



# Design Compiler for Synthesis

## (1 : 디자인 합성의 시작)


2014. 02. 11  
KYUNG-HEE UNIV.  
CSA & VLSI  
CHOI MIN SU

### 1. UNIX 기본 명령어 및 VI 편집기

- ✓ 현재 대부분의 디자인 툴은 UNIX 기반의 OS가 설치된 서버 (우분투, 솔라리스 등)에서 실행되기 때문에 Design Compiler (DC)를 사용하기 전에 반드시 UNIX 기본 명령어 및 VI 편집기 기능을 알아두어야 한다. 그러나 프로젝트를 시작하기 전부터 아래의 명령어들을 외우려고 노력하지 마라. 처음 프로젝트를 수행할 때 매뉴얼을 보고 하나씩 따라하다 보면 차츰 자연스럽게 사용할 수 있게 될 것이다.
- ✓ root 계정 : UNIX 시스템은 여러 명의 사용자가 동시에 하나의 시스템을 사용하는 것을 허락하지만 근본적으로 시스템을 관리할 수 있는 계정은 root 단 하나뿐이다.
- ✓ root는 시스템에서 가장 강력한 권한을 갖는 계정이며 시스템의 중요한 파일을 삭제할 수 있는 권한이 있기 때문에 주의하여 사용해야 한다. 만일 본인이 초보자라면 서버의 root 계정에 접근하지 않을 것을 추천한다. root 계정을 통한 툴 설치 및 시스템 제어의 연습이 필요할 경우 본인 PC에 VM-ware를 설치하여 연습하기 바란다.
- ✓ 사용자 계정을 전환하는 명령어는 다음과 같다. [사용하고자하는 계정]에 전환하고자 하는 ID를 입력한 후 암호를 입력한다. root 계정으로 전환을 원한다면 [사용하고자하는 계정]에 아무것도 넣지 마라.

```
사용법 : su   
su [사용하고자하는 계정] 
```

- ✓ 파일 리스트를 확인하는 명령어는 다음과 같다. 이 명령어는 DOS 운영체제의 dir과 같은 기능을 수행한다.

```
사용법 : ls [옵션] 
```


[옵션]

- a : 'all' 숨김파일 포함, 모든 파일 표시
- l : 'long' 세부적인 정보 확인 가능

- ✓ 현재 위치를 확인하는 명령어는 다음과 같다. UNIX기반의 툴은 GUI 기반이 어색하고 개발환경에 적합하지 않기 때문에 현재 path를 확인하는 경우가 빈번히 발생한다.

사용법 : pwd 

- ✓ 현재 작업 폴더에서 다른 작업 폴더로 경로를 변경하기 위한 명령어는 다음과 같다. 이것은 DOS 운영체제에서 사용하는 명령어와 동일하다.

사용법 : cd [변경하고자하는 디렉토리명] 


- ✓ 상대경로(Relative Path) 표시방법과 절대경로(Absolute Path) 표시방법에 대해 알아보자.

- ✓ 상대경로란 현재 자신이 위치한 폴더부터 시작되는 경로명을 의미한다. 리눅스 시스템에서 상대경로를 지정하는 방법으로 ./ 표시를 사용한다. ./ 표시는 현재 폴더를 의미한다. 만약 내가 상대경로를 사용하여 현재 폴더내의 A라는 폴더를 지정하고 싶다면 ./A 를 사용한다.


- ✓ ../ 표시는 현재 폴더의 상위 폴더를 의미한다. 현재 내 위치가 C 폴더 내의 B 폴더에 위치하고 있으며 하위 폴더로 A 폴더를 갖고 있다고 가정하자. ../를 사용할 경우 상위 폴더인 C 폴더를 지칭하게 되며 ./를 사용할 경우 현재 폴더인 B 폴더를 의미한다. 또한 ./A를 사용할 경우 현재 폴더 내에 있는 A 폴더를 의미하게 된다.

- ✓ 절대경로란 루트(/) 폴더로부터 시작하는 경로명을 의미하며 / 표시를 사용한다. 만약 현재 작업 폴더에서 상대경로를 사용하여 특정위치의 라이브러리를 설정했을 경우 작업 폴더가 변경될 경우 라이브러리 Path 설정 또한 변경되어야 한다. 그러나 절대경로로 라이브러리를 설정했을 경우 이러한 수고로움을 덜 수 있다. 물론 단점도 있다. Path명이 매우 길어진다.

- ✓ 새로운 폴더를 생성하는 명령어는 다음과 같다.

사용법 : mkdir [옵션] [생성하고자하는 디렉토리명] 

- ✓ 파일 및 폴더의 접근 권한 변경을 위한 명령어는 다음과 같다. ls -l 명령어를 사용하여 파일 및 폴더의 접근 권한을 확인할 수 있다.

사용법 : chmod [8진수표기] [파일 및 디렉토리명] 

ex) rwxrwxrwx (r : 읽기, w : 쓰기, x : 실행)



왼쪽 3bit : 소유자 자신의 접근에 대한 허가사항

중간 3bit : 소유자가 속한 그룹의 접근에 대한 허가사항

오른쪽 3bit : 그 외의 기타 사용자

[8진수표기] 777 일 경우 "111111111" -> 모든 사용자에게 관하여 권한 부여



✓ 파일 및 폴더(옵션설정)의 복사 명령어는 다음과 같다.

```
사용법 : cp [옵션] [복사할파일] [대상파일]   
cp [옵션] [복사할디렉토리] [대상디렉토리] 
```

[옵션]

- a : 파일의 속성 링크 정보들을 그대로 유지하면서 복사
- r : 폴더 복사
- f : 강제복사(복사할건지 물어보지 않음)




✓ 파일 및 폴더(옵션설정)의 삭제 명령어는 다음과 같다.

```
사용법 : rm [옵션] [삭제할파일]   
rm [옵션] [삭제할디렉토리] 
```

[옵션]

- a : 파일의 속성 링크 정보들을 그대로 유지하면서 삭제
- r : 폴더 삭제
- f : 강제삭제(삭제할건지 물어보지 않음)


✓ 파일 및 폴더(옵션설정)의 이동 명령어는 다음과 같다.

```
사용법 : mv [옵션] [원본파일명] [변경경로 및 파일명]   
mv [옵션] [원본파일명1] [원본파일명2] ... [변경경로 및 폴더명]   
mv [옵션] [원본폴더명] [변경경로 및 폴더명] 
```

