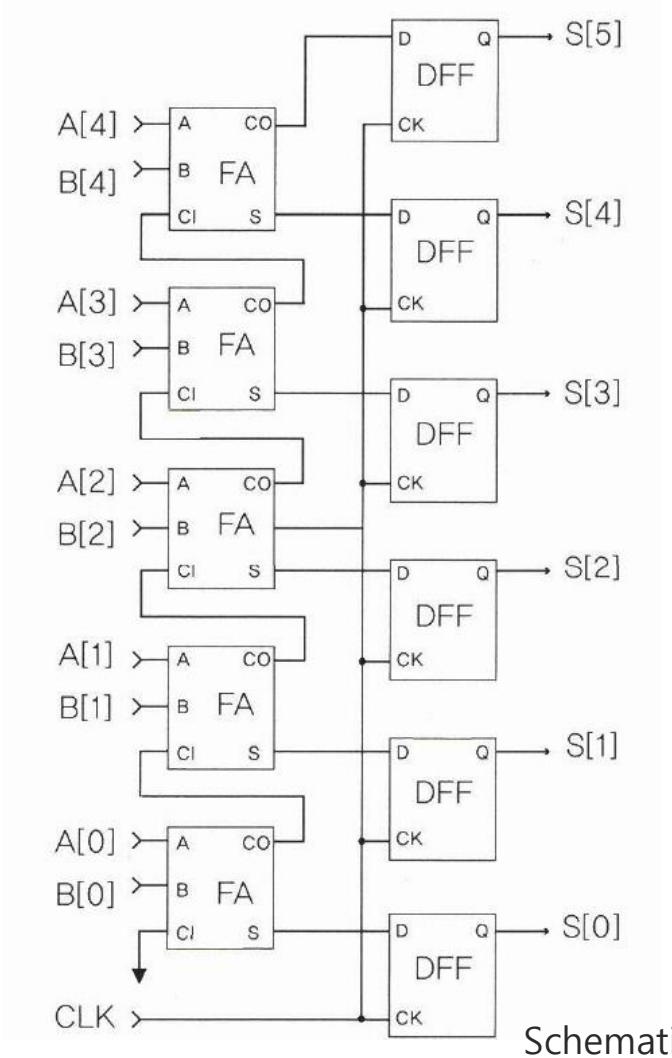
3. 합성 (Synthesis)

```
1  module  sum (S, A, B, clk);
2
3  output[5:0]     S;
4  input[4:0]      A, B;
5  input           clk;
6
7  wire[4:0]       pS, CO;
8
9  supply0 gnd;
10
11 FA d0 (pS[0], CO[0], A[0], B[0], gnd);
12 FA d1 (pS[1], CO[1], A[1], B[1], CO[0]);
13 FA d2 (pS[2], CO[2], A[2], B[2], CO[1]);
14 FA d3 (pS[3], CO[3], A[3], B[3], CO[2]);
15 FA d4 (pS[4], CO[4], A[4], B[4], CO[3]);
16
17 DFF i0 (S[0], pS[0], clk);
18 DFF i1 (S[1], pS[1], clk);
19 DFF i2 (S[2], pS[2], clk);
20 DFF i3 (S[3], pS[3], clk);
21 DFF i4 (S[4], pS[4], clk);
22 DFF i5 (S[5], CO[4], clk);
23
24 endmodule
```



