# VPR: Placement and Routing for FPGAs

## An Introduction to "compilation" for FPGAs

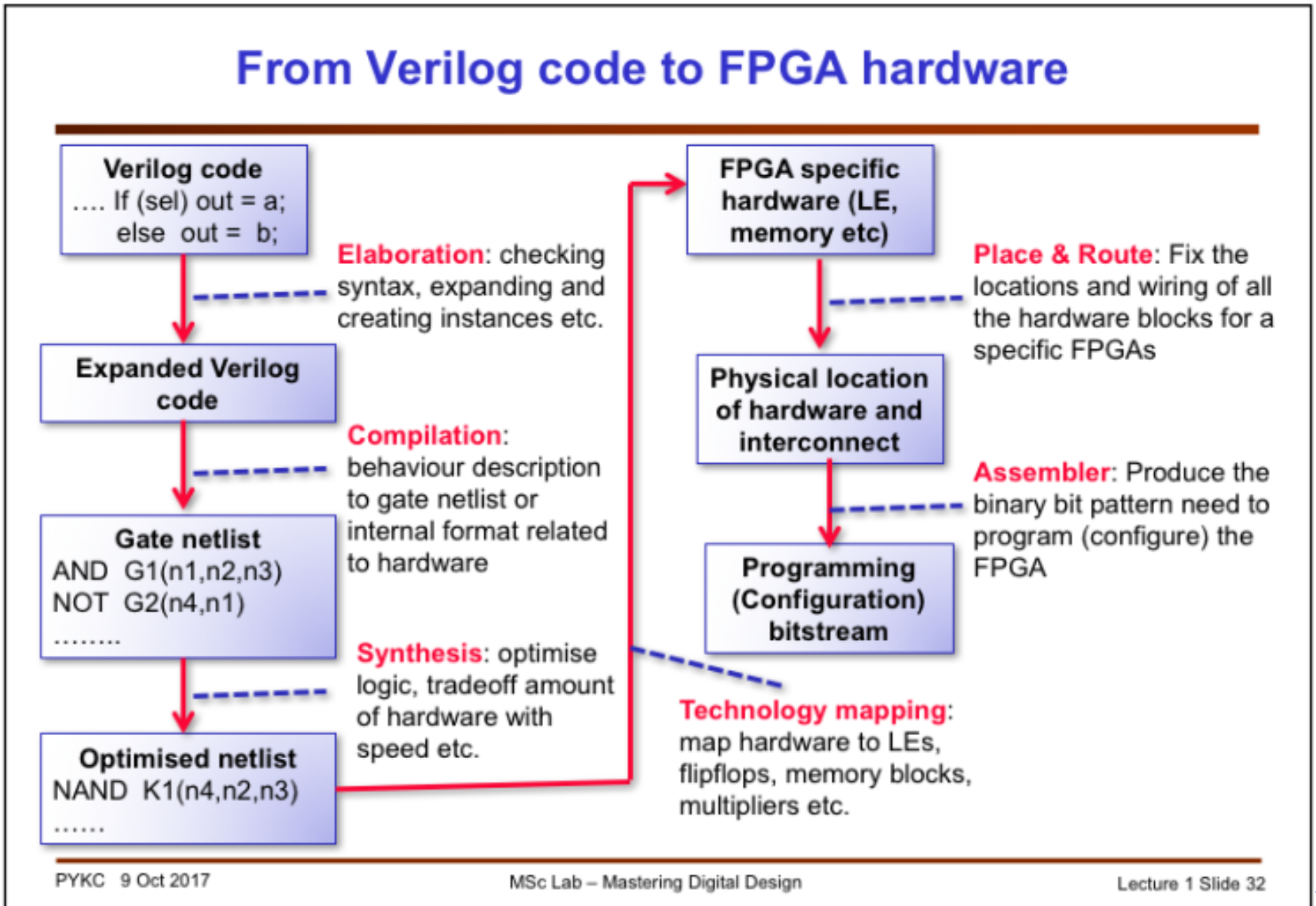
Shreeyash Pandey

Vicharak

5th November, 2023

# Contents

# Compilation flow for verilog



# Synthesis

## Register-Transfer level

```
module my_module (in1, in2, out);
  input in1;
  input in2;
  output out;
  assign out = in1 & in2;
endmodule
```

## Gate Level

```
module my_module (in1, in2, out);
  input in1;
  input in2;
  output out;
  and g1(out, in1, in2);
endmodule
```

