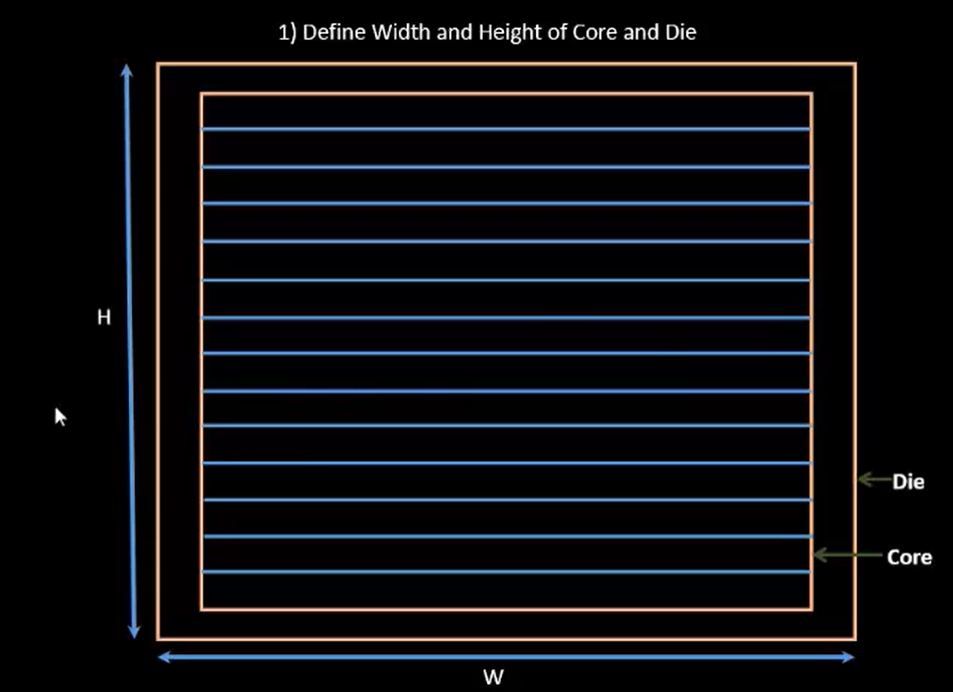
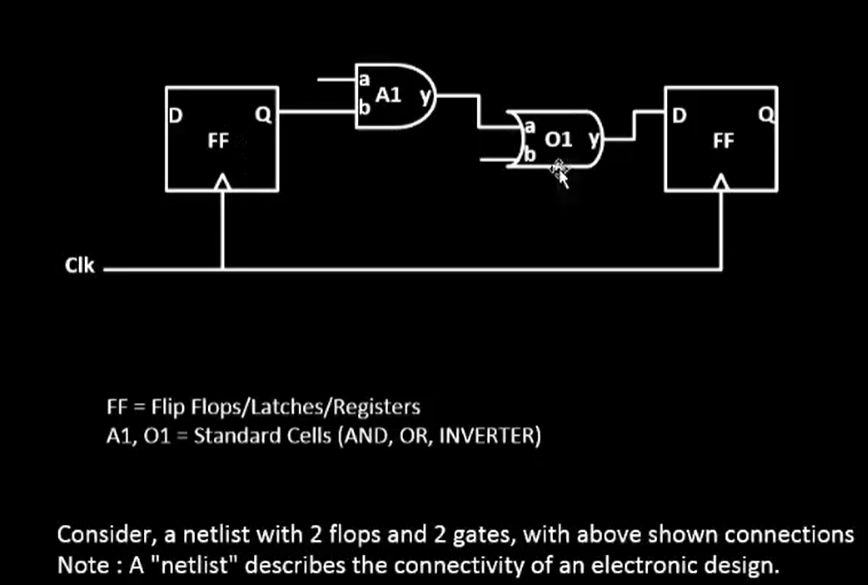
**Section 2 : Good floorplan vs bad floorplan and introduction to library cells**

**Theory: FLOORPLAN**

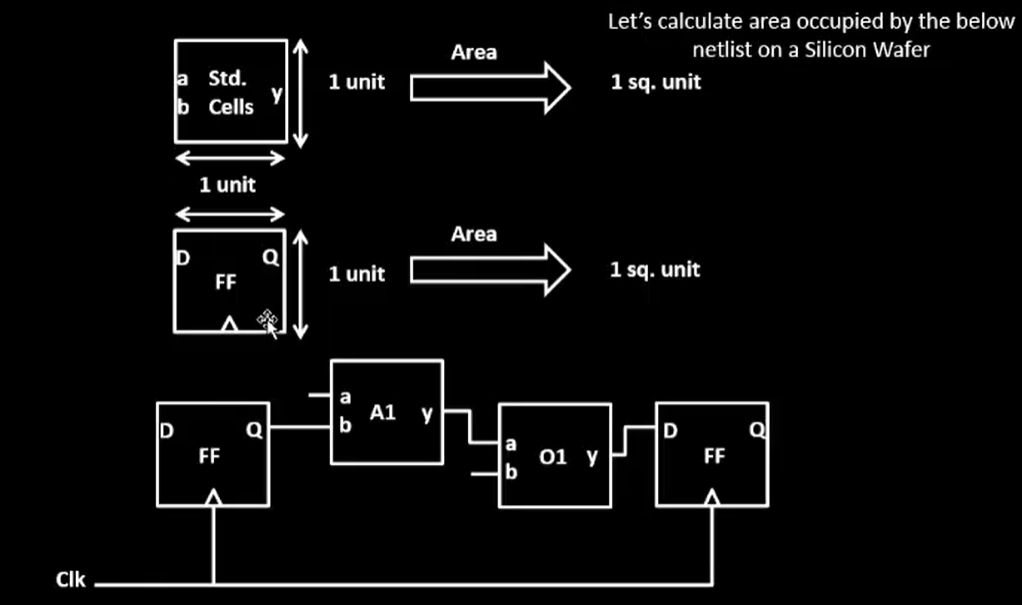
1. How to calculate width, height of core and die?