합성툴의 발전으로 일일이
schematic을 그리지 않고 RTL로
복잡한 설계의 생산성 향상

=

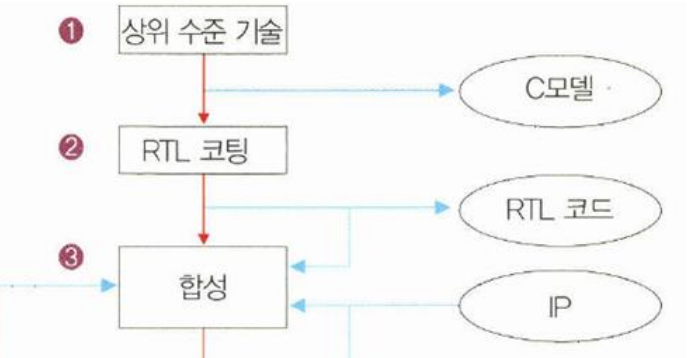


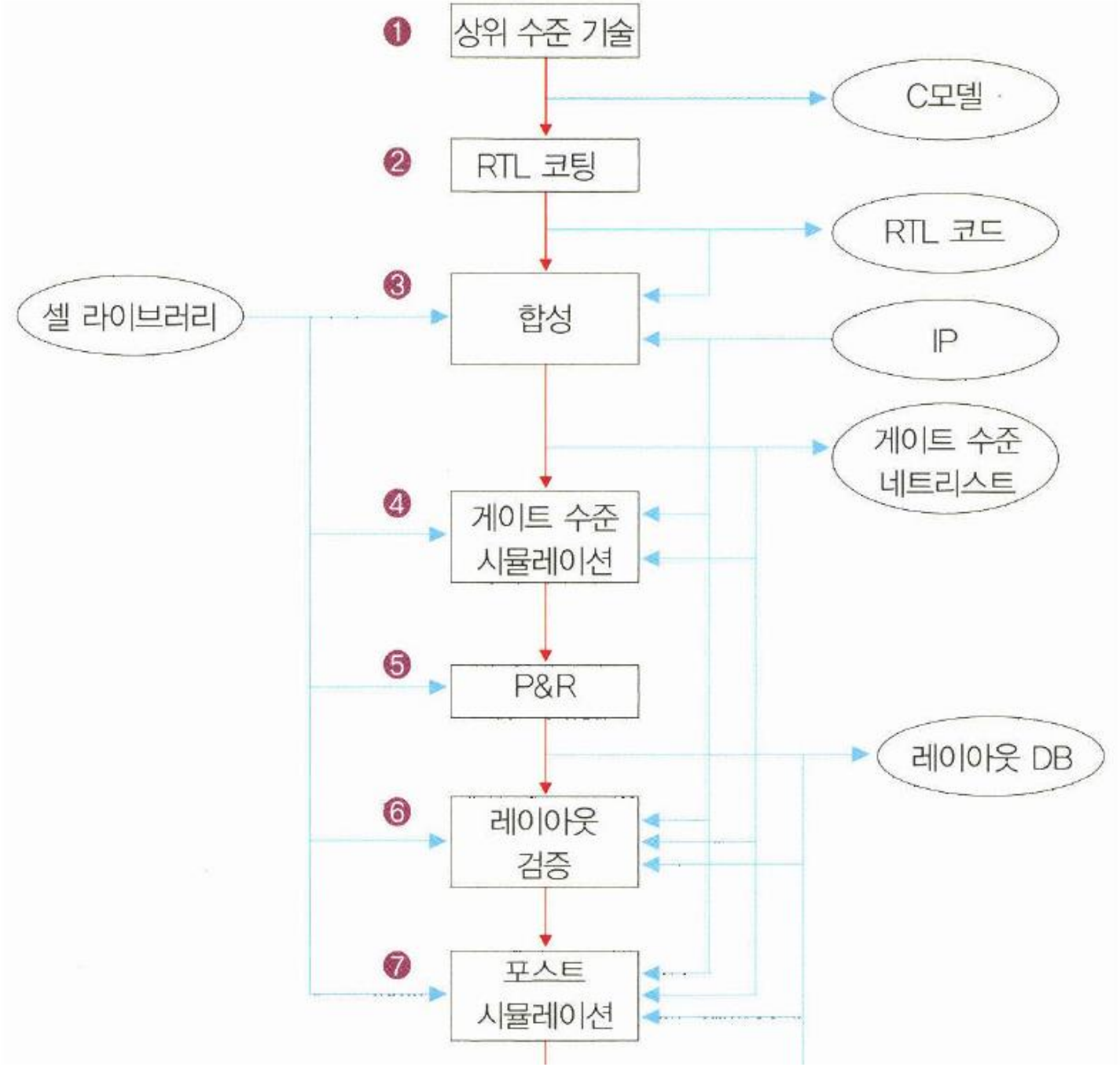
3. 합성 (Synthesis)

- 1) 설계 시간 단축, 생산성 향상

- 2) 상위수준 RTL 언어로 설계
  - SW 프로그래밍이 가능한 엔지니어들도
    (전자회로에 깊은 지식이 없어도) 반도체 설계가 가능

바라건데..
희망컨데..