[옵션]

- i : 파일을 이동하기 전 겹쳐 쓰기 여부를 확인
- r : 폴더 이동
- f : 강제이동


✓ 지정된 여러 개의 파일들을 Archive라 불리는 하나의 파일로 묶거나, 하나의 Archive 파일을 원본의 형태로 풀어주는 명령어는 다음과 같다. 이것은 일반적으로 압축하기 전 다수의 파일을 하나로 묶기 위해 사용되며 압축과는 다른 기능이다. tar은 Tape Archive의 약어이다.

```
사용법 : tar [옵션] [만들파일이름] [묶을파일이름] 
```


[옵션]

- c : tar 파일을 생성할 때 사용
- x : 묶여진 tar 파일을 해체할 때 사용
- v : 파일들을 묶거나 해체할 때 그 과정을 자세하게 보여줌
- f : 사용할 tar 파일을 지정
- ※ 일반적으로 파일을 묶을 경우 -cvf옵션을 사용하며 파일을 풀어낼 경우 -xvf 옵션을 사용


✓ 압축 명령어는 다음과 같다.

```
사용법 : gzip/gunzip [옵션] [파일이름] 
```

✓ Tcl 파일 혹은 매크로 파일을 실행하기 위한 명령어는 다음과 같다.

```
사용법 : source [실행할파일이름] 
```

✓ VI 편집기는 간단한 문서편집부터 소스코딩까지 다양한 활용이 가능한 소프트웨어다. 실행방법은 다음과 같다.

```
사용법 : vi [파일이름] 
```

※ [파일이름]과 동일한 파일이 있을 경우 불러오기가 실행, 없을 경우 새 파일 생성

- 세 가지 모드 지원

- ① 명령모드 : 명령어를 사용하여 글자 삭제 및 교체, 문자열을 검색하는 모드로 다양한 편집을 가능하게 하는 모드이다. 따라서 명령모드에서 명령이 아닌 키보드 입력은 모두 에러처리 된다.
- ② 입력모드 : 원하는 글자를 입력 및 수정하고, 화면은 입력한 상태 그대로를 보여주는 모드
- ③ 라인모드 : ESC 키를 누른 후 콜론(:) 프롬프트에서 명령을 입력하며 저장, 편집, 검색, 종료기능을 제공한다.

< 명령모드에서 수정을 위한 입력모드로 변경 시 명령어 사용 >

- a : 현재 위치의 다음부터 입력시작
- A : 현재 줄의 끝에서부터 입력 시작
- i : 현재 위치의 앞에서부터 입력 시작
- I : 현재 줄의 처음에서 입력 시작
- o : 현재 줄과 다음 줄 사이에 입력 시작
- O : 현재 줄과 앞 줄 사이에 입력 시작

< 라인모드에서 종료명령 사용 >

- 명령모드에서 ESC를 누를 경우 라인모드로 전환
- 입력모드에서 ESC를 누를 경우 명령모드로 전환
- :q 그대로 종료
- :wq 저장 후 종료

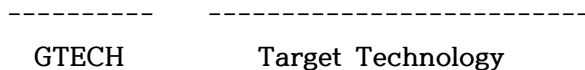
※ 명령모드의 명령어 및 그 외의 명령어는 생략

(본 가이드에서는 작업 시 필요한 최소 명령어만 언급하였다).

## 2. Design Compiler의 시작

✓ 논리합성(Logic Synthesis)이란 무엇인가? Logic Synthesis는 HDL로 기술된 Design을 Read 단계에서 GTECH(generic technology)라 불리는 일반적인 basic logic gate 및 flip-flop으로 변환 (Translation)하는 과정을 거친 후, Compiler이라는 명령을 통하여 logic의 크기를 변환하거나 통합하여 최적화(Optimization) 및 연결(Mapping) 과정을 수행하여 결과(netlist)를 얻는 작업이다. 이때 Optimization 및 Mapping을 위한 각 반도체 제조 벤더에 맞는 공정 라이브러리를 이용한다.

✓ Synthesis = Translation + Logic Optimization + mapping



✓ Design Compiler는 설계자가 정한 요구조건(Constraints)에 맞춰 자동으로 논리합성(Logic Synthesis)을 수행하는 Synopsys 업체에서 개발된 툴이다.

✓ 유사한 기능의 합성기로 다음과 같은 툴이 있다.

- Cadence 업체에서 개발한 논리 합성기 : Encounter RTL Compiler
- Altera 업체에서 개발한 FPGA 합성기 : Quartus
- Xilinx 업체에서 개발한 FPGA 합성기 : ISE, Vivado(상위레벨 통합버전)

✓ Design Compiler와 Encounter RTL Compiler는 Custom IC 제작을 위한 로직 레벨의 논리 합성기로 서로 유사한 기능을 수행하기 때문에 설계자가 사용하고자하는 툴을 결정할 수 있지만, FPGA 툴인 Quartus 및 ISE는 사용하고자 하는 FPGA 칩이 어느 벤더인지(Altera or Xilinx)에 따라서 달라질 수 있다.

