ETRI 0.5um CMOS Std-Cell DK:

오픈-소스 반도체 설계 도구 설치

연구과제명	반도체 기술 개발 지원 고경력 전문인력 활용 사업(25JB1710)
연구기간	2025 년 6 월~2026 년 12 월
연구책임자	고상춘
기록자	국일호
확인자	
작성일자	2025 년 6 월 20 일

ETRI 한국전자통신연구원
www.etri.re.kr

ETRI 0.5um CMOS Std-Cell DK:

오픈-소스 반도체 설계 도구 설치

목차:

---

1. 개요

"내 칩 제작 서비스(이하 '내 칩 MPW')"의 ETRI 0.5um CMOS 공정용 표준-셀 기반 디자인 킷(이하 '디자인 킷')을 개발하였다. 이 디자인 킷은 오픈-소스 EDA 도구들을 사용하여 SystemC/C++ 에 의한 알고리즘의 기술, 레이아웃 및 RTL 설계, 회로 및 베릴로그 HDL 시뮬레이션, RTL 합성과 표준 셀 자동 배치배선 그리고 칩 테스트까지 전 과정을 수행 할 수 있는 체계를 갖췄다. 본 문서는 내 칩 MPW를 통해 칩 제작을 의뢰하기 위한 레이아웃 GDS 생성에 필요한 설계 도구들의 설치 방법을 설명한다. 사용된 도구들은 모두 '오픈-소스'로써 무료다.

오픈-소스 도구들은 C++의 소스 형식으로 배포되는 경우가 많으므로 사용자가 직접 빌드(build) 및 설치(install) 해야 한다. 물론 무료 사용으로 인해 발생할 오류도 사용자 몫이다. 따라서 사용자는 끝없는 교차 검증에 익숙해 져야 한다. 교차 검증하는 습관은 상용 도구 사용자도 예외는 아니다. 무결한 소프트웨어는 없다는 점을 잊지 말자. 설계 도구들을 운용할 운영체제 역시 오픈-소스 리눅스다. 각 도구들 마다 요구하는 패키지들은 사용 중인 운영체제의 여건에 따라 다르다. 오픈-소스 설계 도구들을 설치하기 전에 APT(Advanced Package Tool)를 통해 필요 패키지들을 미리 설치한다. 이런 번거로움은 일관된 관리 주체가 없다는 오픈-소스의 단점이지만 모두에게 열려 있다는 장점에 비하면 아무것도 아니다.

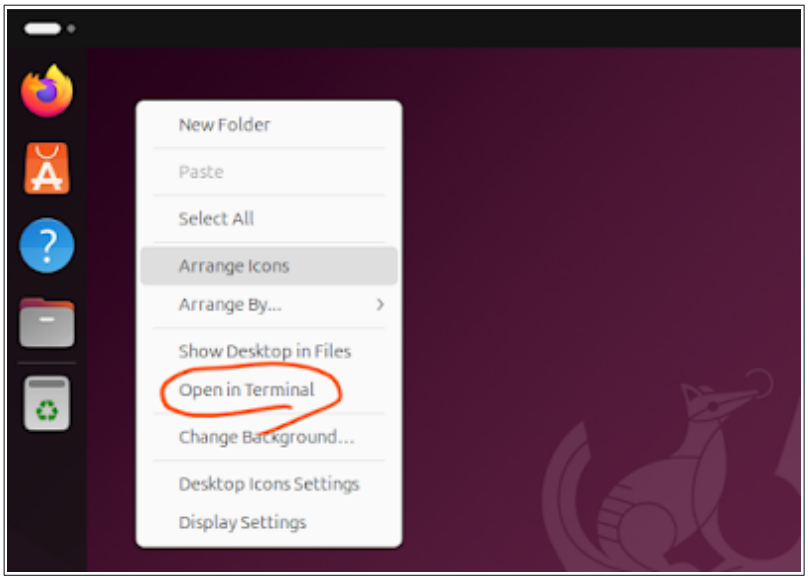
2. 윈도우즈 PC에 리눅스 설치

리눅스가 설치된 독립된 컴퓨터에 리눅스를 설치할 수도 있지만 윈도우즈 11의 가상 머신에 리눅스를 설치하면 윈도우즈 소프트웨어를 활용 하면서 파일 교환, 복사와 붙여넣기, 드래그와 드롭 등 장점이 많다. 윈도우즈 PC에서 가상 머신 리눅스를 설치하는 방법은 아래 문서를 참조한다.

연구노트1, "표준 셀 디자인 킷의 설계 환경, 가상 머신 리눅스 설치" [링크]

3. "내 칩 제작 서비스" 표준 셀 디자인 킷 입수

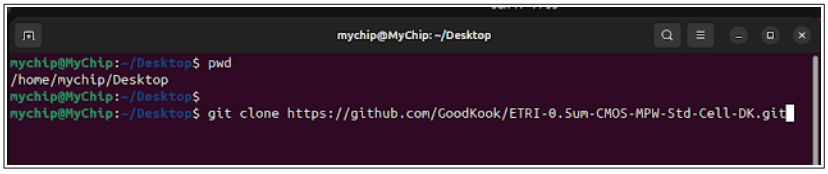
'디자인 킷'은 깃-허브 저장소에서 공개되어있다. 리눅스를 부팅 한 후 데스크 탑 바탕화면에서 마우스 오른쪽 버튼을 눌러 플로팅 메뉴가 나타나면 'Open in Terminal'을 선택하여 터미널을 연다.



