Chapter 2 : ASIC Synthesis

- ASIC Design Flow
- Chip Synthesis Process
- Levels of Synthesis

ASIC Design Flow

What is an ASIC?

- An ASIC is an application-specific integrated circuit .

  Most of ASICs use a library of pre-designed and pre-characterized logic cells. In fact, we could define an ASIC as a design style that uses a cell library rather than in terms of what an ASIC is or what an ASIC does.

Types of ASIC

- Full-Custom : Analog/digital with all customized mask layers and some logic cells.
- Semi-custom :
  - Cell-based – all customized mask layers
  - Masked gate array – some customized mask layers
- Programmable :
  - Field-programmable gate array (FPGA) – no customized mask layers
  - Programmable logic device (PLD) – no customized mask layers

Cell-Based Design Flow

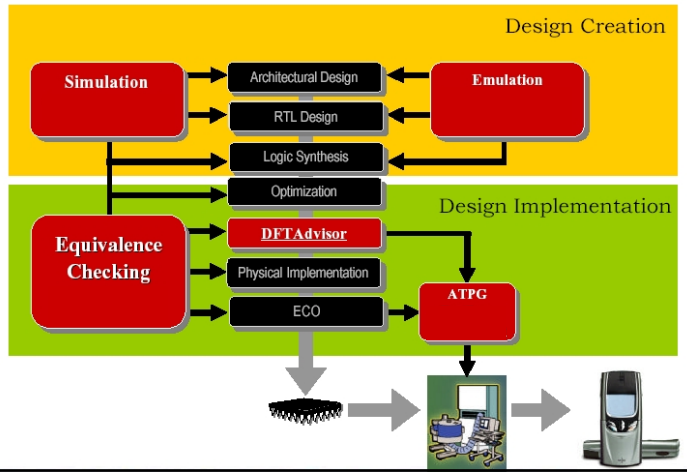
- Spec Development
- HDL (RTL) Coding; Simulation;
- Preliminary Synthesis
- Preliminary Floorplanning; Synthesis
- Design for testability; Test Pattern Generation
- Pre-layout Simulation
- Layout (Floorplanning, Placement, Routing)
- Post-layout Simulation; Static Timing Analysis
- ECO
- Layout Verification
- DRC, ERC, LVS, Antenna, Metal Density

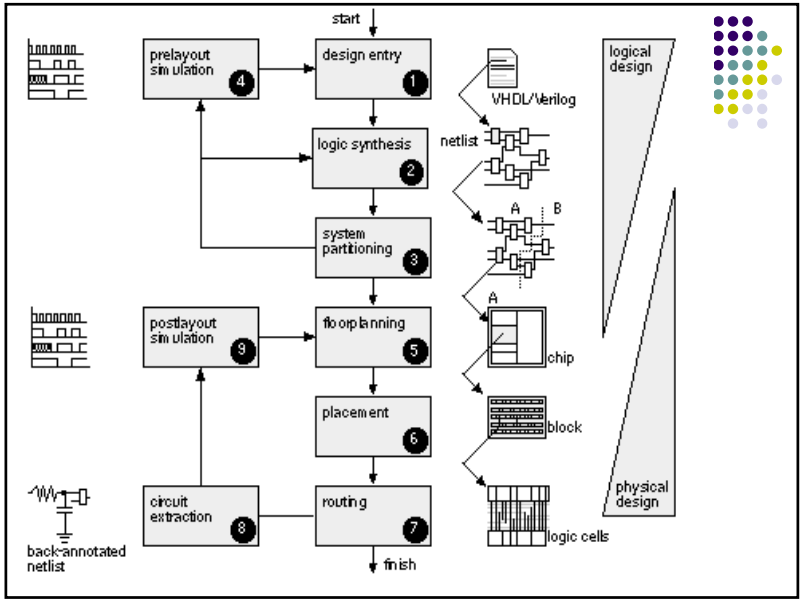
Acronyms

- HDL– Hardware Description Language
- RTL– Register Transfer Level
- ECO – Engineering Change Order
- DRC – Design Rule Checking
- ERC – Electrical Rule Checking
- LVS – Layout versus Schematic
- More can be found at :