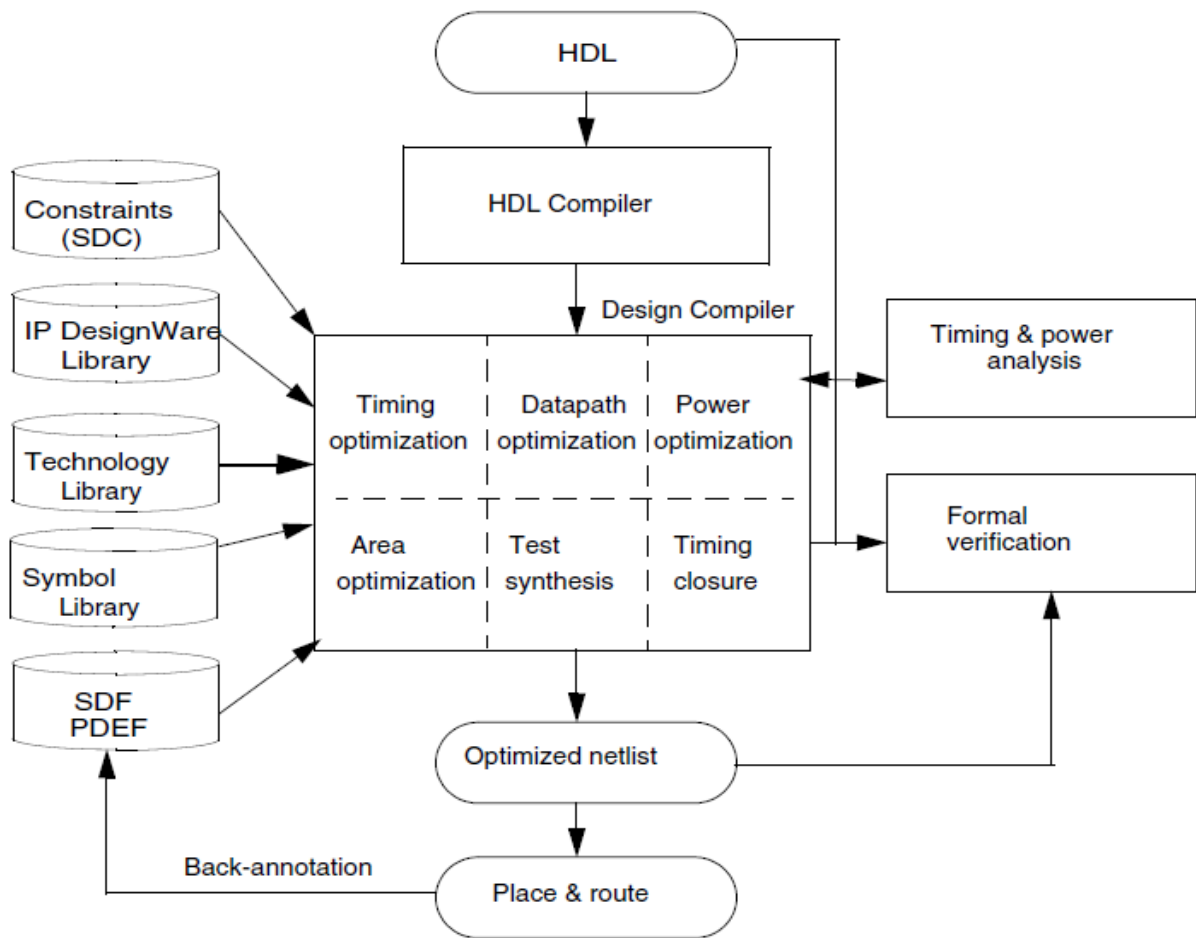
✓ SoC를 처음 접해보는 학부생의 경우 학부 때 배운 Quartus가 합성의 Total Solution이라고 생각하는 경우가 간혹 있는데 Custom IC를 위한 로직합성과 FPGA 합성은 분명히 다른 종류이며 Quartus 또한 모든 종류의 FPGA 합성을 지원하지 않는다(해당 벤더의 칩만 가능).

✓ 본 매뉴얼에서는 Design Compiler를 중심으로 설명한다.

✓ 아래 그림은 Design Compiler의 전체 설계 흐름을 보여준다.

✓ 이 매뉴얼에서는 여러분들이 기본적으로 HDL (Hardware description language) 코딩 능력을 갖추고 있다고 가정한다. HDL이란 의미 그대로 하드웨어를 묘사한 언어적 표현이다. 이 언어적 표현을 실제 하드웨어로 바꾸기 위한 첫 번째 단계가 논리 합성이며 우리는 Design Compiler라는 툴을 사용하겠다고 이미 언급했다. 따라서 논리 합성을 위해서는 반드시 이미 설계된 HDL 디자인이 있어야 한다.

✓ 그럼 HDL과 Design Compiler 중간에 위치한 HDL Compiler란 무엇인가? HDL 컴파일러란 HDL의 문법 및 논리적 구문 오류를 검사하고 Functional 시뮬레이션을 수행하기 위한 도구로 대표적으로 알려진 Model SIM, Active HDL, VCS 및 NC-verilog 등이 있다.



✓ HDL Compiler는 논리합성을 수행하는 툴이 아니다. 문법검사, 구문검사를 거쳐 Functional 시뮬레이션을 수행하며, 합성을 수행하지 않기 때문에 시뮬레이션 파형을 확인하는데 소요되는 시간이 매우 짧다. 그렇지만 이것은 HDL 언어의 구문 구조와 문법 오류에 대한 검사만을 수행한 것이므로 합성에 관한 어떠한 오류나 경고 메시지를 설계자에게 제공하지 않는다.

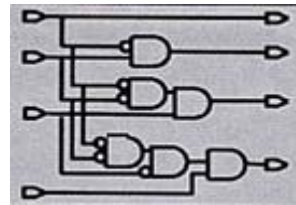
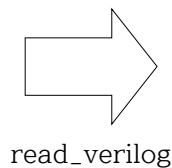
✓ HDL Compiler로 컴파일 했을 때 오류도 경고도 없는 상태로 정상 파형까지 확인했더라도 동일한 디자인을 합성기로 합성했을 때 오류나 경고가 발생할 가능성이 있다.

✓ 이제 Design Compiler를 살펴보자. DC가 HDL 파일을 읽어 들이는 순간(Read) 이 녀석은 GTECH 및 Design ware library를 사용하여 아래의 그림과 같이 HDL Code의 Translation을 수행한다. GTECH는 가장 기본적인 Gate 및 flip-flop만을 포함하고 있으며 Design ware library는 adder 및 comparator와 같은 조금 더 복잡한 cell을 포함한다. GTECH와 Design ware library는 Synopsys 툴에서 기본적으로 제공하는 라이브러리로 이 결과는 specific technology library와 Mapping이 수행된 상태는 아니다.

```

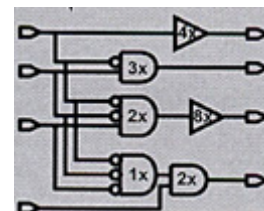
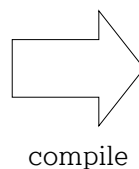
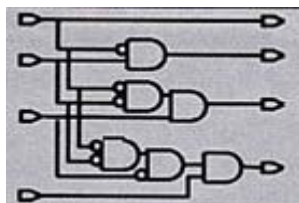
if (high_bits == 2'b10)
    residue = state_table[index];
else
    state_table[index] = 4'b0001;

```



✓ 본격적인 합성이 시작되면 GTECH로 합성된 게이트들은 정해진 Target library를 사용하여 Mapping을 수행한다. 이 매뉴얼에서는 Target library로 삼성 0.13um 공정을 사용할 것이다.