터미널에서 아래 명령 입력

```
$ pwd
```

'$'는 리눅스 본-쉘의 명령 입력을 대기하는 프롬프트다. 명령 pwd 는 현재 위치한 디렉토리의 경로명을 보여준다. 현재 디렉토리의 위치는 사용자의 홈 디렉토리 아래의 Desktop 임을 보여준다. '~'는 사용자의 홈(HOME) 디렉토리를 나타내는 기호다.



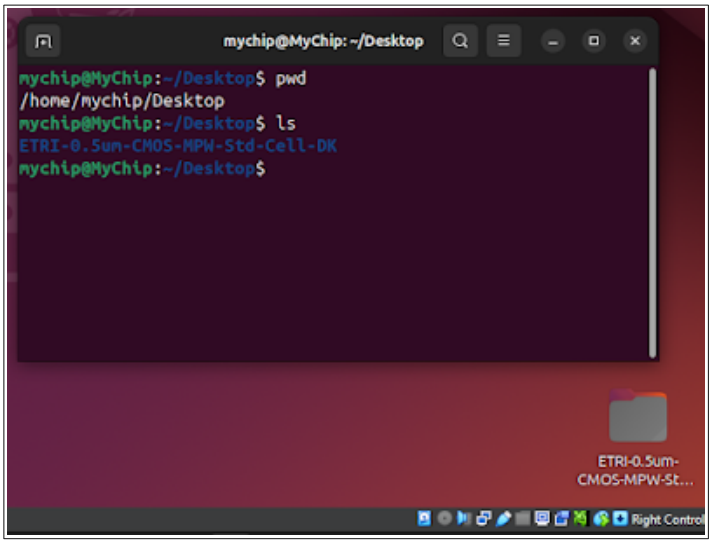
아래 명령으로 깃-허브 저장소에서 디자인 킷을 내려 받을 수 있다.

```
$ git clone https://github.com/GoodKook/ETRI-0.5um-CMOS-MPW-Std-Cell-DK.git
```

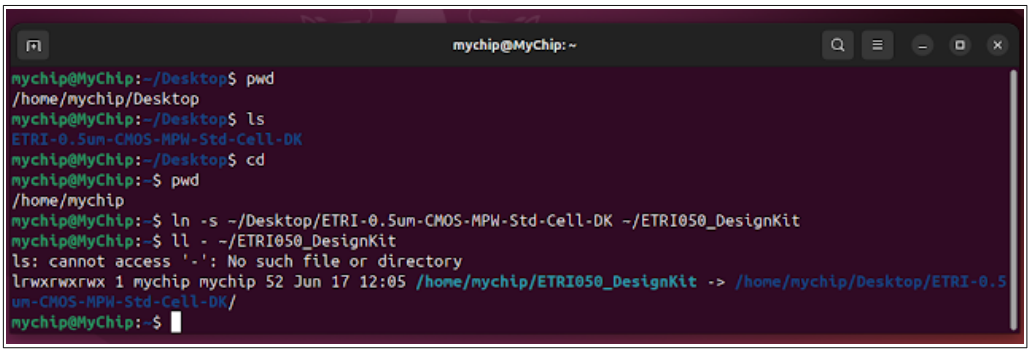
내려받을 자료의 양은 약 1G 바이트 가량이다. 연결된 인터넷의 속도에 따라 달렸지만 자료를 내려받고 파일을 푸는데 약 10분 가량 소요될 것이다. 인터넷 브라우저의 검색창에 아래의 주소를 입력하면 깃-허브 저장소의 디자인 킷을 살펴볼 수 있다.

```
https://github.com/GoodKook/ETRI-0.5um-CMOS-MPW-Std-Cell-DK
```

디자인 킷 내려받기가 정상적으로 완료 되었다면 데스크 탑의 오른쪽 하단에 폴더가 보인다. pwd 명령으로 내려받은 디자인 킷의 디렉토리를 확인 하자.



이어 사용자 홈 디렉토리로 이동한다. 명령 cd는 디렉토리 이동이다. 인수를 주지 않을 경우 사용자의 홈 디렉토리로 이동 한다.



내려받은 디자인 킷의 디렉토리를 홈 디렉토리에 'ETRI050_DesignKit' 이름으로 심볼링크를 건다.

```
$ ln -s ~/Desktop/ETRI-0.5um-CMOS-MPW-Std-Cell-DK ~/ETRI050_DesignKit
```

명령 ln은 파일(또는 디렉토리)을 다른 이름으로 연결 한다. 이 명령에 -s 를 붙이면 윈도우즈의 '바로가기'와 같다. 파일 리스트 명령 ll 으로 앞서 내려받은 디자인 킷의 디렉토리를 사용자 홈 디렉토리의 ETRI050_DesignKit 라는 이름으로 연결 되었음을 알 수 있다. 파일 목록 보기 명령 ll 은 ls -la 의 별칭(alias) 이다. 만일 붉은 색으로 표시 된다면 연결이 틀렸다는 뜻이므로 연결 명을 제거하고 다시 연결 해주자. 파일(디렉토리)의 제거 명령은 rm 이다. 깃 허브에서 내려받은 디자인 킷에는 설계에 필요한 각종 오픈-소스 도구들의 빌드와 설치에 필요한 쉘 스크립는 물론 다양한 예제들을 포함한다.

4. 사전 설치 패키지

오픈-소스 도구들은 개발자의 취향에 따라 사용되는 개발 패키지(라이브러리)들이 제각각이다. 오픈-소스 반도체 설계 도구들은 여러 개발자들의 산물로 구성되었다. 이들이 채택한 개발도구들을 모두 찾아서 설치해 주어야 한다.