# Synthesis

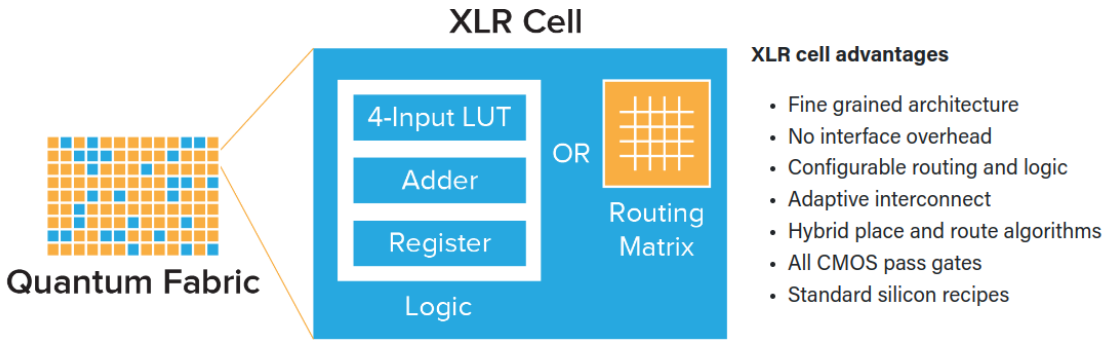
## Register-Transfer level

```
module counter ( clk, reset, enable, dat_out );
output [15:0] dat_out;
input clk, reset, enable;
wire N69, N70, N71, N72, N73, N74, N75, N76,
     N83, N84, N86, N88, N90, N92, N94, N96,
     n48, n49, n50, n51, n52, n53, n54, n55,
     n62, n63, n64, n65, n66, n67;

     ...
  NAND3X1AD U65 ( .A(dat_out[12]), .B(n51), .C(dat_out[13]),
  NOR2X1AD U66 ( .A(n52), .B(n43), .Y(N82) );
  XOR2X1AD U67 ( .A(n53), .B(dat_out[13]), .Y(n52) );
  NAND2X1AD U68 ( .A(dat_out[12]), .B(n51), .Y(n53) );
  INVX1AD U100 ( .A(reset), .Y(n43) );
endmodule
```

# Synthesis

## What does an FPGA look like on the inside? and How efinix does it




- "Quantum" architecture that uses a XLR (exchangeable routing and logic) cell
- Dynamic configuration of cell to be a switch or a logic block
- This ratio of logic cells to routing blocks is critical because too few routing resources means that it is impossible to connect the logic elements and the FPGA is underutilized. Too many routing resources means that silicon area is wasted, driving up chip costs.

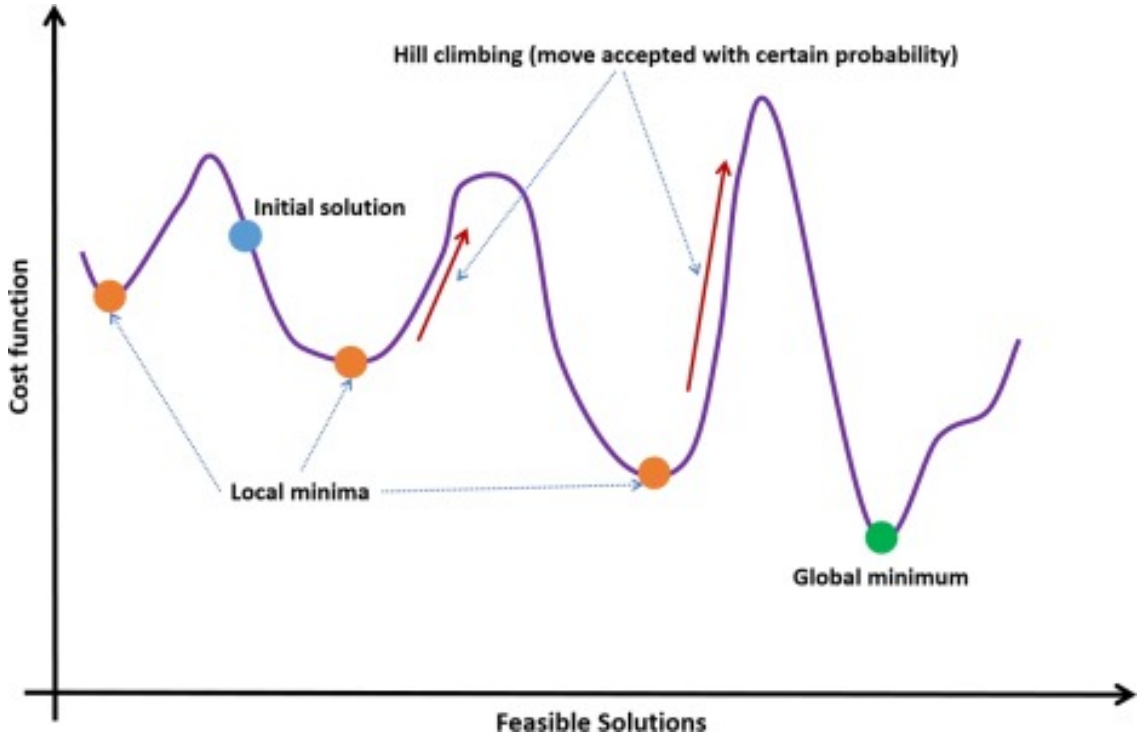
https://www.efinixinc.com/technology.html