✓ GTECH는 기본 게이트와 flip-flop만으로 구성된다. 하지만 Target library는 다양한 게이트 및 최적화된 구조의 cell들로 구성되어 있다. 아래 그림과 같이 논리합성은 단순한 Mapping만을 수행하는 것이 아니라 Target library를 이용하여(여기서는 삼성 공정) 최적화된 게이트로 변환하거나 통합하는 과정을 함께 수행한다.



✓ 한 가지만 더 생각해보자. 컴퓨터는 단순하고 빠르고 멍청하다. 멍청한 것이 여기서는 키워드다. 멍청한데 어떻게 회로를 최적화를 시킬까? 누군가 이렇게 저렇게 최적화 시켜라 라는 오더를 줘야한다. 바로 그 오더 파일이 Constraint 파일(SDC)이다. 이건 누가 만드나? 당연히 너넨이지. 이것은 뒤에서 더 자세히 설명할 것이다.

✓ 국일호 교수님께서 이전 수업 시간에 이런 말씀을 하셨다.

“너 HDL 코딩 잘하니? 진짜 코딩 잘하는 사람은 CLK랑 딱 필요한 Constraint만 쫓~금 넣고 합성한다.

왜 그런지 아니? 그래도 결과가 잘나와

합성할 때 Constraint이 주절주절 많은 건 네가 만든 HDL 코드가 거지같기 때문이야.

그렇잖아~ 성능도 안 좋고 동작도 안 하니까 이것저것 넣어서 컴퓨터한테 알아서 해달라고 맡기는 거잖아 근데 세상에 그런 완벽한 코드가 어디 있니? 전부 사람이 짜는 건데~“

✓ 이제 위 그림의 왼쪽 입력들은 Symbol library와 SDF,PDEF를 제외하고 전부 설명했다. 그럼 Symbol library는 뭘까? 이름만 들어도 예측할 수 있을 것이다. 이것은 Schematic을 위한 라이브러리의 Symbol 이 저장되어 있다. GUI 환경의 Design vision을 통해 합성된 결과를 symbol 구조로 확인할 수 있다. SDF 및 PDEF는 타이밍 정보를 포함한다. 합성을 수행할 때 각 라이브러리는 게이트 내부 지연 정보를 포함하고 있으므로 이를 기반으로 최적의 타이밍 path를 추출하여 Optimized netlist를 추출할 수 있다. 그러나 이것은 cell 과 cell 사이의 inter connect delay를 포함하고 있지 않다. 왜냐하면 interconnect delay는 Place & route과정을 거쳐서 실제 물리적 라인의 배치가 된 후 알 수 있는 정보이기 때문이다. 따라서 P&R이 수행된 후 이러한 interconnect delay 정보의 최적화를 위해 다시 합성기에 보내줄 수 있는데 이것을

back-annotation이라고 한다.

- ✓ 이런 모든 정보를 기반으로 DC는 Mapping 및 Timing, Data-path, Power, Area Optimization을 수행한다. 이 결과는 Timing & Power analysis를 통해 설계자가 확인할 수 있다.
- ✓ DC는 기본적으로 STA(Static timing analysis) 및 Power analysis 기능을 포함하고 있으나 더욱 더 정밀한 분석을 위해서는 Prime time이라는 툴을 사용할 것을 추천한다. Prime time의 자세한 설명은 해당 매뉴얼에서 하도록 하겠다.
- ✓ Formal verification은 HDL 코드와 합성 후 추출된 netlist를 수식적으로 비교하여 Match여부를 확인하는 과정이다. Synopsys에서는 Formality라는 툴을 사용한다. Formality의 자세한 설명은 해당 매뉴얼에서 하도록 하겠다.
- ✓ 지금부터는 우리가 수행하게 될 Design Compiler에 어떤 Sub 기능들이 존재하는지 살펴보자. Design Compiler라고 부르지만 실제로 이 Tool은 여러 가지 Sub-Tool을 포함한 하나의 Family로 존재한다.
- ✓ Design Compiler는 합성을 위한 솔루션으로 DC Expert와 DC Ultra를 지원한다. 이름에서도 알 수 있겠지만 Ultra는 high-performance를 위한 합성도구이며 Ultra 라이선스를 추가로 구매해야 사용할 수 있다. 사용방법은 매우 간단하다. 최종 컴파일 명령어 뒤에 옵션으로 ultra를 붙여주면 DC Ultra를 사용하여 합성을 수행한다. 현재 연구실은 Ultra 라이선스를 보유하고 있다.
- ✓ Ultra를 적용했을 때 수행되는 합성 알고리즘에 대한 설명은 'Design Compiler Optimization Reference Manual'에 자세하게 기록되어 있다. 이 부분에 대해서는 Ultra 기능만을 자세하게 언급할 예정인 후속 매뉴얼에서 상세하게 설명하도록 하겠다.
- ✓ 또한 DC는 HDL Compiler, DesignWare Library, DFT Compiler, Power Compiler 및 Design Vision의 기능을 추가로 함께 지원한다. HDL Compiler와 DesignWare Library는 위에서 이미 설명했으니 생략하도록 하겠고, DFT Compiler는 Scan chain을 포함한 디자인 검증을 위한 합성을 수행하며 Power Compiler는 전력 소비를 최적화하기 위한 라이브러리 및 합성을 제공한다. 'DTF Compiler documentation' 및 'Power Compiler User Guide' 문서를 통해 자세한 기능을 확인 할 수 있으며 Ultra와 마찬가지로 자세한 기능을 후속 매뉴얼에서 상세하게 설명하도록 하겠다.
- ✓ Design Vision은 GUI 환경을 제공하는 Tool이지만 대부분의 개발자들은 배치 파일의 편리함 때문에 GUI 이용을 권장하지 않는다. 일반적으로 디자인의 최적 Constraint만을 찾는데 최소 10회 이상의 합성이 필요하다. 또한 매 합성마다 40~50 종류의 명령어를 입력하는데 이를 하나의 배치 파일로 만든 후 파라미터만 변경하여 한 번에 실행하는 것은 작업 효율을 높이는 좋은 방법 중 하나다. 만약 GUI 환경에서 합성을 할 때마다 40~50번씩 마우스 클릭을 해야 한다고 생각해보자. 원하는 결과를 보기 전에 직업을 바꾸게 될 것이다. 배치 파일 작성하는 것을 꼭 습관화하기 바란다.