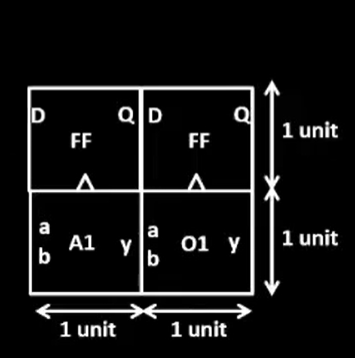
* Consider a simple netlist consists of and,or gate and flipflops



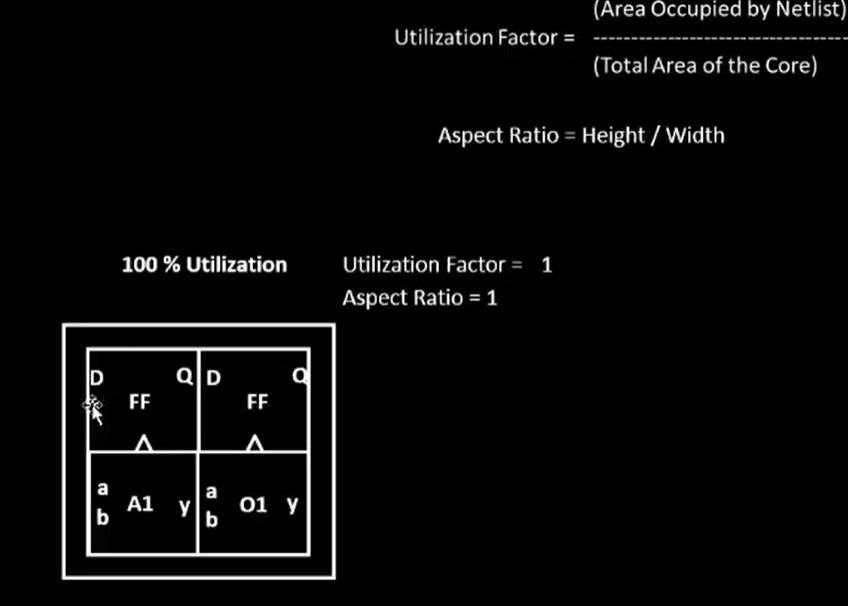
* Calculate the area of standard cells and flipflop first and then cal the total core area



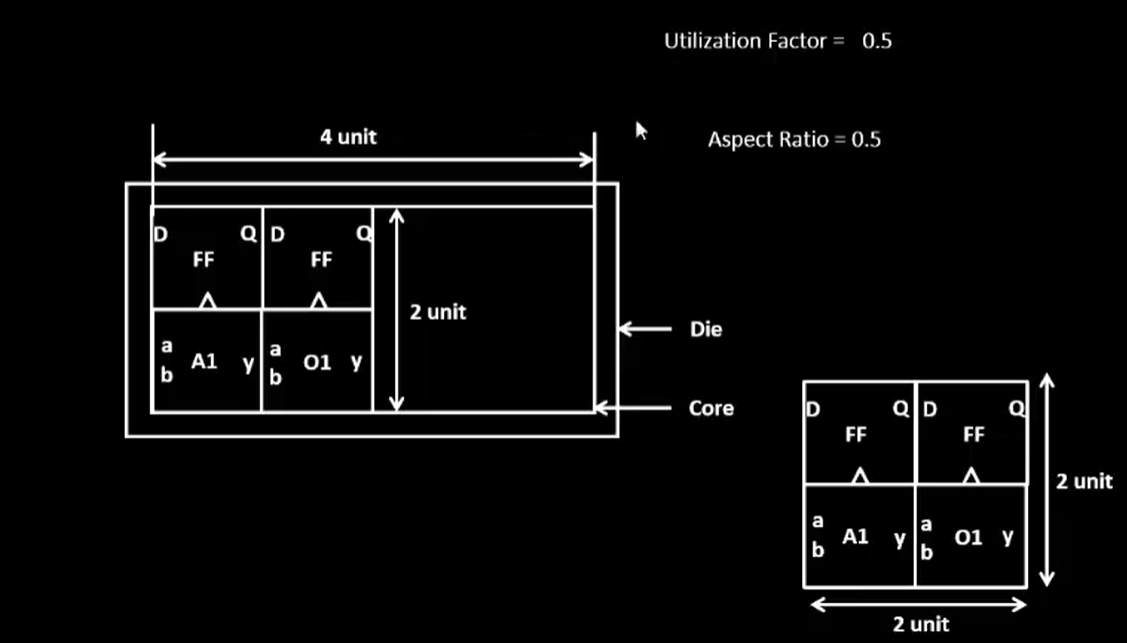
* Arrange the cells together and calculate the area