디자인 킷의 도구 설치 스크립트가 있는 디렉토리로 이동 한 후 쉘 스크립트에 실행 가능 속성을 부여해 준다.

```
$ cd ~/ETRI050_DesignKit/Tools
```

```
$ chmod +x *.sh
```

미리 준비된 개발용 패키지들을 설치하는 쉘 스크립트를 실행 한다.

```
$ ./prerequisites.sh
```

GNU C/C++ 컴파일러를 비롯하여 bison, flex 등 언어 개발 도구, make, configure 등 소프트웨어 개발에 유용한 유틸리티와 디버깅 도구들이 설치된다. 그밖에 X11, gtk 그래픽 패키지, tcl/tk, python, perl 스크립트 엔진들이다.

5. 오픈-소스 반도체 설계 도구 설치

디자인 킷의 예제에서 사용할 오픈-소스 반도체 설계 도구들의 목록은 아래와 같다. 모두 깃-허브를 통해 소스로 배포 되고 있다.

1. z3: Theorem Prover
2. SDL2: Simple Direct Layer
3. gsl: GNU Scientific Library
4. graywolf: GrayWolf Auto Placer (자동 배치 도구)
5. qrouter: Auto-Router (자동 배선도구)
6. irsim: IRSim Switching Level Simulator (스위치 레벨 시뮬레이터)
7. magic: VLSI Layout Tool (레이아웃 편집기)
8. klayout: Layout Tool (레이아웃 도구)
9. netgen: NetGen Layout vs Schematic compare (LVS 도구)
10. ngspice44: SPICE simulator (회로 시뮬레이터)
11. xschem: XSchem Schematic entry tool (그래픽 회로 편집기)
12. systemc: SystemC (시스템 모델링 C++ 라이브러리)
13. iverilog: iVerilog HDL simulator (베릴로그 시뮬레이터)
14. Verilator: Verilog to C++/SystemC converter (베릴로그 C++/SystemC 변환기)
15. gtkwave: Digital waveform viewer (디지털 파형 보기)
16. yosys: Yosys RTL Synthesizer (베릴로그 합성기)
17. QFlow: qflow-1.4.100_etri050 (설계 툴 플로우 통합 환경)

각 도구들은 모두 깃-허브 저장소를 통해 C++ 소스로 내려 받아 직접 내 컴퓨터에서 빌드 하고 설치한다. 디자인 킷에는 위의 도구들을 저장소에서 내려받아 빌드 하고 설치하는 스크립트를 제공한다. 필요한 도구들을 일괄 설치하는 스크립트의 실행은 다음과 같다.

```
$ cd ~/ETRI050_DesignKit/Tools

$ ./build_tools.sh
```

높은 추상화 수준(C++ 또는 베릴로그 HDL)에서 반도체를 설계하고 제조도면(레이아웃)을 생성하기 까지 여러 종류의 도구들이 동원된다. 다양한 도구들의 내려받은 소스의 빌드와 설치에 많은 시간이 소요된다. 대략 2시간 가량 걸리며 설치하는 중에 관리자 비밀번호(password)를 묻게 될 것이다. 설치가 완료될 때까지 컴퓨터 앞을 떠나지 않도록 한다.

오픈-소스 도구들은 수시로 갱신 되므로 각 도구들을 업 데이트할 필요가 있다. 디자인 킷에 설계 도구들은 개별적으로 설치 할 수 있는 쉘 스크립트들을 갖추고 있다. 각 도구의 내용을 살펴보자.

5-1. Yosys: 베릴로그 합성기

레지스터 전송 수준(RTL)의 베릴로그 HDL을 게이트와 플립플롭의 연결도(netlist)로 변환해 준다. 이때 사용 가능한 게이트와 플립-플롭은 반도체 공장에서 준비한 표준 셀(Standard Cell)을 토대로 디지털 회로를 구성한다.

합성기 Yosys의 설치 스크립트: yosys_build.sh

```
#!/usr/bin/bash

if [ ! -d ./yosys ]; then
  git clone https://github.com/YosysHQ/yosys.git
fi

cd yosys
git pull          # Make sure git repository is up-to-date
git submodule update --init
export CXX=clang++
export CXXFLAGS=-std=c++17

make config-clang
make -j`nproc`
make -j`nproc` test
sudo make install
make clean
```

5-2. Graywolf & QRouter: 배치와 배선(Place & Route)

실리콘 기판(평면) 위에 게이트들을 펼쳐놓고 합성기에서 얻은 연결도 대로 배선을 수행한다. 이 과정은 인쇄회로 기판(PCB) 설계에서 부품의 자동 배치와 자동 배선을 수행 하는 것과 같다. 다만 부품의 갯수가 PCB 설계에 비해 감당 할 수 없이 많다. 따라서 자동으로 수행한다.

배치도구 GrayWolf의 설치 스크립트: graywolf_build.sh

```
#!/usr/bin/bash
if [ ! -d ./graywolf ]; then
  git clone https://github.com/rubund/graywolf.git
fi
```

```
cd graywolf
git pull
mkdir build
cd build
cmake -DCMAKE_C_COMPILER=/usr/bin/clang \
        -DCMAKE_CXX_COMPILER=/usr/bin/clang++ ..
make
make test
sudo make install
```

배선도구 QRouter의 설치 스크립트: qrouter_build.sh

```
#!/usr/bin/bash
if [ ! -d ./qrouter ]; then
    git clone https://github.com/RTimothyEdwards/qrouter.git
fi

cd qrouter
git pull
./configure
make
sudo make install
```

5-3. netgen: 레이아웃과 회로 비교(Layout vs. Schematic)

자동 배치와 배선된 도면 레이아웃이 합성하여 얻은 연결도와 일치하는지 비교한다. 배치와 배선은 자동 소프트웨어가 해준다. 경우에 따라 사람이 고칠 수도 있다. 소프트웨어의 버그, 인간이 저지른 오류가 끼어들 수 있다. 배선이 끝나면 빠진 부분이 없는지 레이아웃에서 연결도(netlist)를 합성으로 생성되었던 연결도와 반드시 비교해야 한다.