### 3. 합성을 위한 준비

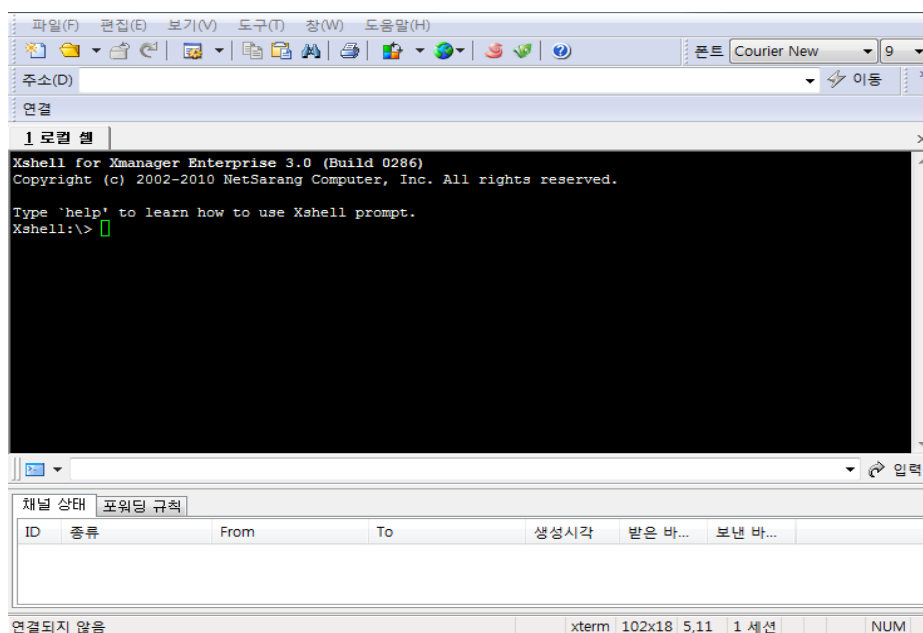
✓ 이제부터 본격적으로 합성을 위한 준비를 시작하겠다. 제일 먼저 무엇을 해야 할까? 일단 완성된 HDL 코드가 필요하다. 사실 정확한 표현은 미완성 HDL 코드가 맞을 것이다. 난 코딩도 잘했고 시뮬레이션 파형도 확인했는데 왜 미완성이야? 라고 반박할 수 있을지도 모른다. 미리 말하겠지만 시뮬레이션 결과를 보기 위한 Functional HDL 코딩과 합성을 위한 코딩은 완전히 다르다. IF문 하나를 짜면서 이게 합성되면 어떤 모듈이 생성 될 지, 다음 Flip-Flop까지 조합논리회로가 얼마나 많이 생성 될 지, 3상태 버퍼가 생성되는 것은 아닐 지, Critical path가 너무 길지는 않을지 혹은 Hold-time에 영향을 주는 라인을 삽입한 것은 아닐지 이 모든 것들을 생각하며 설계 했나 곰곰이 생각해보도록 하자.

✓ 아마 처음 HDL 코딩을 하는 대부분의 학생은 C언어 코딩 스타일을 고려하여 HDL을 다뤘을 것이다. 이 경우 HDL Compiler는 위에서 언급한 어떤 경우의 오류나 경고도 제공하지 않지만 시뮬레이션은 정상적인 파형을 출력한다. 하지만 정작 합성기를 돌렸을 때 얻는 수많은 오류와 경고 메시지를 하나하나 풀어가다 보면 미완성 HDL 코드라는 말을 조금은 공감할지도 모른다.

#### ✓ 준비물 1. HDL 코드

✓ HDL 코드가 준비 되었다면 지금부터 실제 합성을 위한 환경인 서버로 접속을 시도해 보자. 연구실의 라이선스 서버는 솔라리스 OS를 사용 중이고 Tool이 설치되어 있는 워크스테이션은 CentOS를 사용한다. 개발에 사용되는 대부분의 Tool과 라이선스의 설치는 이미 완료되어 있는 상태이며 이상이 있을 경우 연구실 서버 담당자에게 문의하기 바란다. (절대 root계정을 사용하여 혼자 설치를 시도하지 말고 아무리 간단한 설치라도 서버 담당자에게 꼭 확인을 받기 바란다.)

✓ 이 서버에 접속하기 위해서 접속 터미널이 필요하다. 여기서는 프리웨어인 Xshell 3.0을 사용한다. 아래 그림은 Xshell 3.0을 실행했을 때 초기 화면을 보여준다.



## ✓ 준비물 2. 서버 접속을 위한 Xshell 프로그램 설치

✓ 이제 접속을 해보자. Tool이 설치된 워크스테이션의 서버 이름은 Dell이며 주소는 163.180.118.163번이다. 이것은 2014년 2월 기준이며 서버 관리자에 의해 변경되어질 수 있다.

✓ Xshell과 위의 주소를 적용하여 접속을 시작한다. 단 패스워드가 설정된 사용자 ID가 있어야 한다. 처음 접속하는 학생이라면 서버 관리자에게 ID 생성을 미리 요청하도록 해라.

✓ 접속 했을 때 화면이다. 지금부터 모든 파일 관리 및 실행은 1장에서 언급했던 UNIX 기본 명령어만을 사용해야 한다.

```
Xshell for Xmanager Enterprise 3.0 (Build 0286)
Copyright (c) 2002-2010 NetSarang Computer, Inc. All rights reserved.

Type 'help' to learn how to use Xshell prompt.
Xshell:\>
Connecting to 163.180.118.163:22...
Connection established.
Escape character is '^@]'.

Last login: Mon Jan 20 18:47:31 2014 from 163.180.117.64
[mschoi@Dell ~]$
```

✓ 여기서 중요한 점 하나!! 지금부터 파일 및 폴더 관리에 각별히 신경 쓰자. 앞으로 무수히 많은 합성을 시도하고 각기 다른 결과를 얻게 될 것이다. 각 버전, 종류마다 올바른 파일 관리를 하지 않았다면 돌아오는 것은 큰 한숨뿐이다.