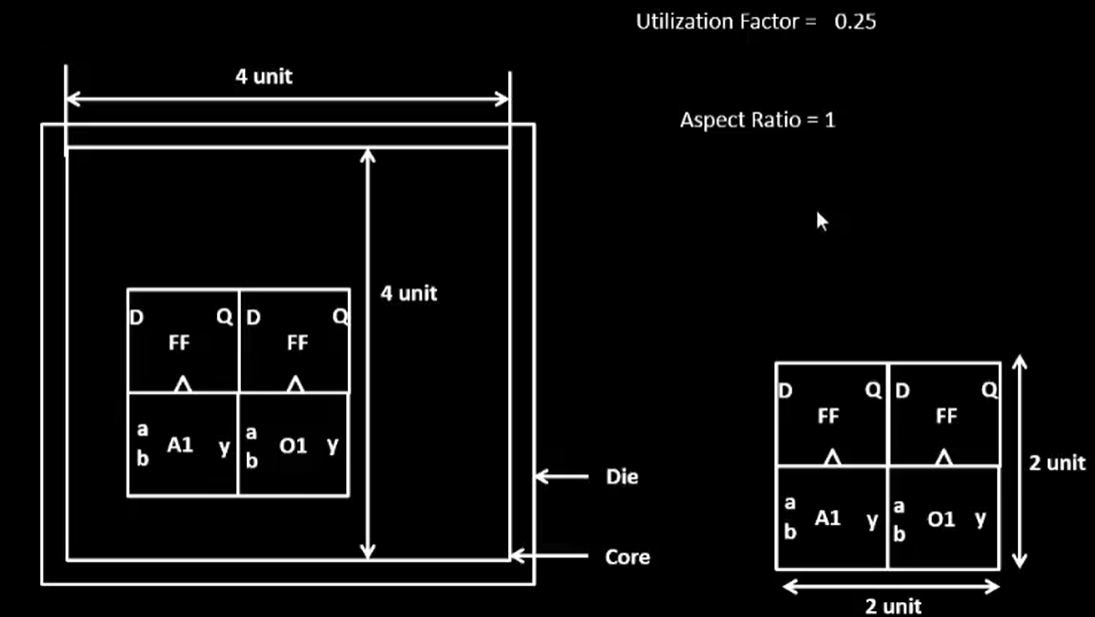
* Place the netlist inside the die, calculate the utilization, utilization factor, aspect ratio.
* If the utilization factor is 1 then the core is square shape, if aspect ratio is less than 1 then it is rectangle, has more space to add extra cells.



* For example, take rectangle core area and place the netlist and find the utilization factor and aspect ratio, it must be less than 1.

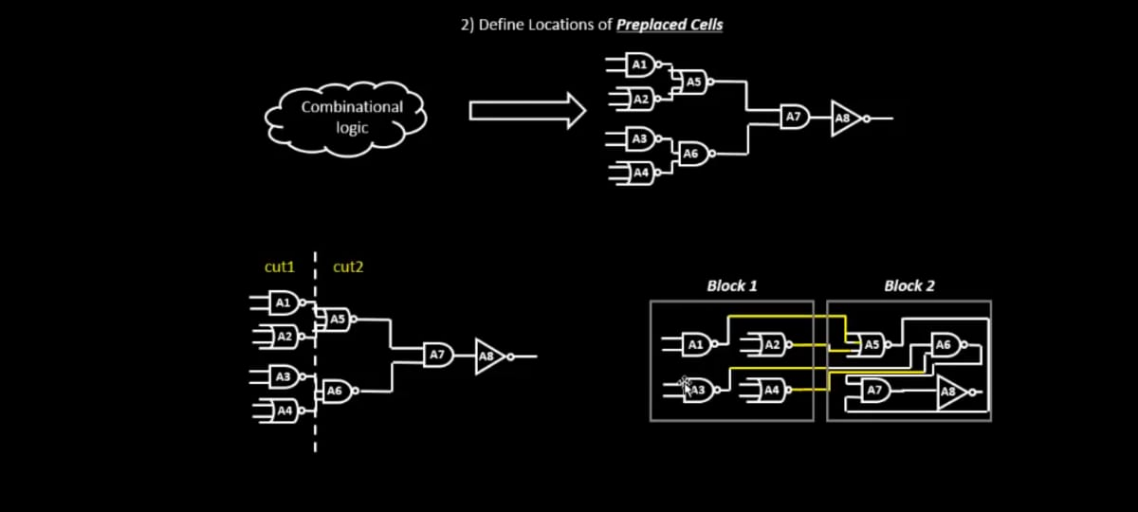


* If the core area has width of 4 units and height of 4 units, calculate the utilization and aspect ratio.
* Here, the aspect ratio is 1, it has a square core area.
* And utilization factor is 0.25, that means 25% of core area is occupied and 75% area is pending.