LVS 도구 netgen 의 설치 스크립트: netgen_build.sh

```
#!/usr/bin/bash
if [ ! -f ./netgen ]; then
  git clone https://github.com/RTimothyEdwards/netgen.git
fi

cd netgen
git pull
./configure
make
sudo make install
make clean
```

5-4. Magic: 디자인 룰 체크 (Design Rule Check)

배치 배선이 끝나면 표준 셀 라이브러리를 합쳐(merge) 완성된 도면을 생성 한다. 이 도면(레이아웃)이 반도체 공장의 제조 규정(design rule), 예를 들어 선폭, 이격 거리 같은 기하학적인 문제가 없는지 검사한다. 디자인 룰 검사는 레이아웃 편집기 Magic 을 쓴다.

5-5. Magic: GDS 생성

설계 도구들이 읽고 쓰는 파일의 형식은 저마다 다르다. GDS는 반도체 공장에 제출할 도면의 형식이다.

DRC 및 GDS 생성 도구 Magic의 설치 스크립트: magic_build.sh

```
#!/usr/bin/bash
if [ ! -d ./magic ]; then
  git clone https://github.com/RTimothyEdwards/magic.git
fi

cd magic
git pull
./configure
make
sudo make install
```

5-6. QFlow: 통합 관리도구(Design Flow)

RTL의 HDL로부터 GDS 레이아웃에 이르는 과정은 연속적인 추상화 수준을 낮추는 과정이다. 각 단계에서 저마다 사용되는 도구들이 다르다. 입출력 파일의 양식 또한 매우 상이하다. QFlow는 이들 도구들이 요구하는 입출력 파일을 변환 하고 추상화 간격을 맞춰주는 역활은 한다. 아울러 여더 도구들의 운용에 필요한 스크립트를 생성하고 옵션을 설정해준다.

반도체 제조 도면은 칩을 제작할 공장의 설비 조건에 맞춰져야 한다. 이를 보통 테크놀러지 노드(technology node), 줄여서 '노드' 라 한다. "내 칩 제작 서비스"의 반도체 제조공정은 0.5um CMOS이며 2중의 폴리 실리콘과 3개의 메탈층을 가지고 있다. QFlow는 이 공정을 지원하지 않는다. '디자인 킷'은 QFlow에 "내 칩 제작 서비스"의 공정을 이식하여 따로 배포한다. 아래 설치 스크립트를 살펴보면 깃-허브에서 내려 받지 않고 디자인 킷의 압축 tar 를 풀어 빌드 및 설치를 수행한다.

통합 관리도구 QFlow 설치 스크립트: qflow-1.4.100_etri050_build.sh

```
#!/usr/bin/bash
if [ ! -f ./qflow-1.4.100_etri050.tar.gz ]; then
  echo "qflow-1.4.100_etri050 NOT found!"
  exit
fi

tar xvf qflow-1.4.100_etri050.tar.gz

cd qflow-1.4.100_etri050/
./configure
make
sudo make install
make clean

cd /usr/local/share/qflow/tech
if [ -d ./etri050.bak ]; then
  sudo rm -rf etri050.bak
fi
sudo mv etri050 etri050.bak
sudo mkdir etri050
cd etri050

sudo ln -s ~/ETRI050_DesignKit/tech/SCN3ME_SUBM.30.ETRI.tech
SCN3ME_SUBM.30.ETRI.tech
```

```
    sudo ln -s ~/ETRI050_DesignKit/tech/etri050.par etri050.par
    sudo ln -s ~/ETRI050_DesignKit/tech/etri050.sh etri050.sh
    sudo ln -s ~/ETRI050_DesignKit/tech/etri050_setup.tcl etri050_setup.tcl
    sudo ln -s ~/ETRI050_DesignKit/digital_ETRI/khu_etri050.magicrc
etri050.magicrc
    sudo ln -s ~/ETRI050_DesignKit/digital_ETRI/khu_etri050.prm etri050.prm
    sudo ln -s ~/ETRI050_DesignKit/digital_ETRI/etri050_stdcells.lef
etri050_stdcells.lef
    sudo ln -s ~/ETRI050_DesignKit/digital_ETRI/khu_etri050_stdcells.spice
etri050_stdcells.sp
    sudo ln -s ~/ETRI050_DesignKit/digital_ETRI/khu_etri050_stdcells.gds2
etri05_stdcells.gds2
    sudo ln -s ~/ETRI050_DesignKit/digital_ETRI/khu_etri05_stdcells.lib
etri05_stdcells.lib
    sudo ln -s ~/ETRI050_DesignKit/digital_ETRI/khu_etri05_stdcells.v
etri05_stdcells.v
```

5-7. SystemC: 시스템 수준 모델링 및 시뮬레이션

시스템 수준 모델링과 시뮬레이션을 위한 C++ 크래스 라이브러리다. 시스템 수준 테스트 벤치 작성에 매우 유용하다. HDL과 병행 시뮬레이션(Co-Simulation)의 훌륭한 도구다. RTL 설계는 HDL로 테스트 모델은 SystemC/C++로 한다. C++이므로 컴퓨터로 할 수 있는 모든것이 가능하다.