# Placement

- Placement determines which logic blocks should implement required logical function
- Goal of the placement is the optimize closeness of connected blocks.
- Simulated annealing is one of the algorithms that helps us decide.
- Borrowed from metallurgy

# Simulated Annealing

- How to find a global maximum of a function?
- Simulated Annealing is a stochastic process.
- Happens in multiple passes



# Simulated Annealing

```
S = RandomPlacement ();
T = InitialTemperature ();
R_limit = InitialR_limit ();

while (ExitCriterion () == False) {          /* "Outer loop" */
    while (InnerLoopCriterion () == False) {     /* "Inner loop" */
        S_new = GenerateViaMove (S, R_limit);
        ΔC = Cost (S_new) - Cost (S);
        r = random (0,1);
        if (r < e^(-ΔC/T)) {
            S = S_new;
        }
    }            /* End "inner loop" */
    T = UpdateTemp ();
    R_limit = UpdateR_limit ();
}    /* End "outer loop" */
```

**FIGURE 2.9** Pseudo-code of a generic simulated annealing-based placer.

# Simulated Annealing

- Create initial placement randomly.
- Select a logic block at random and move it somewhere else
- Compare new and old placement with a **cost function**
- if cost decreses, accept the move
- if cost increses, keep going
- even if the cost increases, there is a chance of the move being accepted, given by a probability

$$e^{-\delta C/T}$$

- T (Temperature) is a dynamic parameter of the algorithm
- Annealing schedule determines the rate at which T changes, InnerLoopCriterion and ExitCriterion.

# Cost Function

$$Cost = \sum_{n=1}^{N_{nets}} q(n)\left[\frac{bb_x(n)}{C_{av,x}(n)} + \frac{bb_y(n)}{C_{av,y}(n)}\right]$$

- Linear congestion cost function
- bbx and bby are horizontal and vertical spans of its bounding box
- q(n) is the compensation factor for wire length
- Cavx and Czvy are average channel capacities
- penalizes placements that require more routing in narrower channels (due to Cav* denominators)

# Annealing Schedule

- Annealing schedule determines the rate at which T changes, InnerLoopCriterion and ExitCriterion.
- Good annealing schedule leads to better quality placments
- To determine the initial temperature, N blocks are placed randomly and a Standard Deviation for N samples is calculated. The initial temperature is set to 20x this Standard deviation.
- High temperature $\rightarrow$ High chance of acceptance.
- Temperature is calculated with

$$T_{new} = \alpha * T_{old}$$

- $\alpha$ depend on $R_{accept}$ (fraction of moves accepted at the previous temperature)
- The anneal is terminated when

$$T < 0.005(Cost/N_{nets})$$

# Routing

- Patfinder negotiated congestion algorithm
- Initially route by finding the shortest path even if overused
- In later iterations, rip up the routed nets and re-route based on lowest cost path.
- Cost is a function of current overuse and any overuse that occured previously.
- Uses Djikstra's algorithm to find the shortest path
- For k terminal net, it is invoked k-1 times. A path from source to sink is found for the first wavefront and it is emptied. Similarly, a path for all the k-1 terminals are found.
- This has high cost for high fan-out nets

# Conclusions

- VPR is better than other PNR tools etc. is not interesting.
- We draw a different conclusion from what we've gathered
- Efinix uses VPR for its place and route, but does not provide a direct interface to access it.
- Is it possible to hook VPR optimizations suitable for our designs into Efinity?
- Is this the intersection b/w manually placing and allowing the synthesizer full control of our algorithm?