1. **Defining the location of preplace cells?**

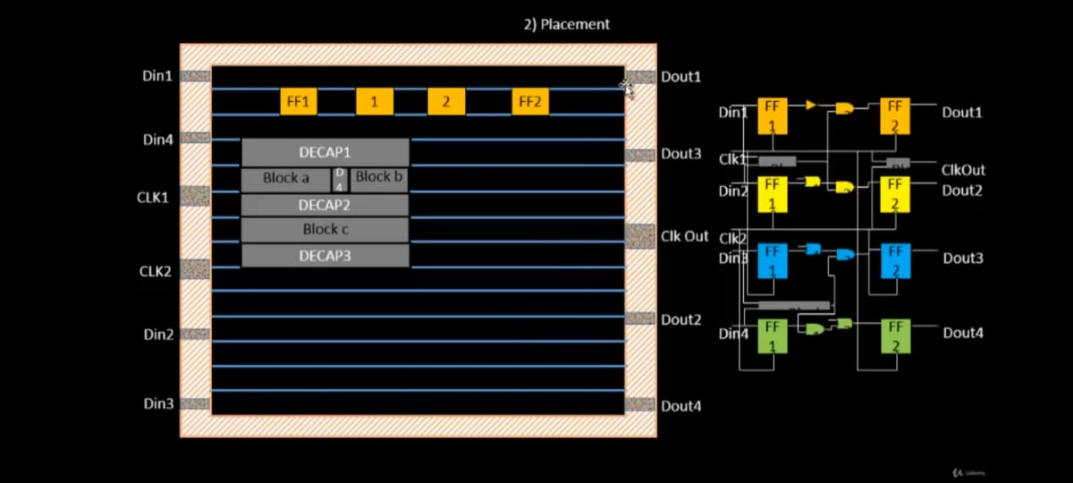
What is preplace cells?



* If there’s a combinational block of 100 gates, then we’ll cut them and made 2 different blocks and implement them separately.
* Extend the IO pins and black box the 2 different blocks, so that inside circuitry is invisible from the top view.
* By this, each block can implemented easily called IP’s and reused many times.
* Other IP’s are memory, clock gating cell, comparator, mux.
* These cells functionality is implemented only once and called preplace cells because these are placed once in a chip.
* Arrangement of these IP’s in a chip is referred as Floorplanning.
* These IPs/blocks have user defined locations and hence are place in chip before automated placement and routing(placing remaining logic cells onto chip) and are called as pre-placed cells.

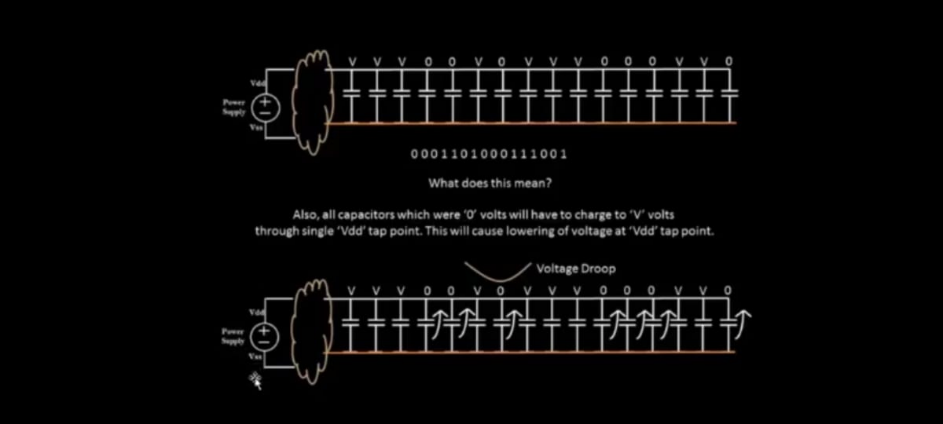
**location of preplace cells:**

* location of preplace cells are such that, once placed cant be moved.
* These cells are placed near the input pins.

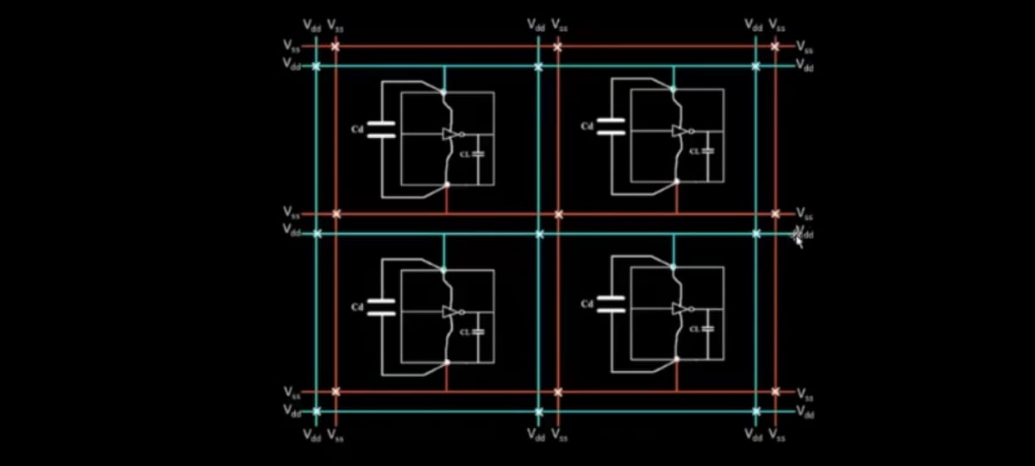
**Surround the preplaced cells with Decoupling capacitors.**

* When there is large physical distance from the main supply to the circuit, due to the presence of more number of R,I there’ll be more voltage drop across this physical distance.
* The amount of voltage delivered to the circuit may not be equal to supply voltage, it’ll be less
* But if the voltage is within the Noise margin then its fine, its not in noise margin it’ll effect the output.
* To minimise this, the decoupling capacitor(voltage level is very much equal to supply voltage) are placed near to the circuit, these capacitor will charge to supply voltage level and deliver the voltage to the circuit.