시스템 모델링 라이브러리 SystemC 설치 스크립트: systemc_build.sh

```
#!/usr/bin/bash
if [ ! -d ./systemc ]; then
  git clone https://github.com/accellera-official/systemc.git
fi

source SC_env
cd systemc
git pull

if [ -d ./build ]; then
  rm -rf build
fi

mkdir build
cd build

cmake ..
make
make check
sudo make install
```

SystemC 는 리눅스의 표준 라이브러리가 아니다. 따라서 SystemC 를 활용하여 시스템 모델 개발을 수행 할때 해당하는 정적 라이브러리와 실행형 라이브러리가 존재하는 위치를 시스템 변수로 알려 주어야 한다. SystemC 환경 변수가 자주 사용되므로 필요시 source 명령으로 일괄 설정할 수 있도록 'SC_env' 파일로 작성해 두었다.

SystemC 활용 시 설정해야할 시스템 변수: SC_env

```
#!/usr/bin/bash
export SYSTEMC=/opt/systemc
export SYSTEMC_HOME=$SYSTEMC
export SYSTEMC_INCLUDE=$SYSTEMC_HOME/include
export SYSTEMC_LIBDIR=$SYSTEMC_HOME/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$SYSTEMC_LIBDIR
```

5-8. Verilator: 언어 변환기

Verilator는 RTL 베릴로그 HDL을 SystemC/C++로 변환하는 도구다. 빠른 시뮬레이션을 수행 할 수 있다. 게이트 모델은 변환시 제약이 있다.

Verilator 설치 스크립트: verilator_build.sh

```
#!/usr/bin/bash
if [ ! -d ./verilator ]; then
  git clone https://github.com/verilator/verilator
fi

source SC_env
unset VERILATOR_ROOT
cd verilator
git pull

autoconf
./configure
make -j `nproc`
sudo make install
make test
make clean
```

5-9. iVerilog: 베릴로그 HDL 시뮬레이터

ICARUS Verilog는 오픈-소스 베릴로그 시뮬레이터다.

iVerilog 설치 스크립트: iverilog_build.sh

```
#!/usr/bin/bash
source SC_env
if [ ! -d ./iverilog ]; then
  git clone https://github.com/steveicarus/iverilog.git
fi

cd iverilog
git pull

sh autoconf.sh
./configure
make
sudo make install
make clean
```

주] QuestaSim (상용 HDL 시뮬레이터)은 상용 HDL 시뮬레이터이지만 FPGA 스타터 버젼을 무료로 쓸 수 있다. 상용판에 비해 기능상 제약은 전혀없다. 단지 시뮬레이션 속도가 절반이라고 한다. 아래 링크를 통해 쿼터스 표준 버젼 (Intel® Quartus® Prime Standard Edition)의 설치 파일을 내려 받을 수 있다.

```
  https://www.intel.com/content/www/us/en/software-kit/849752/intel-quartus-
prime-standard-edition-design-software-version-24-1-for-linux.html
```

5-10. ngSpice: 회로 시뮬레이터

트랜지스터와 R(저항), C(컨덴서) 수준 전자회로 시뮬레이터다. 아날로그 회로 설계 및 표준 셀 제작시 시뮬레이터로 사용한다. 시뮬레이션이 정밀 한 만큼 매우 느리다. 작은 설계라도 트랜지스터 갯수가 수천개를 넘기는 RTL 합성 기반 설계환경에서는 사용 불가다. ngSpice는 github 는 버젼 업데이트를 안하고 있다. 소스포지에서 다운 받는다.

회로 시뮬레이터 ngSpice 설치 스크립트: ngspice44_build.sh

```
  #!/usr/bin/bash

  if [ ! -f ./ngspice-44.tar.gz ]; then
    wget https://sourceforge.net/projects/ngspice/files/ng-spice-rework/44/
ngspice-44.tar.gz
  fi

  tar xvf ngspice-44.tar.gz
  cd ngspice-44
  ./autogen.sh
  mkdir debug
  cd debug
  ../configure --with-x --with-readline=yes
  make
  make test
  sudo make install
```

5-11. XSchem: 회로 입력기

문자 편집기를 이용해 네트리스트를 작성하는 대신 그래픽 화면 상에서 직관적으로 회로를 그려줄 수 있다. SPICE는 물론 HDL 네트리스트를 출력해 준다. 여러 오픈 소스 도구중 XSchem 이 있다. ngSpice와 잘 어울린다.

XSchem 설치 스크립트: xschem_build.sh

```
  #!/usr/bin/bash

  if [ ! -d ./xschem ]; then
    git clone https://github.com/StefanSchippers/xschem.git
  fi

  cd xschem
  git pull
  ./configure
  make
  sudo make install
  make clean
```

```
if [ ! -d ./xschem ]; then
  git clone https://github.com/StefanSchippers/xschem-gaw.git
fi

cd xschem-gaw
git pull
aclocal
autoconf
autoheader
automake --add-missing
./configure
make
sudo make install
```

GAW는 ngSpice로 회로 시뮬레이션을 수행 한 결과를 그래픽 파형으로 보여주는 유틸리티다. GAW를 빌드하면서 GETTEXT 버젼이 맞지 않아 오류가 나는 경우, xschem/xschem-gaw/po/Makefile.in.in 에서 아래 변수 설정 변경해 준 후 다시 빌드한다.

```
GETTEXT_MACRO_VERSION = 0.20
```

