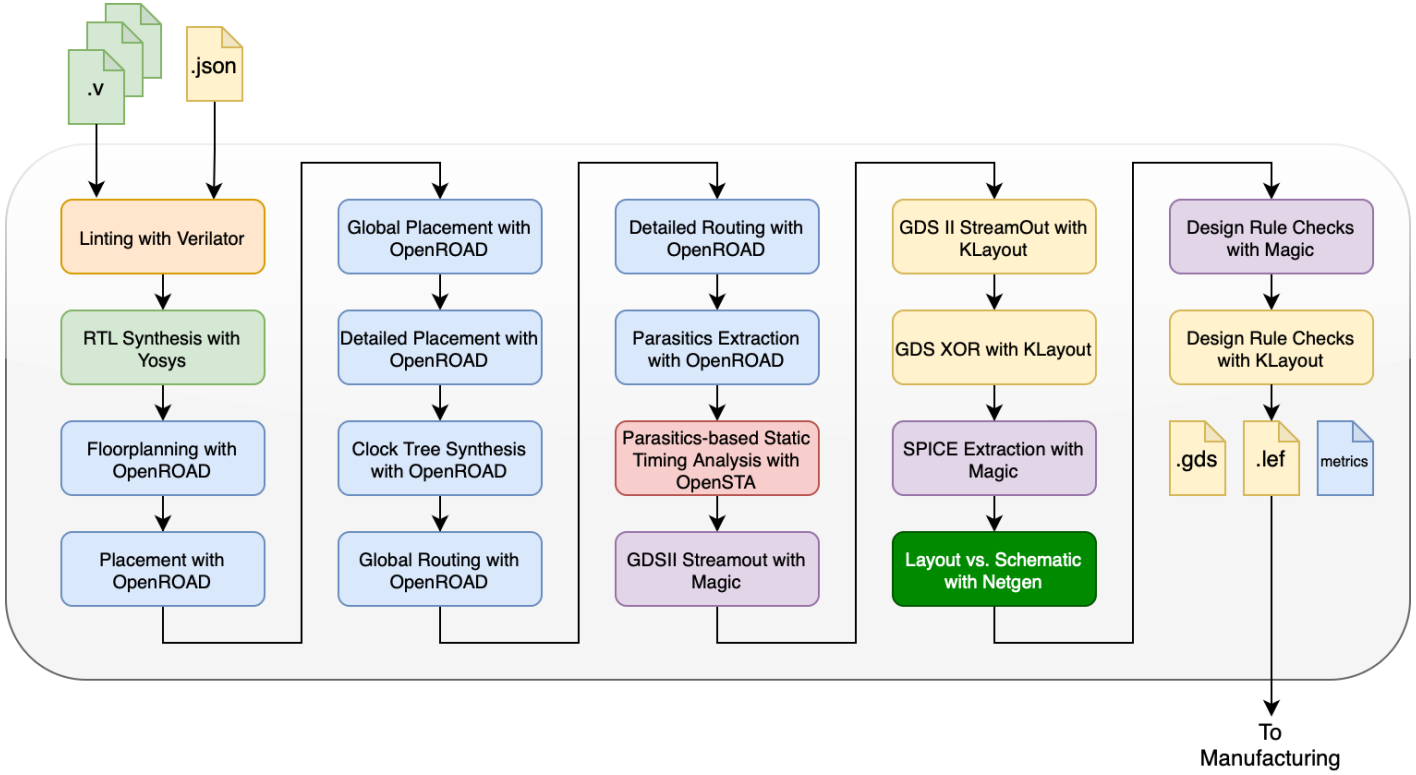
Company : Monk9 Tech pvt ltd , Rajkot -Gujarat

By: Riya.Soni (Team-Leader)

Subject : "OpenLane: Automated RTL-to-GDSII Flow for Digital Chip Design"

OpenLane Architecture



OpenLane Flow Stages

- **1. Synthesis:**
- Yosys: Translates your Verilog code (RTL) into a gate-level netlist using standard cells.
  - Think of it as converting your design logic into basic building blocks.
- OpenSTA: Verifies the timing of the synthesized netlist to ensure it works within the required clock speed.