**The global issue is related to power supply.**



* If there is only once power supply, that has to fed to many cells, where the vdd becomes low (as many cells drains input or charges) and vss will start to increases( create a ground bound) as it discharges many power.
* Solution for this is to increase the number of vdd and vss(power sources).where it charges or dump its charge to its nearest vdd or ground.



**Pin Placement:**

* Order of pin placement is random.
* Pads are only for pin placement, and are blocked using blockage for other cell placement.
* Hands on experiment on floorplan: run floorplan
* See all the switches that has set in config.tcl, did it take precedence over system defaults:

**Netlist Binding with Physical cells:**

* In the netlist each logic gates and flipflops are considered as a box for the placement
* Each element / component of netlist is given a proper width and height.
* This information is present in the library also delay of particular gate is present in the library.
* Library has different flavour for each cells, cells, delay information, shape &sizes of cells and the timing information.

**PLACEMENT:**

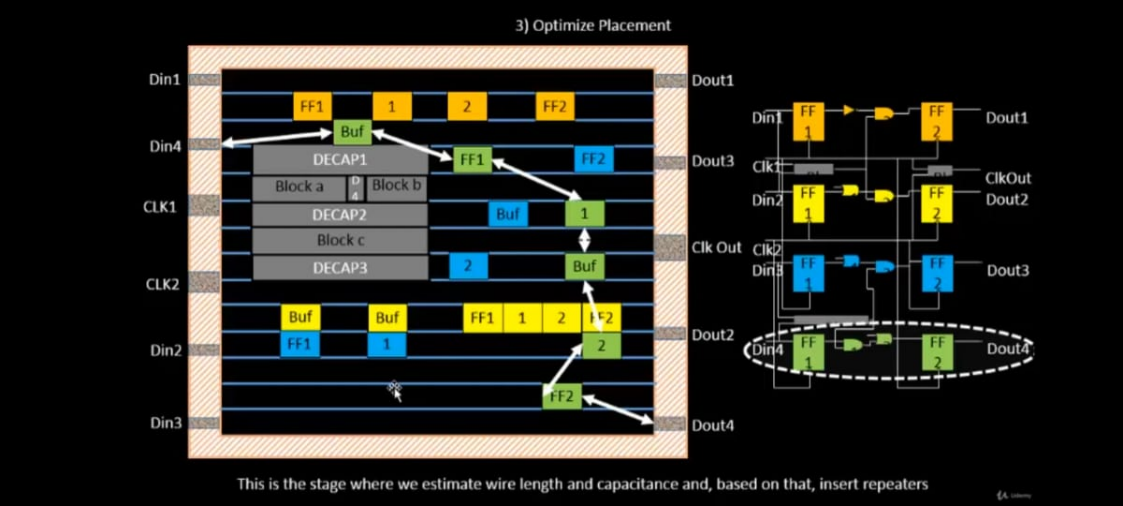
* Placement of netlist into the floorplan.
* First convert netlist into physical view of logic gates.
* Preplace cells are not affected in the placement stage.
* Place the logic close to its input output pins .
* Placement of 4 logic circuits into core area.
* Faced some problem with placing logics 3 and 4 due to preplaced block and random pin placement.
* And the solution to it is optimize placement.

**Optimize placement:**

* In this stage where we estimate wire length and capacitance and based on that insert repeaters.
* When the distance between the input pin and the flipflop is more, the resistance is more.
* In order to maintain signal integrity, we’ll add repeaters.
* Repeaters are buffers, that’ll recondition your original signal, make a new signal which replicates the original signal and send it again.
* More buffers, more area consumed.
* Based on capacitances, slew is detected. More the capacitances value, more is the slew.

**Optimization in all stages:**

* Stage 1: din 1 to Dout 1:
* Here the minimum signal integrity has maintained and no need of repeaters.
* Stage 2: din 2 to dout 2:
* Here din2 is very far from ff1, so there needs a repeater in between them, but between the flipflops and gates very minimum integrity is maintained hence no need of repeaters in between.
* Stage 3: din3 to dout 3:
* Hence the no repeaters is needed for input pin to FF1 to gate 1 and gate 1 to gate2, but buffer is needed for gate 2 to FF2, and no buffer needed for FF2 to output pin.
* Stage 4: din4 to dout 4:
* Buffered required for input pin to FF1 and also in between gate 1 to gate 2.



**Library characterization and modelling:**

**IC design Flow:**

Step 1: logic synthesis-functionality coded into RTL converted into legal hardware is called logic synthesis.

Output of logic synthesis is a arrangement of gates, that will represent our original functionality, that is described using an RTL.