5-12. Klayout: 레이아웃 도구

반도체 제조용 도면(레이아웃)의 양식은 다양하다. 전통적으로 CIF와 GDS 양식이 사용되었으나 보안에 취약하여 설계 도구 회사들은 저마다 양식을 써왔다. 최근 혼란을 피하기 위해 oa(open-access, 상용 EDA 연합체에서 제정[Sillicon Integration Initiative, Si2])와 oas(OASIS, 오픈-소스)가 제안되고 사용중이다. 레이아웃은 기본적으로 적층된 레이어를 평면 도면으로 표현한 것으로 소자들 사이의 배선을 사각형으로 표현하므로 엄청난 량의 사각형 정보를 담게된다. 실예로 예제의 Z80 마이크로 프로세서의 레이아웃에 사각형은 4.12M 가까이된다(패드 제외). 레이아웃의 검토를 위해 보기만 해도 정교한 소프트웨어가 필요하다. KLayout 은 굉장히 빠른 레이아웃 보기 소프트웨어다. 깃허브의 소스를 내려받을 수 있지만 운영체제별 바이너리로 배포되고 있다. KLayout 웹페이지에서 최신 배포 패키지를 내려받은 후 설치한다. 데비안 패키지 .deb의 설치 명령은 dpkg 다.

```
https://www.klayout.de/
```

klayout은 자유 실리콘 컨퍼런스(Free Silicon Conference)에 참여하고 있다.

```
https://wiki.f-si.org/index.php/FSiC2024
```

KLayout 설치 스크립트: klayout_install.sh

```
#!/usr/bin/bash

if [ ! -f ./klayout_0.30.0-1_amd64.deb ]; then
  wget https://www.klayout.org/downloads/Ubuntu-24/klayout_0.30.0-1_amd64.deb
fi

sudo dpkg -i klayout_0.30.0-1_amd64.deb
sudo apt-get install -f
```

5-13. 공정 자료(PDK)

반도체 설계의 후반부 과정(레이아웃 GDS 생성)은 공장과 매우 근접되어있다. 공장에서 제공할 수 있는 기본 부품(게이트와 플립 플롭 들)과 공정상 물리적, 기하학적 특성이 제공 되어야 한다. 공장에서 제공하는 이런류의 각종

자료들을 공정 설계자료 PDK (Process Design Kit)라고 한다. 본 디자인 킷은 "내 칩 제작 서비스"의 공정 자료를 근거로 개발 되었다. 공정 자료의 원본은 아래 링크에서 받을 수 있다.

```
http://mpw.kion.or.kr
```

6. 오픈-소스 도구 설치 확인

반도체 설계를 위해 다양한 도구들을 설치하였다. "오픈-소스" 도구들은 소스를 받아 사용자가 직접 빌드하고 설치해야 한다. 준비된 예제를 돌려서 십여개에 이르는 도구들이 제대로 설치 되었는지 확인해 본다.

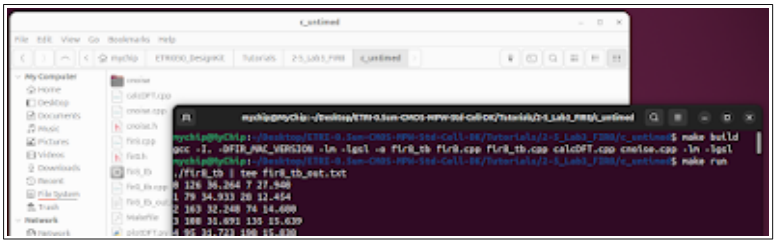
6-1. 알고리즘 개발 툴 설치 확인

C/C++ 언어로 표현한 알고리즘을 빌드하고 시뮬레이션 한다. GNU 개발도구(GCC, make 등)와 과학기술 라이브러리 (GSL, GNU Scientific Library)의 설치를 확인 할 수 있다.

예제 폴더로 이동 후 빌드,

```
$ cd ~/ETRI050_DesignKit/Tutorials/2-5_Lab3_FIR8/c_untimed
```

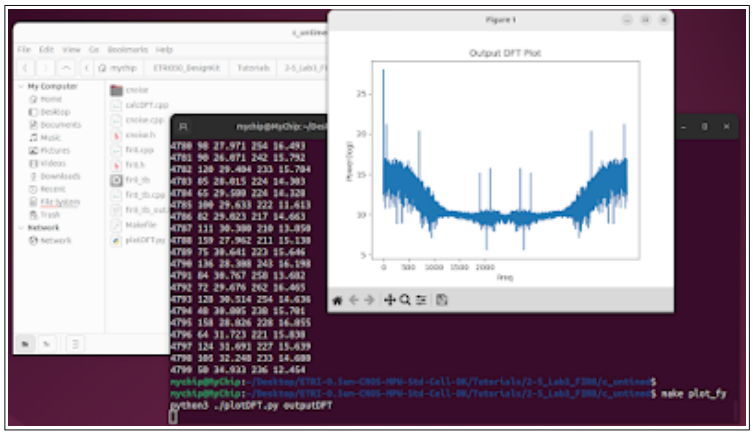
```
$ make build
```



C로 기술한 예제 fir 필터 알고리즘 실행,

```
$ make run
```

```
$make plot_fy
```

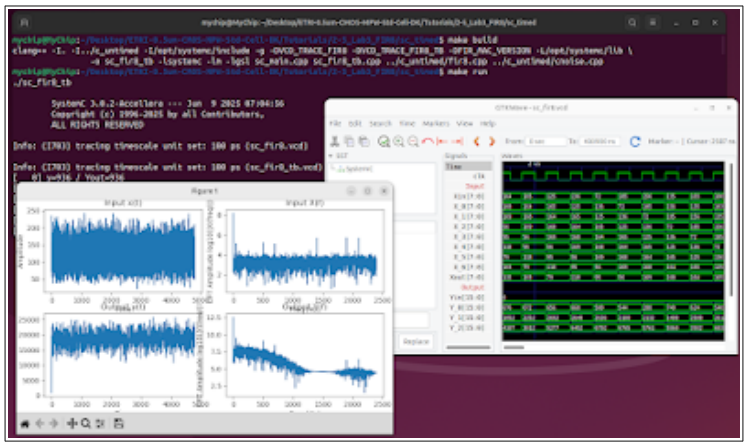


fir 필터의 타임드 테스트벤치 모델 예제 빌드,

```
$ cd ~/ETRI050_DesignKit/Tutorials/2-5_Lab3_FIR8/sc_timed
```

```
$ make build
```