1
상위 수준 기술
C모델
2
RTL 코팅
RTL 코드
3
셀 라이브러리
합성
IP
게이트 수준
네트리스트
4
게이트 수준
시뮬레이션
5
P&R
레이아웃 DB
6
레이아웃
검증
7
포스트
시뮬레이션

4. 게이트 수준 시뮬레이션

- Gate-level Simulation
  - gate-level netlist에 여러 가지 입력 신호를 넣어 시뮬레이션

▪ 시뮬레이션의 정확도 비교

예 1.1.1

S=(A+B);

<

```
1  module  sum (S, A, B, clk);
2
3  output[5:0]          S;
4  input[4:0]           A, B;
5  input                clk;
6
7  reg[5:0]             S;
8
9        always @ (posedge clk) S <= (A+B);
10
11 endmodule
```

정확한 bit 와
수행 cycle 표현 가능

<

```
1  module  sum (S, A, B, clk);
2
3  output[5:0]    S;
4  input[4:0]     A, B;
5  input          clk;
6
7  wire[4:0]      pS, CO;
8
9  supply0 gnd;
10
11 FA d0 (pS[0], CO[0], A[0], B[0], gnd);
12 FA d1 (pS[1], CO[1], A[1], B[1], CO[0]);
13 FA d2 (pS[2], CO[2], A[2], B[2], CO[1]);
14 FA d3 (pS[3], CO[3], A[3], B[3], CO[2]);
15 FA d4 (pS[4], CO[4], A[4], B[4], CO[3]);
16
17 DFF i0 (S[0], pS[0], clk);
18 DFF i1 (S[1], pS[1], clk);
19 DFF i2 (S[2], pS[2], clk);
20 DFF i3 (S[3], pS[3], clk);
21 DFF i4 (S[4], pS[4], clk);
22 DFF i5 (S[5], CO[4], clk);
23
24 endmodule
```

Cell의 종류와 개수,각 cell의 크기와 동작 속도
→ 회로의 cell area계산 가능
→ 동작 가능한 clock frequency 계산 가능