Step 2: floorplan – we import the output of logic synthesis or we import netlists that we get out of logic synthesis. And decides the size of core and a die.

Stage 3: placement – placing logic cells present in netlist and place in core area, such that initial timing is met.

Stage 4: CTS- clock spreading across the all logic cells with equal time. Buffer take care that the clock signal will get equal rise and fall time.

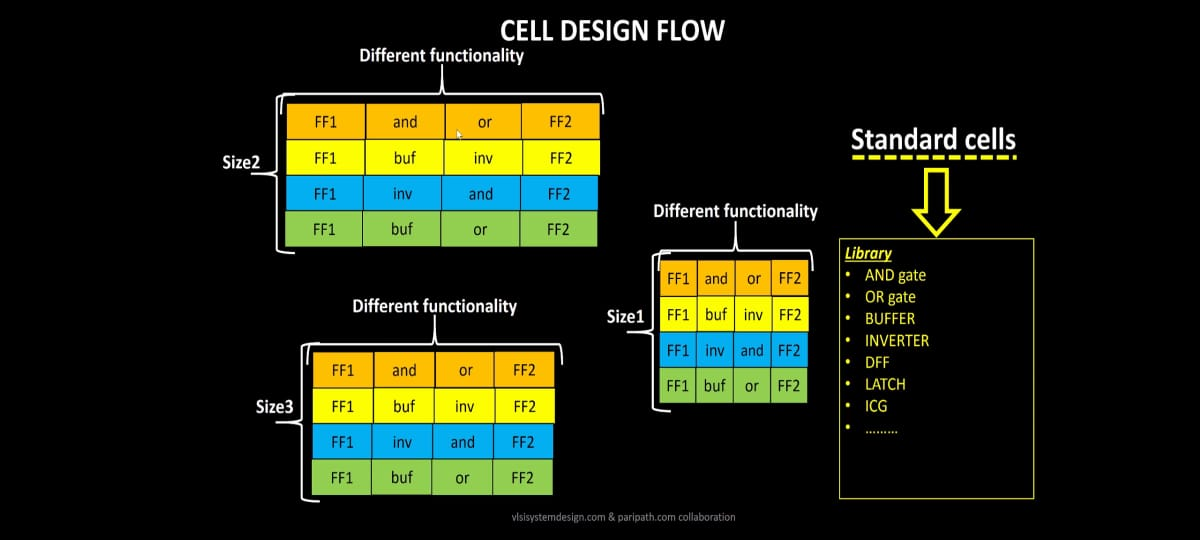
Stage 5: routeing - routeing of cells.

Stage 6:STA – what is setup time and hold time and max achievable frequency.

* In all these steps, cells are common.
* The collection of cells are called as the library.

**Inputs for Cell Design Flow**

Standard cells: Flipflop, buffer, gates.



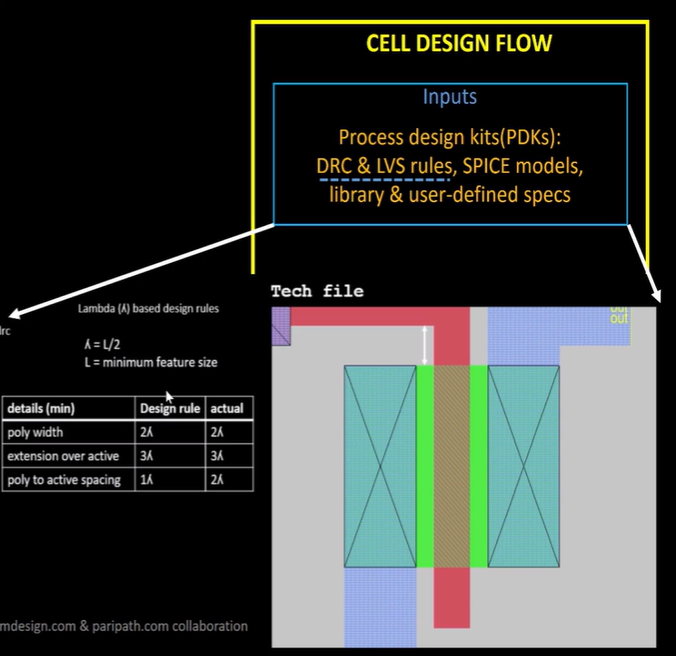
**Library consists:**

* Library is a place where we keep all the standard cells.
* Library is also a place where we keep all the Ips, Decap cells.
* It also contains physical characteristics(L,W,H) of each cells and also have different threshold voltage(hvt,lvt).