http://www.ee.pdx.edu/~mperkows/IC_LAB/glossary.html

Flow Chat

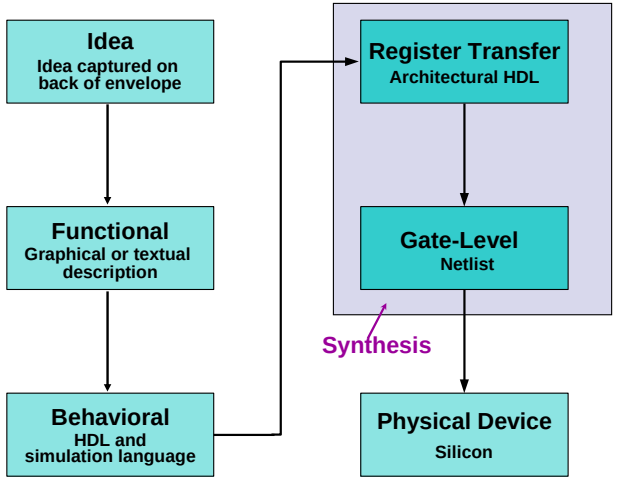




1. *Design entry.* Enter the design into an ASIC design system, either using a **hardware description language** (**HDL**) or *schematic entry* .
2. *Logic synthesis.* Use an HDL (VHDL or Verilog) and a logic synthesis tool,e.g. Design Compiler from Synopsys, to produce a **netlist** —a description of the logic cells and their connections.
3. *System partitioning.* Divide a large system into ASIC-sized pieces.
4. *Prelayout simulation.* Check to see if the design functions correctly.
5. *Floorplanning.* Arrange the blocks of the netlist on the chip.
6. *Placement.* Decide the locations of cells in a block.
7. *Routing.* Make the connections between cells and blocks.
8. *Extraction.* Determine the resistance and capacitance of the interconnect.
9. *Postlayout simulation.* Check to see the design still works with the added loads of the interconnect.

- Steps 1–4 are part of **logical design** , and steps 5–9 are part of **physical design** . There is some overlap. For example, system partitioning might be considered as either logical or physical design. To put it another way, when we are performing system partitioning we have to consider both logical and physical factors.

Levels of Abstraction



Logic Design Approaches

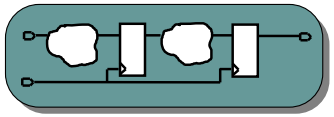
- Capture-and-Simulation
  - Draw gates and flip-flops
  - Debug by simulation
- Describe-and-Synthesis
  - Write Boolean equations, FSM or HDL
  - Transformation and compilation by synthesizer

Behavioral Level Simulation

- Purposes
  - Functionality
  - Algorithmic Correctness
- Ways to do it
  - System Studio(Synopsys SystemC); MatLab; SDL(Specification and Description Language)
  - C/C++/Java
  - SystemVerilog/Verilog/VHDL
- Drawback – No Cycle-Accuracy

RTL-Level Simulation

- **Purposes**
  - **Validation model for structural code**
  - **Full functionality**
- **Register transfer operations**
- **Contains complete functional description**
- **Cycle accurate**



Logic Synthesis

- Logic synthesis provides a link between an HDL (Verilog or VHDL) and a netlist
- Synthesis techniques
  - Two-level and multi-level logic minimization
  - FSM encoding
  - A lot of heuristics
- Synthesis tools – Design Compiler from Synopsys, BuildGates from Cadence

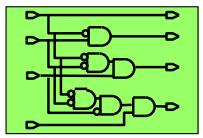
Inside Synthesis

Synthesis = Translation + Optimization + Mapping

```
residue = 16'h0000;
if (high_bits == 2'b10)
   residue = state_table[index];
else
   state_table[index] = 16'h0000;
```

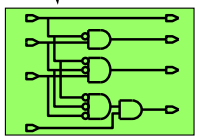
Translate

HDL Source



Generic Boolean (GTECH)

Optimize + Map



Target Technology

Why Synthesis?

Reusability
Parameterized code; Building-block approach; Retarget new libraries;

Design Tricks
DC knows plenty, tries them in context of loads, fanouts, library limitations

Verifiable
Validate, implement, & verify in same language, Less error-prone entry

Why Me?

Abstraction
Focus on high-level issues; tool & computer do dirty work to meet constraints

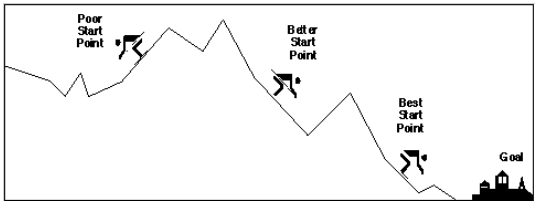
Portability
IEEE standards; HDL portable across tools; technology-independent designs

Productivity
How else to do 10^6 gates in 6 months?