✓ 우선 프로젝트를 위한 전용 폴더를 생성하자. 여기서 SAD4by4\_Reference라는 폴더를 생성하도록 하겠다.

```
[mschoi@Dell dk_home]$ mkdir SAD4by4_Reference
[mschoi@Dell dk_home]$ ls
SAD4by4_Reference      intra_prediction
image_rectification_funSyn  motion_compensation_HEVC
[mschoi@Dell dk_home]$
```

✓ 프로젝트는 합성부터 P&R까지 다양한 작업을 수행할 것이다. 분류를 위하여 폴더 내 합성 작업만을 위한 폴더를 추가로 생성하자.

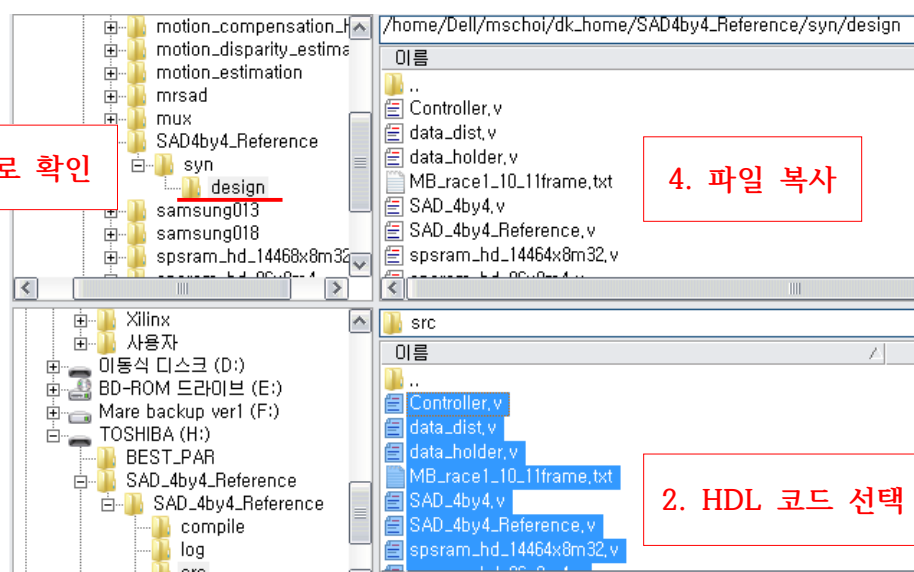
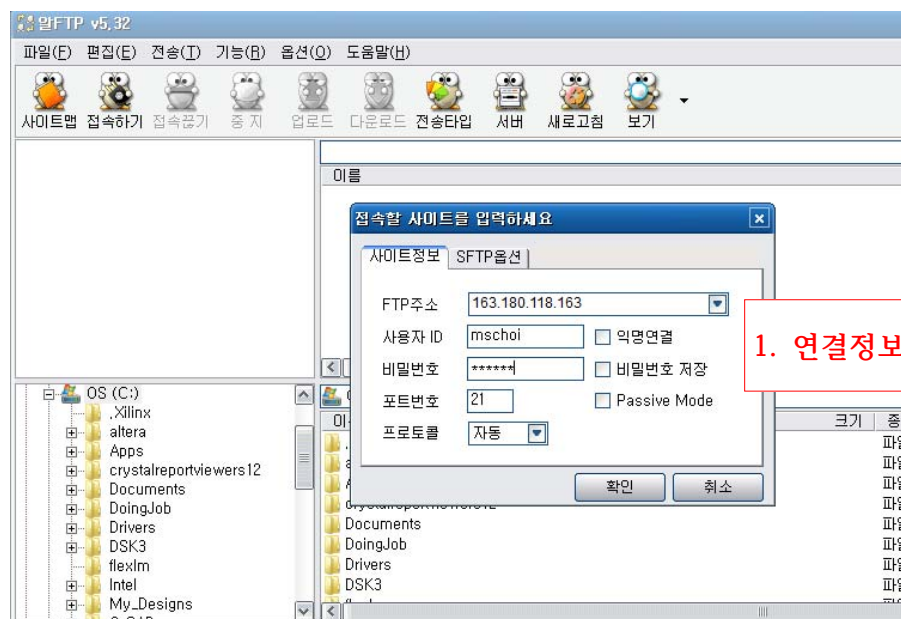
```
[mschoi@Dell dk_home]$ cd SAD4by4_Reference/
[mschoi@Dell SAD4by4_Reference]$ mkdir syn
[mschoi@Dell SAD4by4_Reference]$ ls
syn
[mschoi@Dell SAD4by4_Reference]$
```

✓ 생성된 폴더로 이동하자. 이제부터 모든 합성작업은 이곳에서 이루어 질 것이다.

✓ 이제 준비되었던 HDL 파일을 서버로 가져오자. syn 폴더 안에 HDL 파일을 저장할 수 있는 폴더를(design) 추가로 생성하고 폴더에 파일을 옮기자. 파일을 옮기는 가장 간단한 방법은 리눅스의 FTP 전송 명령어를 사용하는 것이다.

✓ 1장 UNIX 명령어를 설명할 때 FTP 명령어를 사용하는 방법은 언급하지 않았다. 그 이유는 이전 서버담당자였던 영규가 친절하게도 서버에 FTP 프로그램을 설치해 줬다. 그 덕분에 일반 FTP 프로그램 혹은 알FTP 같은 전용 프로그램으로 쉽게 접속하여 파일을 옮길 수 있다. 접속할 때는 서버의 주소를 동일하게 사용하고 ID와 PW도 동일하게 사용한다.

✓ 아래 그림은 알FTP 프로그램을 통해 파일을 옮기는 모습을 캡처한 화면이다.



✓ HDL 파일이 제대로 복사되었는지 서버에서 확인해 보자.

```
[mschoi@Dell syn]$ ls
design
[mschoi@Dell syn]$ cd design/
[mschoi@Dell design]$ ls
Controller.v          SAD_4by4.v           SR_race1_10_11frame.txt  Stage2_RCA_8bits.v  data_dist.v  spsram_hd_14464x8m32.v
MB_race1_10_11frame.txt  SAD_4by4_Reference.v  Stage1_PE_SAD.v         Stage3_RCA_8bits.v  data_holder.v  spsram_hd_96x8m4.v
[mschoi@Dell design]$
```

✓ 이제 HDL 코드의 준비는 끝났다. 이제부터 추가적인 HDL 코드 수정이 필요하다면 서버에서 VI 에디터를 사용하도록 하자. 다음으로 합성에 필요한 라이브러리를 준비하도록 하겠다. 여기서 의미하는 라이브러리는 공정에서 제공하는 Mapping과 Optimization을 위한 Target library를 의미하며 삼성 0.13um의 라이브러리를 사용하겠다고 언급했었다.