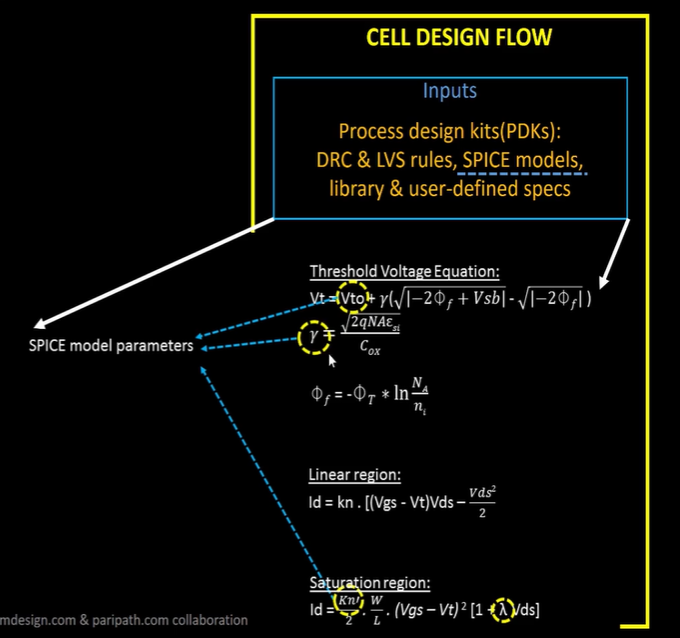
**Cell design flow of inverter:**

Cell design flow has 3 parts : inputs, design steps, outputs.

Inputs: 1. DRC and LVS



Input 2: threshold voltage( foundry parameters)



Input 3: library & user defined specifications:

Cell height – separation between power rail and the ground rail.

* Cell width – depends on timing information.

This is the information given to library developer saying, we need cells from drive strength 1-x( lower the drive strength, difficult to drive huge wire).

* Supply voltage – a typical inverter should work at particular voltage, provided by the top level. (library should consider noise margin )
* Metal layer: specification(like power, gnd, contact) that certain libraries build on certain layer.
* Pin location: library developer need decide pin location.

**2 : Design steps: 3 steps:**

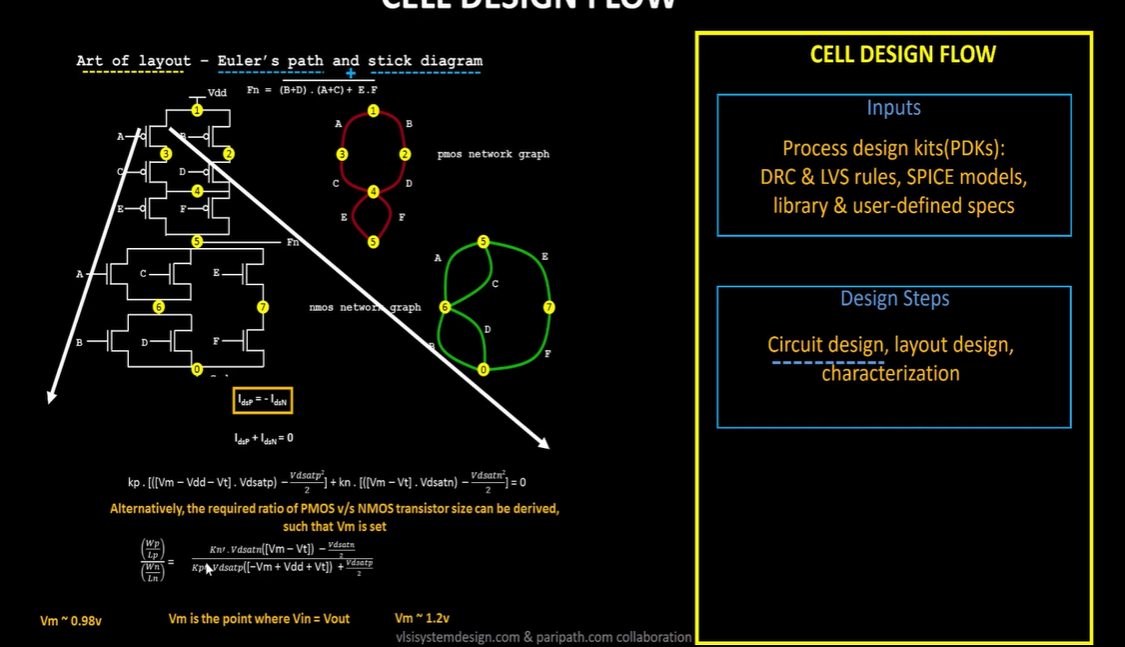
Circuit design

Layout design

Characterization

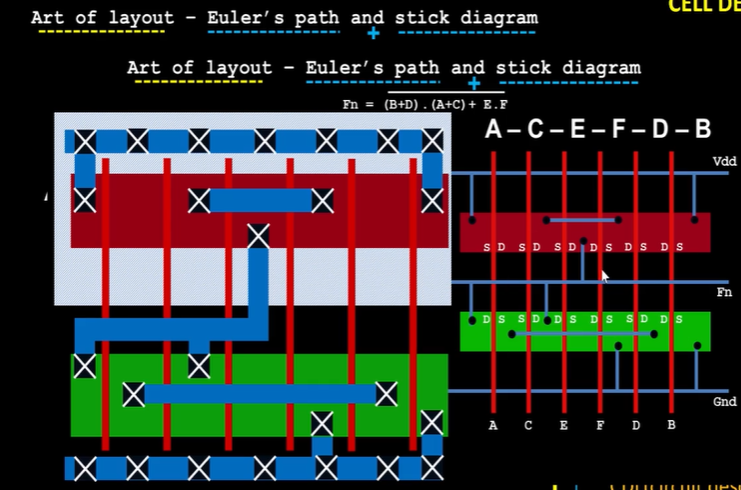
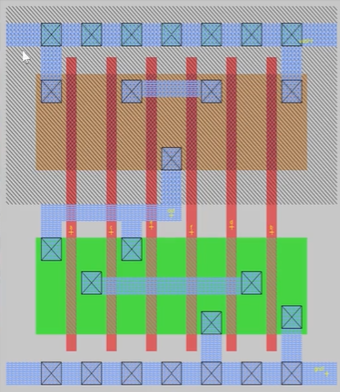
**Circuit design** : pmos and nmos needs to designed based on W/L ratios and drain current to meet the desired requirments.(spice simulation)