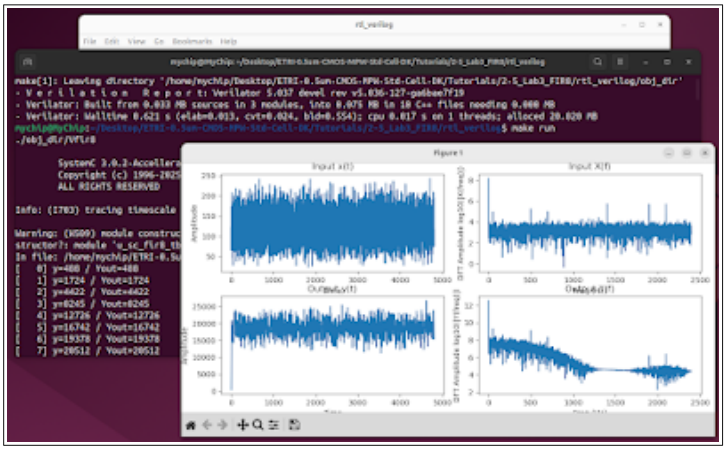
```
$ make run
```



RTL 베릴로그 모델 빌드와 실행,

```
$ cd ~/ETRI050_DesignKit/Tutorials/2-5_Lab3_FIR8/rtl_verilog
$ make build
$ make run
$ make plot
```

이래와 같은 화면을 보게되면 C/C++, SystemC, Verilator, Python 그리고 GSL 의 설치를 확인 할 수 있다.

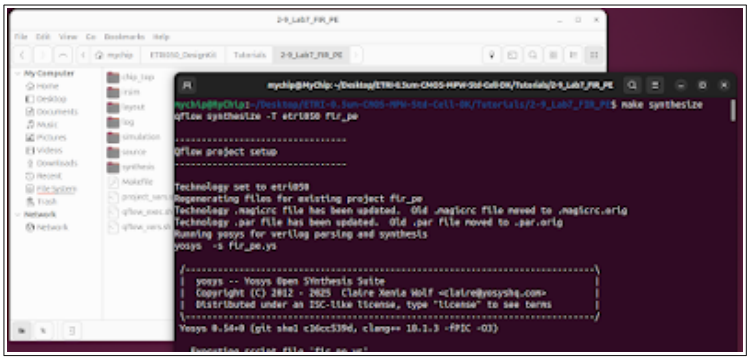


6-2. RTL 합성 및 레이아웃 도구의 설치 확인

베릴로그 RTL 을 합성하고 자동 배치와 배선 그리고 레이아웃 도구의 설치를 확인한다. 아울러 LVS와 겹친 비아 검사 도구등 사인-오프 도구들도 확인한다.

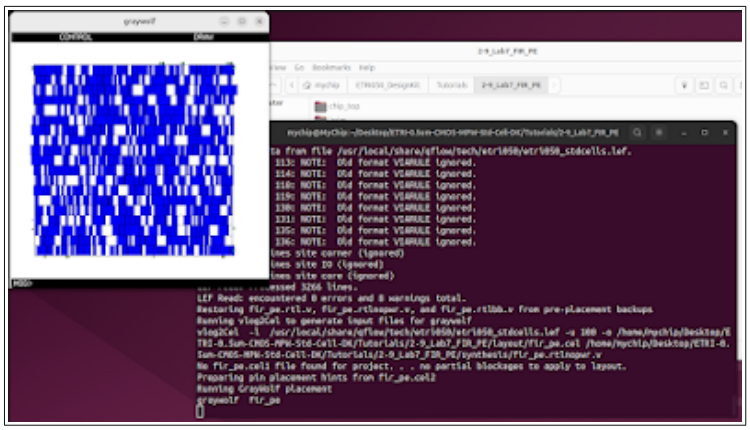
예제 디렉토리로 이동 후 RTL 합성,

```
$ cd ~/ETRI050_DesignKit/Tutorials/2-9_Lab7_FIR_PE
$ make synthesize
```