- **2. Floorplanning :**
- Core Area Setup (Initialize Floorplan): Defines the chip's physical area, rows for placing cells, and tracks for routing wires.
- IO Placement (OpenLane IO Placer): Arranges the input/output ports on the edges of the chip.
- Power Network (PDN Generator): Creates a network of power and ground rails to supply voltage to all components.
- Tapcells (Tapcell Generator): Adds special cells to maintain electrical stability across the chip.

3. Placement

- Global Placement (RePlace): Determines approximate positions of the cells on the chip.
- Optimization (Resizer): Improves timing, area, or power by resizing or buffering cells.
- Legalization (OpenDP): Adjusts cell positions to ensure they fit the rows properly without overlap.

4. Clock Tree Synthesis (CTS)

- Clock Distribution (TritonCTS): Builds a clock tree network to deliver the clock signal evenly to all parts of the chip.

•5. Routing

- Global Routing (FastRoute): Plans an overall path for wires between cells.
- Detailed Routing (TritonRoute): Converts the plan into exact wire paths and ensures no conflicts or overlaps.
- SPEF Extraction (OpenRCX): Extracts parasitic information (resistance and capacitance) from the routed design for timing checks.

•6. Tapeout

- Magic/KLayout: Converts the routed design into a GDSII file (the standard format for chip manufacturing).
    - KLayout serves as a backup.

•7. Signoff

- DRC Checks (Magic): Ensures the layout follows all design rules required by the fabrication process.
- Antenna Checks (Magic): Checks for antenna effects that can damage the chip during manufacturing.
- LVS (Netgen): Verifies that the physical layout matches the original schematic.

How to Think of the Flow :

1. **Write the Design** → Synthesis converts it to gates.
2. **Prepare the Chip Layout** → Floorplanning & Placement arrange the design.
3. **Add Essential Components** → Clock tree and routing connect everything.
4. **Finalize the Design** → Tapeout prepares it for manufacturing.
5. **Check Everything** → Signoff ensures it’s ready to fabricate.

Installtion package Required

- **1. Prerequisites**
- **System Requirements**:
  - Linux-based OS (e.g., Ubuntu 20.04+ recommended)
  - At least 16GB RAM and 100GB of free storage.

- **Install Dependencies**:
  - `sudo apt-get update`
  - `sudo apt-get install -y build-essential python3 python3-venv python3-pip git`

- **Install Docker**:

…….CONTINUE……

```
# Remove old installations
sudo apt-get remove docker docker-engine docker.io containerd runc

# Installation of requirements
sudo apt-get update
sudo apt-get install \
  ca-certificates \
  curl \
  gnupg \
  lsb-release

# Add the keyrings of docker
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

# Add the package repository
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update the package repository
sudo apt-get update
```

# Install Docker
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# Check for installation
sudo docker run hello-world

A successful installation of Docker would have this output:
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

- sudo reboot # REBOOT

Checking the Docker Installation:
After that, you can run Docker Hello World without root. To test it use the following command:

```
# After reboot
docker run hello-world
```

Checking Installation Requirements:
In order to check the installation, you can use the following commands:

```
git --version
docker --version
python3 --version
python3 -m pip --version
make --version
python3 -m venv –h
```

Download and Install OpenLane:
Download OpenLane from GitHub:

```
git clone --depth 1 https://github.com/The-OpenROAD-Project/OpenLane.git
cd OpenLane/
make
make test
```

 # Successful test will output the following line:

Basic test passed

#Enter into docker session :

make mount

 # Starting the OpenLane Environment:

cd OpenLane/

make mount

#Running the flow:

./flow.tcl -design <example>

#Creating new designs:

./flow.tcl -design <example> -init_design_config –add_to_designs

#This will create directory:

designs/<example>

├── config.json

```
designs/<example> ├── config.json  ├── pin.cfg
├── src
            ├── example.v
```