Output of circuit design = CDL(circuit description language)



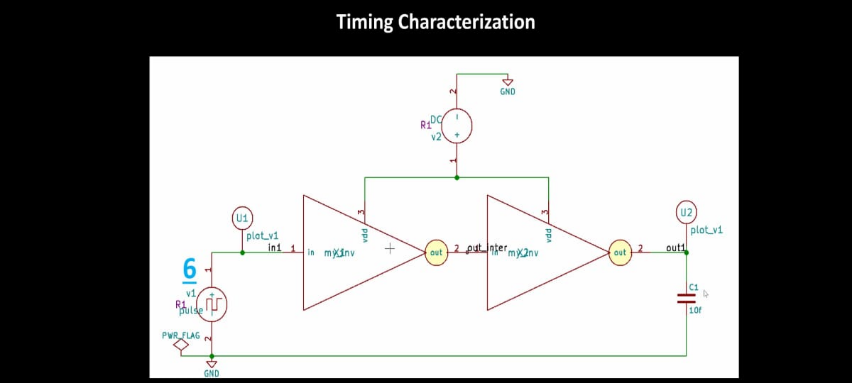
**Layout Design:**

* First step is to implement function using pmos and nmos
* Second step is to get pmos network graph and nmos network graph (euler’s path)
* From that draw the stick diagram.
* Using DRC rules which we got from input and stick diagram, design a layout.
* The layout derived from this, gives the best performance and best area.
* Final step is to extract the parasitic(R,C) out of layout and characterise in terms of timing.
* Output of layout = GDSII ,LEF, extracted spice netlist(.cir)
* Output of characteristics is timing , noise, power .libs, function.

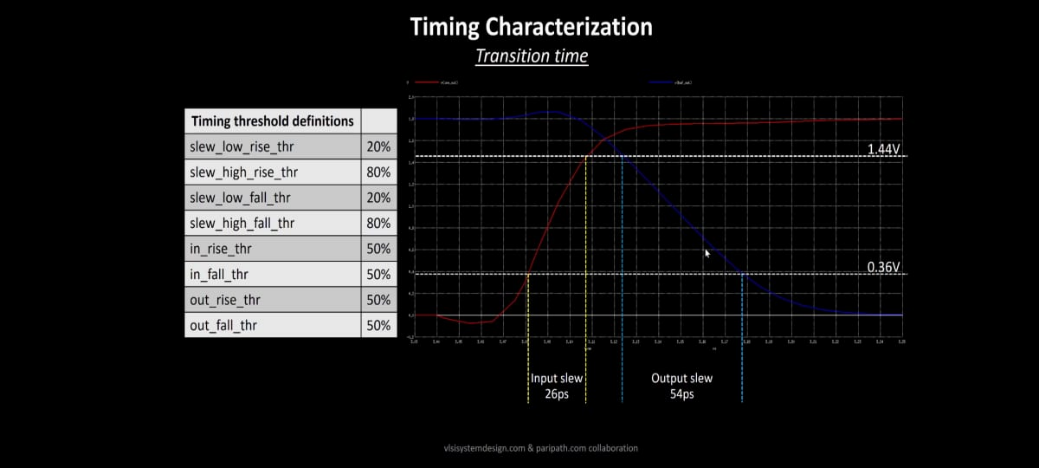
 

**Characterization flow:**

* Has nmos and pmos (spice models) characteristics, like doping concentration, threshold voltage, tox etc,
* Contains resister and capacitor value of buffer.
* Step 1: read the model files (nmos, pmos model)
* Step 2: read extracted spice netlist.
* Step 3: recognize the behaviour of buffer.
* Step 4: read sub circuit of inverter.
* Step 5: attach necessary power sources.
* Step 6: apply the stimulus(i/p).
* Step 7: provide output capacitor.
* Step 8: necessary simulation command. ( eg: trans, dc, ac).
* Feed in all these 1-8 inputs in a form of configuration file to GUNA (characterization software), this software produces timing , noise, power .libs, function model.



**Timing Characterization:**



* Slew\_low\_rise\_thr = is near 0 volt
* Slew\_high\_rise\_thr = is near vdd
* Slew\_low\_fall\_thr = is near 0 volt
* Slew\_high\_fall\_thr = is near vdd
* In\_rise\_thr
* In\_fall\_thr
* Out\_rise\_thr(buffer)
* Out\_fall\_thr(buffer)

**Propogation delay:**

Delay = out time – in time

**time(Slew\_high\_rise\_thr) – time(Slew\_low\_rise\_thr)**

* When there is a movement of output graph, it caused negative delay.
* Negative delay is caused by choosing wrong threshold point.
* Even if the threshold value is correct, but it has huge wire delay. Then it’ll lead to negative delay.