5. P & R

- Place and Routing (P & R)
  - Cell을 배치(place)하고 연결(routing)시키는 작업

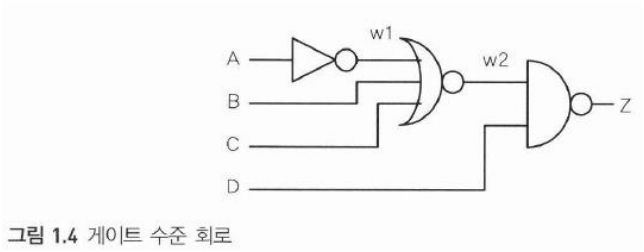


그림 1.4 게이트 수준 회로

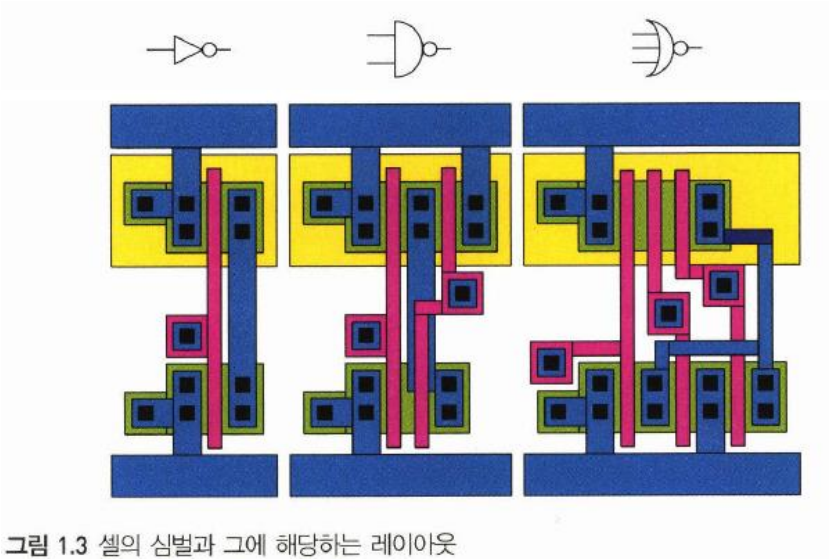


그림 1.3 셀의 심벌과 그에 해당하는 레이아웃

Cell library에
정보 있음

5. P & R

- Place and Routing (P & R)
  - Cell을 배치(place)하고 연결(routing)시키는 작업

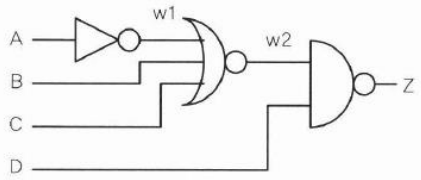


그림 1.4 게이트 수준 회로

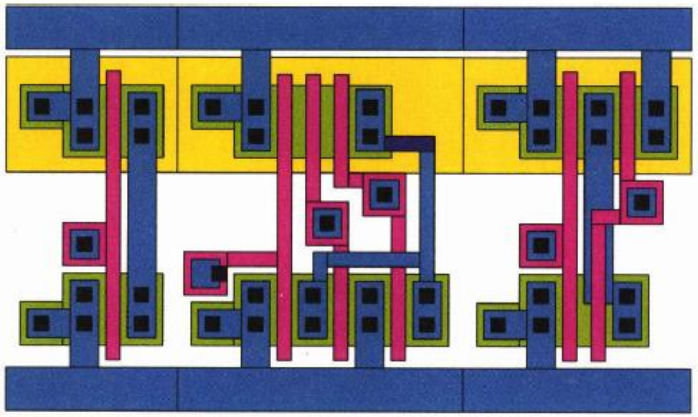


그림 1.5 플레이스먼트(placement)의 예

Netlist를 보고 서로 연결될 cell들을 이웃하게 배치

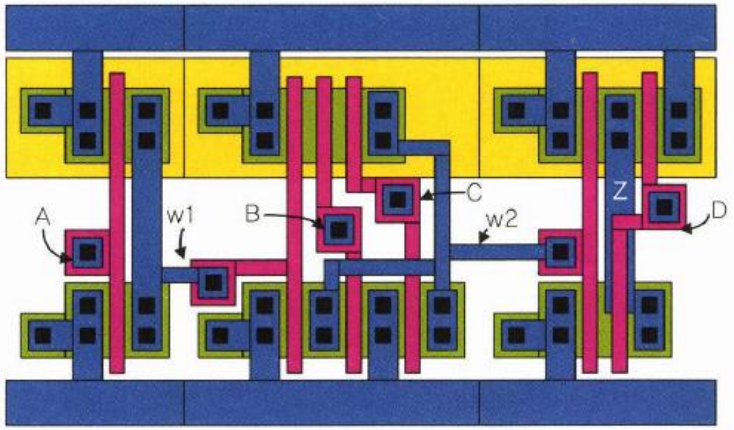


그림 1.6 라우팅(routing)의 예

Netlist를 보고 cell들을 연결

6. 마스크 제작

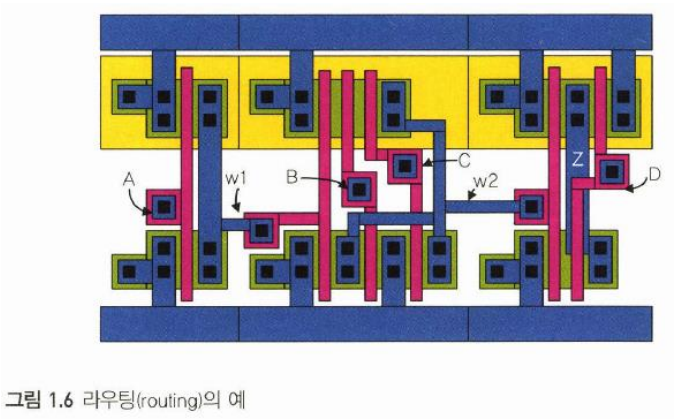
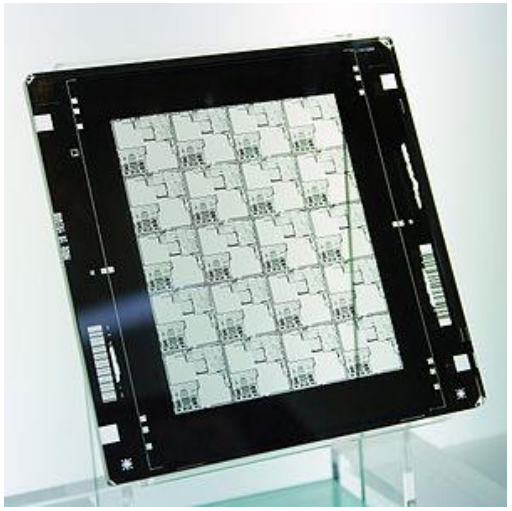