✓ 공정에서 제공하는 합성용 라이브러리는 일반적인 텍스트 파일로 .lib 확장자를 갖는다. 이것은 Library Compiler를 통해서 binary format으로 변환되어진다. 변환되어진 파일은 .db 확장자를 갖는다.

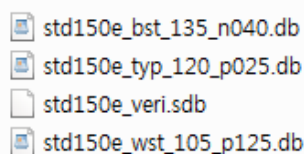
✓ .db 파일을 제공 받았다면 특별한 변환 없이 바로 사용할 수 있다.

✓ .lib 파일을 제공 받았다면 Library Compiler를 사용하여 변환을 수행해야 하며, Design Compile는 Library Compiler 기능을 함께 포함하고 있다.

✓ 변환 방법은 아래 그림과 같다. DC를 실행시킨 후 명령 프롬프트에서 아래의 그림과 같이 입력한다. 물론 해당 폴더에 .lib 라이브러리가 있어야 한다.

```
dc_shell> read_lib libs/aa.lib
Reading '/libs/aa.lib' ...
Warning: Line 45, The default_operating_conditions is not defined. operating_conditions 'WCCOM' is set
as the default_operating_conditions. (LBDB-663)
Technology library 'aa' read successfully
1
dc_shell> write_lib -f db aa
Wrote the 'aa' library to 'aa.db' successfully.
1
```

✓ 그럼 삼성에서는 어떤 형태로 합성용 라이브러리를 제공했는지 확인해 보자. 보이는가? 친절하게도 db형태로 변환하여 주셨다. 3종류의 라이브러리가 있는데 각각 Best, Typical, Worst 환경의 라이브러리이다. 확실한 동작 검증을 위해서는 3가지 경우를 전부 확인해야겠지만 분량 문제로 매뉴얼에서는 Typical만을 사용하도록 하겠다.



```
std150e_bst_135_n040.db
std150e_typ_120_p025.db
std150e_veri.sdb
std150e_wst_105_p125.db
```

- ✓ 라이브러리는 단순히 복사해서 사용하는 것 보다 함께 제공된 설명서를 참조하여 서버에 직접 설치하여 사용할 것을 추천한다. 각 라이브러리마다 아래 그림과 같이 설치를 위한 Make 파일을 제공한다(아래 그림은 설치 파일의 일부임).

```
#!/bin/csh -f
#
# SAMSUNG ASIC Design Kit Install Script
# Scripted by Shin, Dongho (2004.03.05)
# Modified by Shin, Dongho (2004.06.12)
#   - for MS Core dummy DK removal
# Modified by Shin, Dongho (2005.02.23)
# Modified by Shin, Dongho (2005.02.24)
#   - for 90nm DK
# Modified by Shin, Dongho (2005.02.27)
#   - for Cubic DK
# Modified by Park, Dongho (2007.02.09)
#   - for 65nm DK
# Modified by Seo, Christian (2007.08.20)
#   - for changing Custom IP hierarchy level
#
# VERSION 2.3
#
# Usage: sainstall.DK
#       ==> For all of the tar-gzipped DK files in current directory.
#       or sainstall.DK dk_name.tar.gz [dk_name2.tar.gz ... ]
#       ==> For specified tar-gzipped DK file(s).
#
set SUPER_SPECIAL_TOOLS = (synopsys celtic dv conformal physical-compiler ambit)
set COMMON_65 = (doc conformal primetime symbol symbols tetramax synopsys verilog)
set TEMP_DIR = .sainstall
alias sainstall_cat 'cat \! ^ | sed "s| <DK_HOME>| $installed_dir| g"'

set org_dir = `pwd`

onintr ABORT

echo ""
echo "*****"
echo "      SAMSUNG ASIC DESIGN KIT INSTALLER"
echo "  Copyright (c) 2005 SAMSUNG Electronics Co.,Ltd."
echo "      ALL RIGHTS RESERVED."
echo "*****"
echo ""

# Check if the platform is supportable

set os = `uname -s`
switch ($os)
  case "SunOS":
    set rev = `uname -r | cut -d. -f1`
    if ($rev >= 5) then
      set df_cmd = "/bin/df -k"
      set tar_cmd = $PWD/Install/gnutar.solaris
      set zip_cmd = $PWD/Install/gzip.solaris
    else
      echo ""
      echo "$os $rev.x is not supported."
      echo ""
      exit 1
    endif
  default:
    echo ""
    echo "*****"
    echo "      SAMSUNG ASIC DESIGN KIT INSTALLER"
    echo "  Copyright (c) 2005 SAMSUNG Electronics Co.,Ltd."
    echo "      ALL RIGHTS RESERVED."
    echo "*****"
    echo ""
```

- ✓ 설치 파일의 실행을 위해서는 기본적으로 설정되어 있는 bash 셸을 C 셸로의 전환이 필요하다. 전환하는 방법은 csh 명령어를 입력하면 간단히 해결된다. C 셸은 유닉스에서 C언어와 유사하다는 이유로 가장 많이 사용하고 있는 셸 중 하나이다.
- ✓ 여기서 리눅스 셸에 대한 개념을 자세히 설명하지 않겠다. 어려운 내용이 아니니 궁금하다면 직접 찾아서 공부하기 바란다.

- ✓ 현재 기준(2014.02.14.) 우리 연구실의 삼성0.13 라이브러리는 다음 경로에 설치되어 있다. 그러므로 라이브러리를 따로 준비하거나 설치할 필요는 없다. 해당 경로에서 필요한 라이브러리를 찾아서 사용할 수 있다.

/Tools/Library/SS013

- ✓ 합성을 위한 라이브러리는 아래 경로에 존재한다.