Prestige
Impress friends; hot skill on resume; job security; wealth and fame

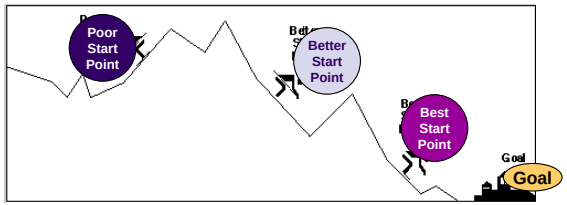
Basic Guideline

The Importance of Quality of Source



- Synthesis is NP-Hard problem - compile time grows exponentially with the circuit size.
- Heuristic minimization methods used to obtain near optimal.
- Rather than guaranteeing the best possible circuit, today's synthesis algorithms improve the starting circuit.

The Importance of Quality of Source

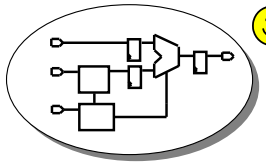


- **Code that is functionally equivalent, but coded differently, will give different synthesis results**
- **You cannot rely solely on the synthesis tool to "fix" a poorly coded design!**
- **Try to understand the "hardware" you are describing, to give the tool the best possible starting point**

The Big Picture, Think Hardware!

- **Write HDL hardware descriptions**
  - Think of the ***topology*** implied by the code
- **Do not write HDL simulation models**
  - No explicit delays
  - No file I/O

Yes!

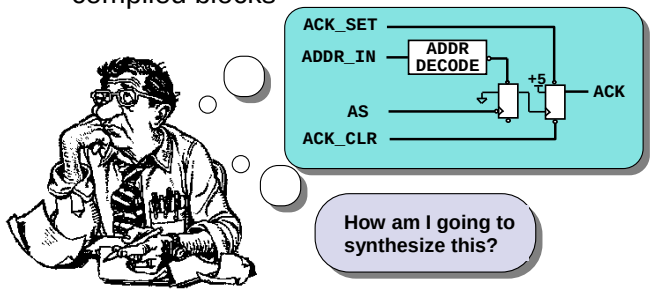


No!

```
after 20 ns and
    2 clock cycles
    OUTPUT <= IN1 + RAM1;
    wait 20 ns;
    ...
```

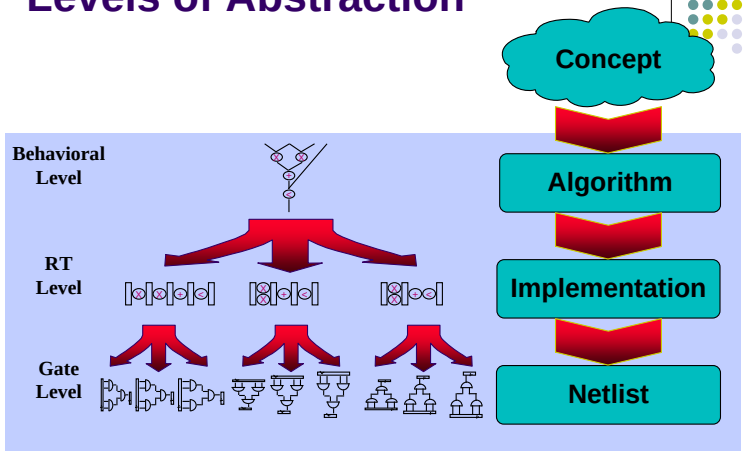
The Big Picture, Think Synchronous!

- **Synchronous designs run smoothly through synthesis, test, simulation, and layout**
- **Asynchronous designs may require hand instantiation and extensive simulation to verify**
  - Isolate asynchronous logic into separately compiled blocks

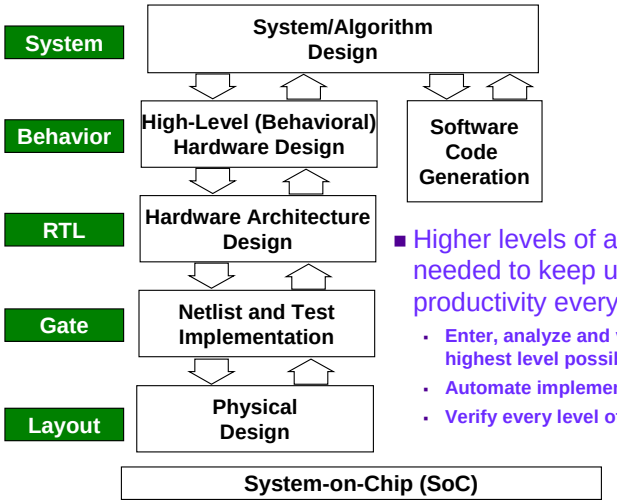


Levels of Synthesis

Levels of Abstraction



High Level Design Methodology



- Higher levels of abstraction needed to keep up with 10X productivity every 6 years
  - Enter, analyze and validate at the highest level possible
  - Automate implementation
  - Verify every level of abstraction