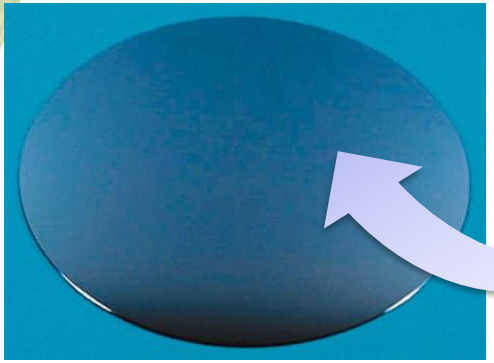
그림 1.6 라우팅(routing)의 예

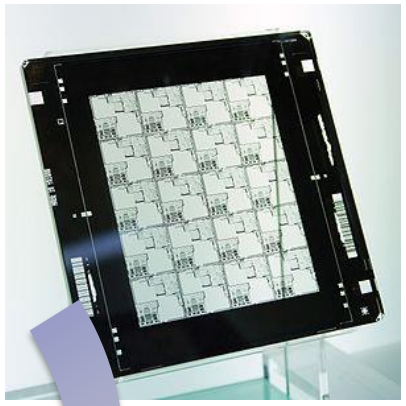


- 마스크(mask) or 레티클(reticle)
  - Layout DB를 실리콘 위에 찍어낼 사진의 필름

7. 반도체 제조

Bare Wafer





프린트 (사진 인화와 비슷)
Fabrication (FAB)
8주~12주

8. 반도체 조립

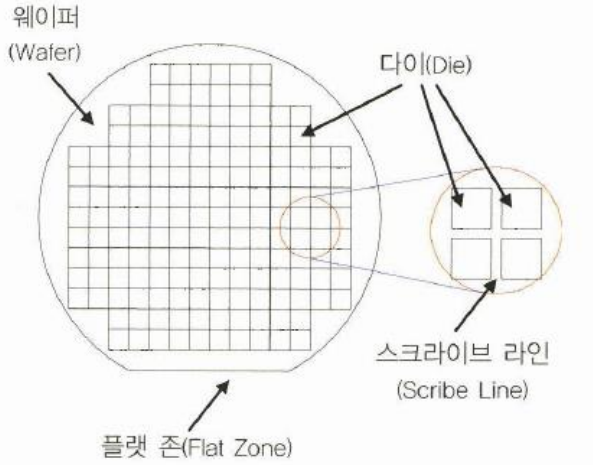


그림 1.8 가공된 실리콘 웨이퍼

- Scribe line을 톱으로 잘라내서(sawing) 각 die들을 분리 후
- 반도체 조립(assembly, ASS'Y, packaging)

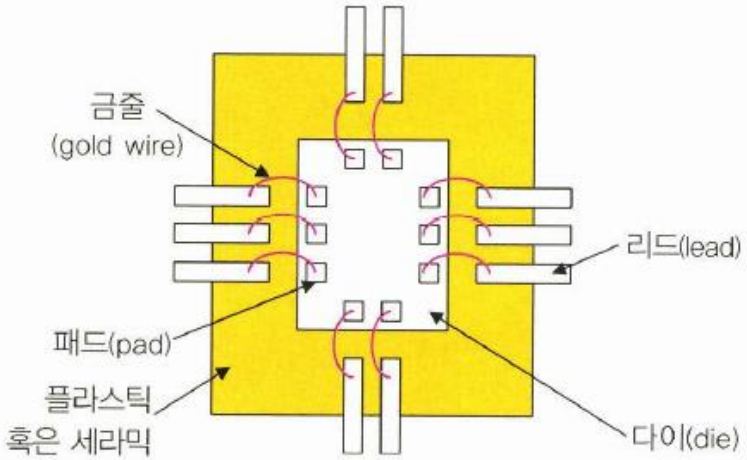
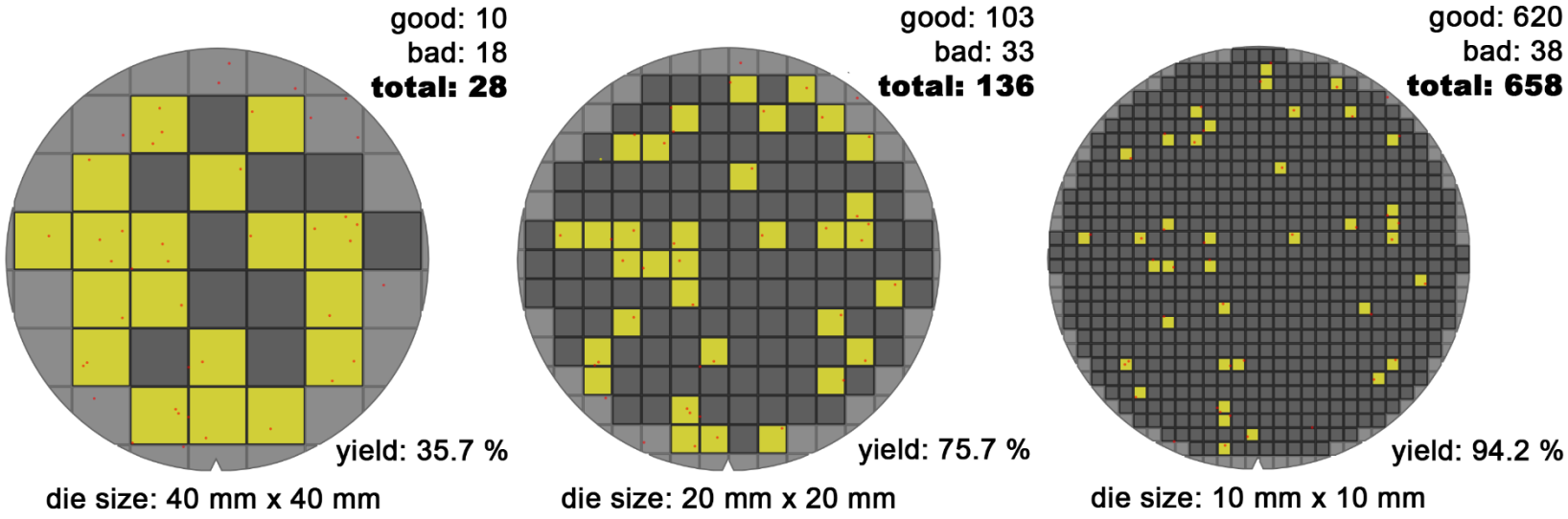