/Tools/Library/SS013/SEC150E\_SYNOPSISYS/syn/STD150E

- ✓ 그러나, 지금은 라이브러리, 툴 및 라이선스 설치가 어느 정도 안정화가 되었지만 내가 작업하던 시절만 해도 당시 서버관리자의 넘치는 호기심과 오타쿠 기질로 인해 서버는 춘추 전국시대를 맞이하고 있었다. 고물 서버들은 하나 둘씩 폐기되고 기존의 서버와 새로운 서버는 서로 연결되어 클라우드 시스템을 구축하려 하고 있었다. 지금 사용하고 있는 Dell 서버(163.180.118.163)도 클라우드 시스템의 입구 중 하나일 뿐이다. 대용량 하드의 추가로 자체 백업 시스템이 설치되고 16개의 CPU와 메모리가 연결되어 고속 작업을 할 수 있는 시스템이 도입되었다. 물론 손실도 있었다. 기존에 수행했던 총 3회의 MPW 자료 중 2회 차에 걸친 이전 MPW 자료들이 이를 옮기는 과정에서 실수로 공중분해 되었고 매일 빈번하게 발생하는 툴과 라이브러리 재 설치로 인해 환경 설정 및 라이브러리와 툴 설치 경로가 자주 바뀌었다. 지금의 서버는 로그인을 함과 동시에 툴에 관한 환경설정 및 경로를 자동으로 설정해 주기 때문에 번거로움이 많이 없어졌지만 예전에는 시작 전 일일이 설정을 해야만 했다.

- ✓ 결국 혼돈의 시절이었던 당시 내가 할 수 있었던 유일한 대안은 사용하는 라이브러리 자체를 개인 계정에 설치하여 기존 프로젝트들의 경로가 변경되는 것을 막는 방법 뿐 이었다. 따라서 현재 삼성 0.13 라이브러리는 내 개인 계정도 설치가 되어 있으며 이 매뉴얼에서는 개인 계정에 설치된 라이브러리를 사용하여 모든 작업을 수행 할 예정이다.

- ✓ 본인 계정에 라이브러리를 설치해서 사용할지 위에서 언급한 통합 라이브러리 경로에 설치된 라이브러리를 사용할 것인지는 알아서 판단하기 바란다.

- ✓ 개인 계정에 설치되어 있는 합성용 라이브러리의 경로는 아래와 같다.

/home/Dell/mschoi/dk\_home/samsung013/sec050915\_0050\_STD150E\_regular\_DK\_Synopsys\_N  
/sec150e\_synopsys/syn/STD150E

- ✓ DC에서 사용되는 라이브러리는 Target library, Link library, Symbol library등이 있다. 먼저 Target library는 현재 내가 어떤 해당 공정의 기본 및 응용 게이트들을 포함한 라이브러리로 합성 과정에서 이를 이용하여 mapping과 optimization을 수행한다.
- ✓ Link library는 서로 다른 두 공정, 예를 들어 GTECH to Target Library 혹은 Target Library\_A to Target Library B 사이의 1:1 mapping을 위한 link 라이브러리를 설정한다. DC에서 link 명령어를 사용할 경우 현재 디자인의 게이트 정보를 설정된 link library를 통해 mapping하고 연결 정보를 생성한다.

✓ Symbol library는 각 로직의 symbol을 정의한 library이다. 특별히 지정하지 않는다면 DC에서 제공하는 기본 symbol 모양을 사용하게 된다.

### ✓ 준비물 3. 합성을 위한 라이브러리

✓ 현재 매뉴얼에서 수행하는 프로젝트는 로직 합성 뿐 만 아니라 HARD IP 타입의 SRAM을 포함하므로 다음과 같은 라이브러리가 필요하다.

- std150e\_typ\_120\_p025.db (Target Library)
- spsram\_hd\_96x8m4.db (Target Library for SRAM, SRAM을 사용할 경우 필요)
- spsram\_hd\_14464x8m32.db (Target Library for SRAM, SRAM을 사용할 경우 필요)
- std150e\_veri.sdb (Symbol Library)

✓ 파일 관리를 위해 최종 합성 결과들이 저장될 db폴더 log 메시지를 저장할 log 폴더 작업 내용이 저장될 work 폴더를 추가로 생성하자.

```
[mschoi@Dell syn]$ mkdir db
[mschoi@Dell syn]$ mkdir log
[mschoi@Dell syn]$ mkdir work
[mschoi@Dell syn]$ ls
db design log work
[mschoi@Dell syn]$
```

✓ 모든 준비가 끝났다면 syn 폴더 내에서 디자인 컴파일러를 실행시켜 보자. 이 매뉴얼에서는 Design\_version을 사용한 GUI 환경으로 DC를 시작하며 Design\_version내에서 배치 파일을 실행하여 합성을 수행할 것이다. 이 경우 앞서 설명한 배치파일의 장점과 GUI를 통한 직관적인 결과 확인까지 모두 가능하다.

✓ Design Vision은 다음 명령어를 통하여 실행 시킬 수 있다.

=> design\_vision

✓ 연구실 서버에서는 환경 설정 파일 내에 design\_vision을 dv 약어로 assign했기 때문에 dv만으로 쉽게 실행이 가능하다.

✓ DC Shell 모드로 DC를 실행하기 위해서 다음 명령어가 필요하다.

=> dc\_shell-t

✓ 배치모드의 실행은 다음과 같다.

=> dc\_shell-t -f RUN.tcl | tee -i my.log

✓ 아래 그림은 Design Vision의 초기화면을 보여준다.

✓ DC에 포함되어 있는 다양한 Sub 기능들을 확인할 수 있다. DC Ultra, Power Compiler, DesignWare, DC Expert, Design Vision 등. 위에서 이미 설명한 내용이므로 자세한 내용은 생략한다.

✓ 아래와 같은 GUI 창을 확인하지 못했다면 툴 설치 및 라이선스를 다시 확인하기 바란다.

```
[mschoi@Dell syn]$ dv
```

```
Design Compiler Graphical
DC Ultra (TM)
DFTMAX (TM)
Power Compiler (TM)
DesignWare (R)
DC Expert (TM)
Design Vision (TM)
HDL Compiler (TM)
VHDL Compiler (TM)
DFT Compiler
Library Compiler (TM)
Design Compiler(R)
```

```
Version F-2011.09-SP2 for linux -- Nov 24, 2011
Copyright (c) 1988-2011 Synopsys, Inc.
```

This software and the associated documentation are confidential and proprietary to Synopsys, Inc. Your use or disclosure of this software is subject to the terms and conditions of a written license agreement between you, or your company, and Synopsys, Inc.

```
Initializing...
```