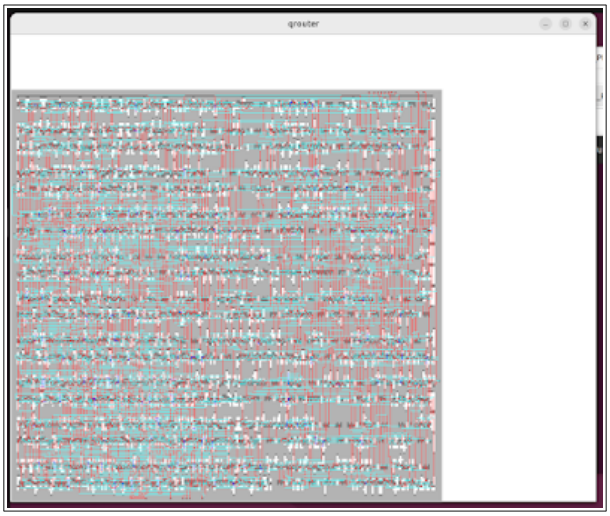
자동 배치,

```
$ make place
```



자동 배선,

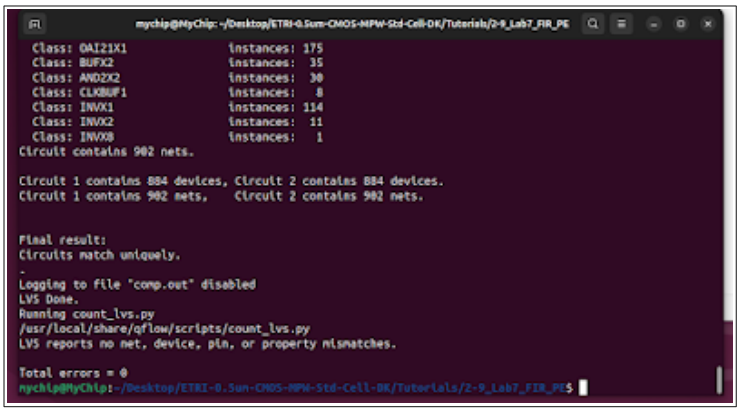
```
$ make route
```



표준 셀 병합 후 LVS,

```
$ make migrate
```

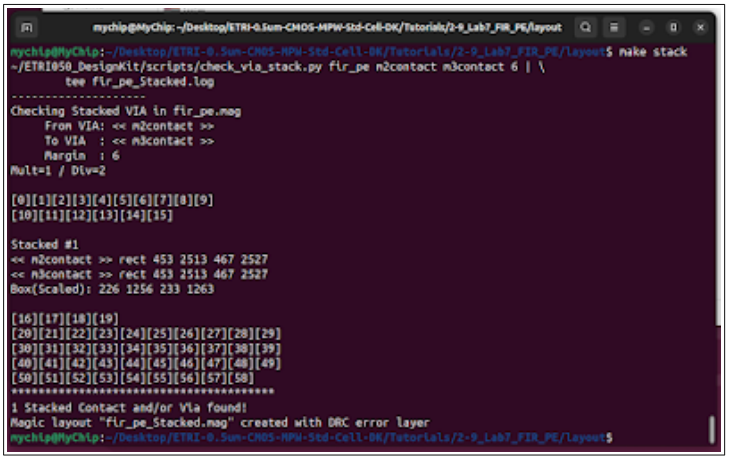
```
$ make lvs
```



자동 생성된 레이아웃 디렉토리로 이동 후 겹친 비아(Stacked VIA) 검사,

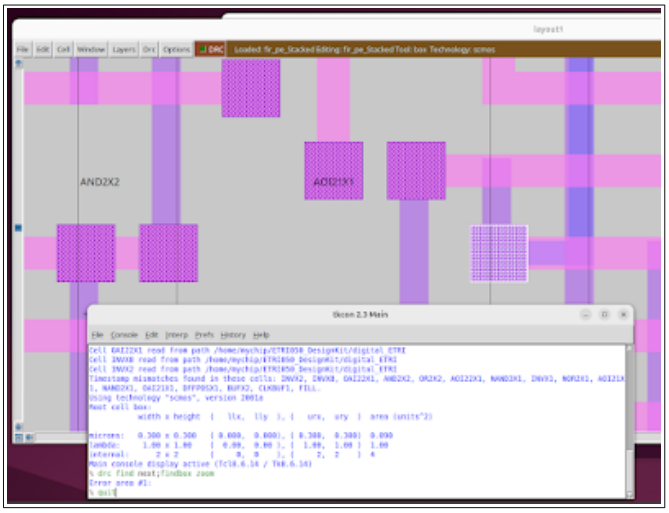
```
$ cd layout
```

```
$ make stack
```



검사 결과 1개의 겹친 VIA 가 검출 되었다. 이 겹침은 레이아웃 툴을 사용하여 겹친 위치를 찾아 수정해 주어야 한다.

```
$ magic -d XR fir_pe_Stacked
```

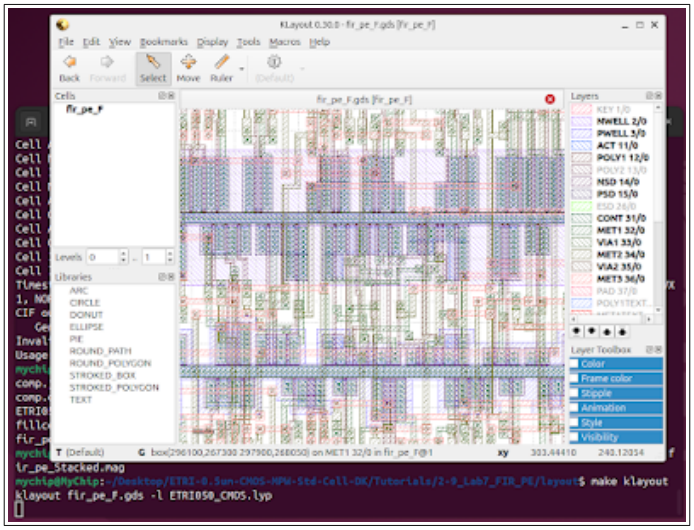


지금은 툴의 설치를 확인 하는 과정이므로 수정은 생략하고 magic에서 빠져 나온다. 이어 magic의 레이아웃을 GDS 로 생성해 보자.

```
$ make mag2gds
```

Klayout 으로 GDS 를 살펴본다.

```
$ make klayout
```



오픈-소스 반도체 설계 도구의 설치가 완료 되었다. 이제 "내 칩"을 설계할 모든 준비가 됐다.