그림 1.10 반도체 조립

Die size와 수율(yield)의 관계



8. 반도체 조립

- 조립 후 모습

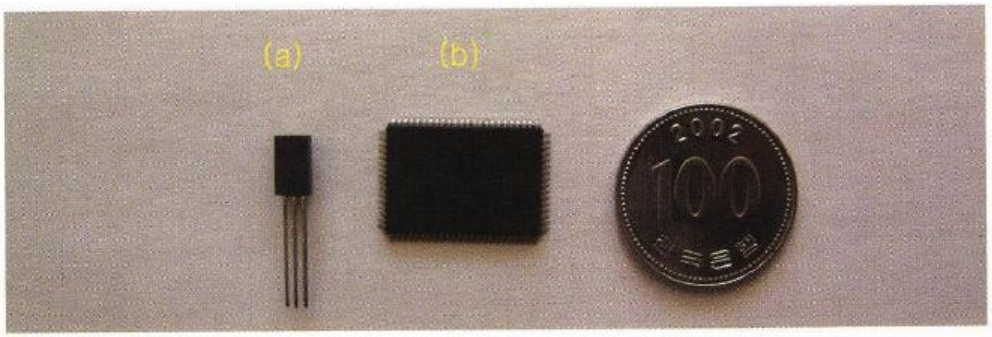


그림 1.11 조립이 완성된 반도체 칩: (a)는 트랜지스터, (b)는 반도체 IC 칩

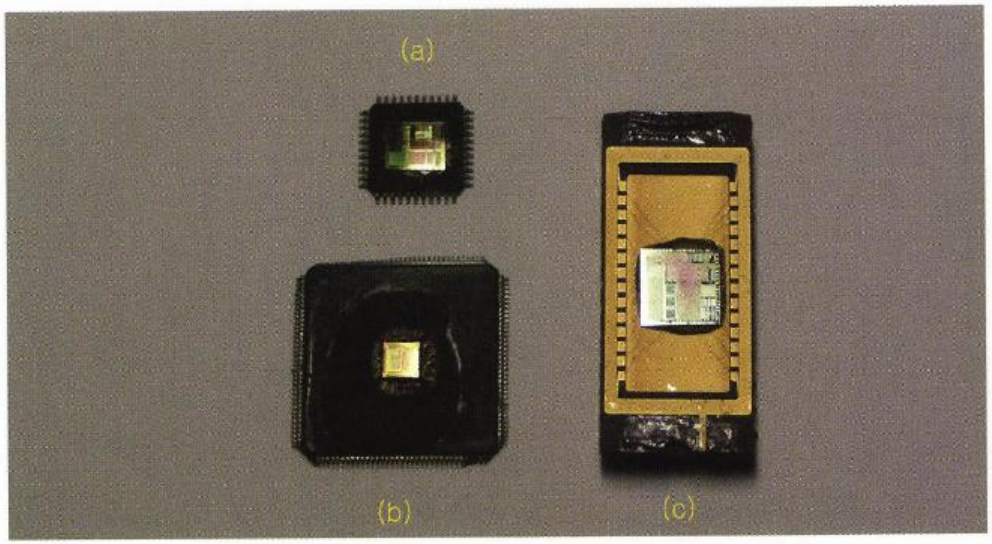


그림 1.12 반도체 칩 내부의 다이들: (a), (b)는 플라스틱 패키지 내부의 다이, (c)는 세라믹 패키지 내부의 다이

9. 판매

누가 가격을 결정할까?

수요자 시장?

메모리와 같이 기능이 정해진 칩은
호환 가능하므로 생산량에 따라
수요자 중심 시장

SK hynix
Micron
SAMSUNG

공급자 시장?

개발 기간이 오래 걸리고
칩마다 기능들이 다른 대부분의
반도체들 (특히 System IC)들은
공급자 중심 시장

QUALCOMM
ARM
intel
AMD
NVIDIA
MEDIATEK
SAMSUNG

HDL (Hardware description language) 을 이용한 설계

설계 사양 결정

- 회로의 기능, 동작성능, 동작 주파수, 칩 면적 및 전력 소모 목표치, 시험 범주 (test coverage), 설계 기간, 칩 단가 등이 포함된 설계 목표

- 설계될 시스템의 분할, 적용될 알고리즘과 아키텍처, 데이터 입/출력 및 제어 신호들의 타이밍, 입력/출력 신호의 이름과 비트 폭, 리셋 및 enable 신호, 클록 신호에 대한 정의 등을 포함

Behavioral level 모델링 및 검증

- 상세 설계 이전에 설계 사양을 확인할 수 있도록 시스템의 전체기능을 모델링하고 검증하는 과정

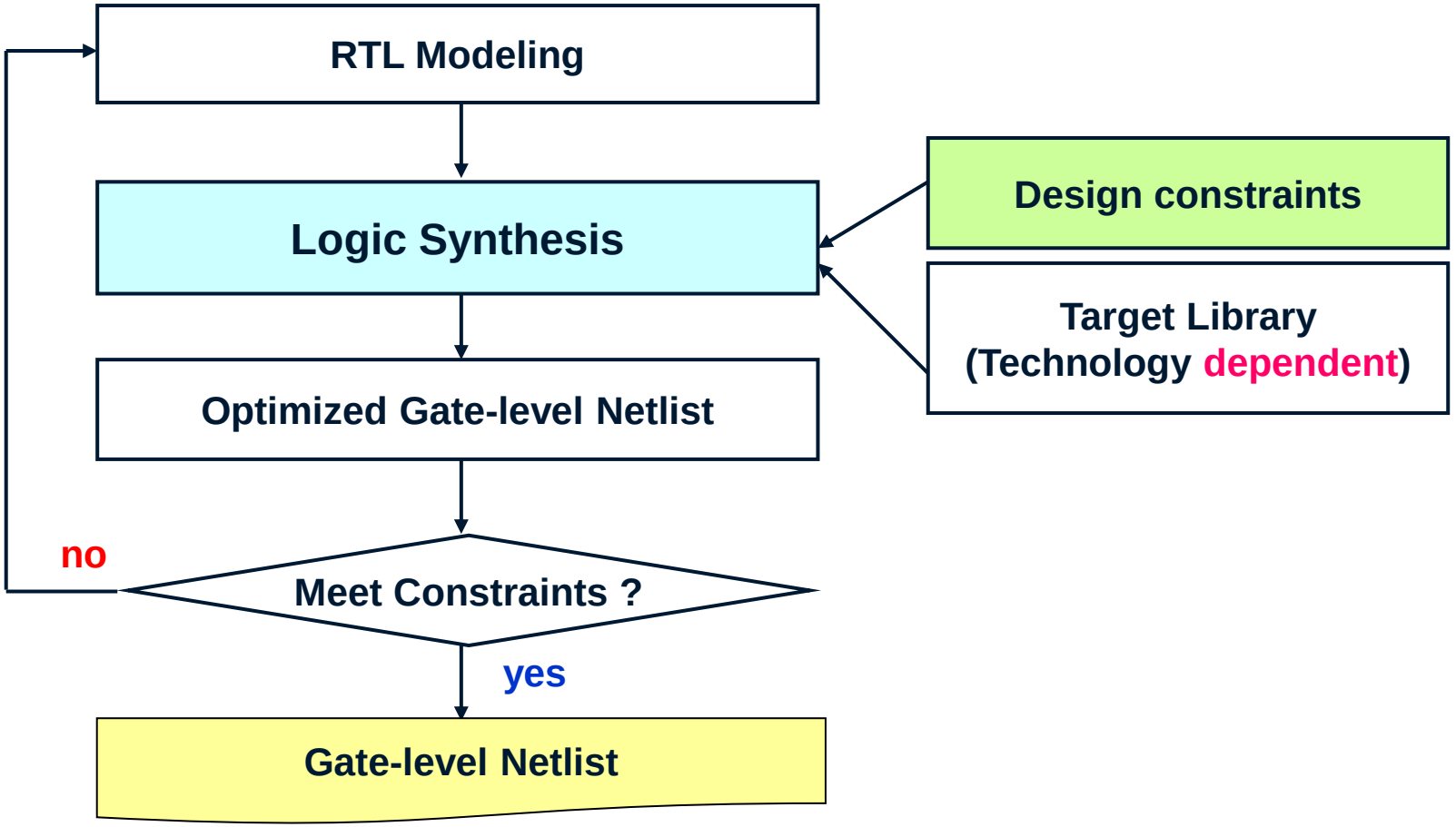
- 설계사양에서 정의된 기능의 만족 여부,
  입/출력 인터페이스의 호환성,
  국제표준규격의 만족 여부 등을 검증

- C 언어, HDL, SystemVerilog, SystemC 등을 사용

RTL 설계

- HDL 사용
  - **Verilog** 또는 VHDL
- RTL (Register Transfer Level) 모델링을 통한 **상세 설계**
  - 논리합성 툴에서 **합성 가능한 코드**로 개발
- 최적의 하드웨어가 합성되도록 코딩하기
- 검증을 위한 테스트벤치 (testbench) 생성

합성


RTL Modeling
Logic Synthesis
Optimized Gate-level Netlist
Meet Constraints ?
no
yes
Gate-level Netlist
Design constraints
Target Library
(Technology dependent)

HDL 기반 설계의 장점

- 설계 시간의 단축
  - 초기 설계과정에서의 설계오류 수정이 용이
  - 합성에 의한 회로 생성과 설계 변경이 용이

- 설계의 질 향상
  - 우수하고 광범위한 하드웨어 기술 능력, 상위 수준의 설계 가능
  - 다양한 설계기법의 검색에 의한 최적화 도달
  - 선택적 최적화 기법을 이용한 합성 설계

- 특정 설계기술이나 공정과 무관한 설계
  - 특정 ASIC 제조업체 및 구현기술과 무관한 설계 가능
  - 동일한 HDL 설계의 다른 라이브러리 이용한 합성

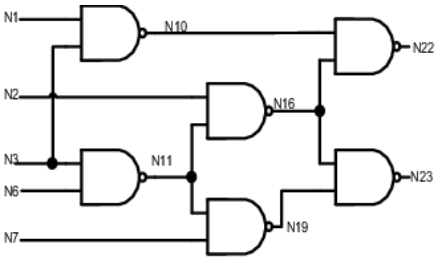
HDL 기반 설계의 장점

- 낮은 설계 비용
  - 상위레벨 설계도구의 사용에 따른 설계 생산성 향상
  - 설계기간의 단축에 따른 설계비용의 감소
  - 설계자산의 재사용에 의한 설계비용의 감소
- 표준 HDL 및 사용자의 확대
  - IEEE 표준인 동시에 미국 정부의 공인 HDL
  - 전세계적으로 설계 및 설계정보 교환의 수단으로 사용이 확대
- 효율적인 설계관리
  - HDL 언어의 구조적 설계 (structured design) 기능을 이용한 전체 설계의 기능별 분할 설계 및 설계관리 및 문서화 용이

HDL 기반 설계는 좋은 점만 있나요?

Verilog:

```
always @({S0,S1}, A, B, C, D)
    case ({S0,S1})
        2'b00: Y = A;
        2'b01: Y = B;
        2'b10: Y = C;
        2'b11: Y = D;
    endcase
```




```
module c17 (N1,N2,N3,N6,N7,N22,N23);
input N1,N2,N3,N6,N7;
output N22,N23;
wire N10,N11,N16,N19;
nand NAND2_1 (N10, N1, N3);
nand NAND2_2 (N11, N3, N6);
nand NAND2_3 (N16, N2, N11);
nand NAND2_4 (N19, N11, N7);
nand NAND2_5 (N22, N10, N16);
nand NAND2_6 (N23, N16, N19);
endmodule
```