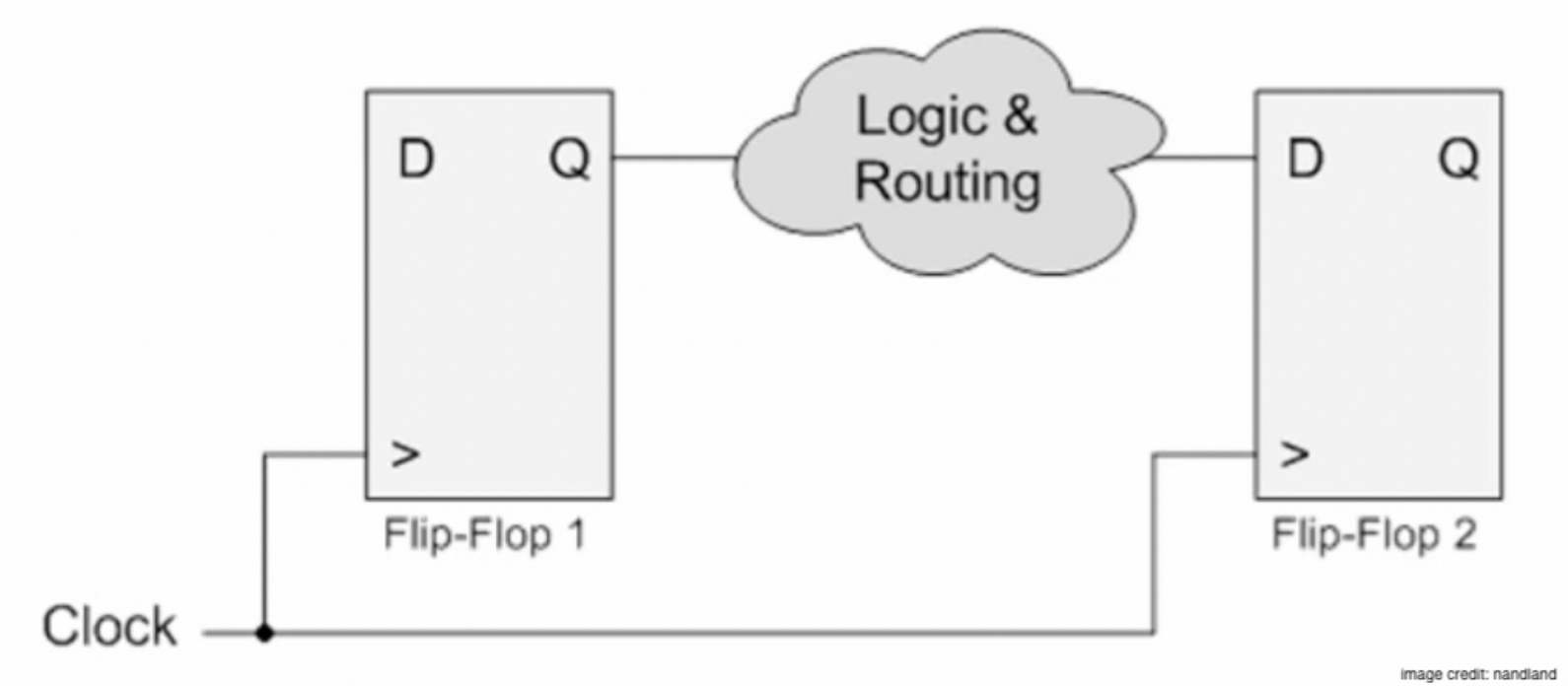
Lecture 2: Digital Logic Synthesis

Intro to Open-Source Chip Design

Design Flows

Digital Logic Refresher



Combinational logic (gates, LUTs) between synchronous clocked elements (flip-flops/registers, latches)

FPGA Design Flow

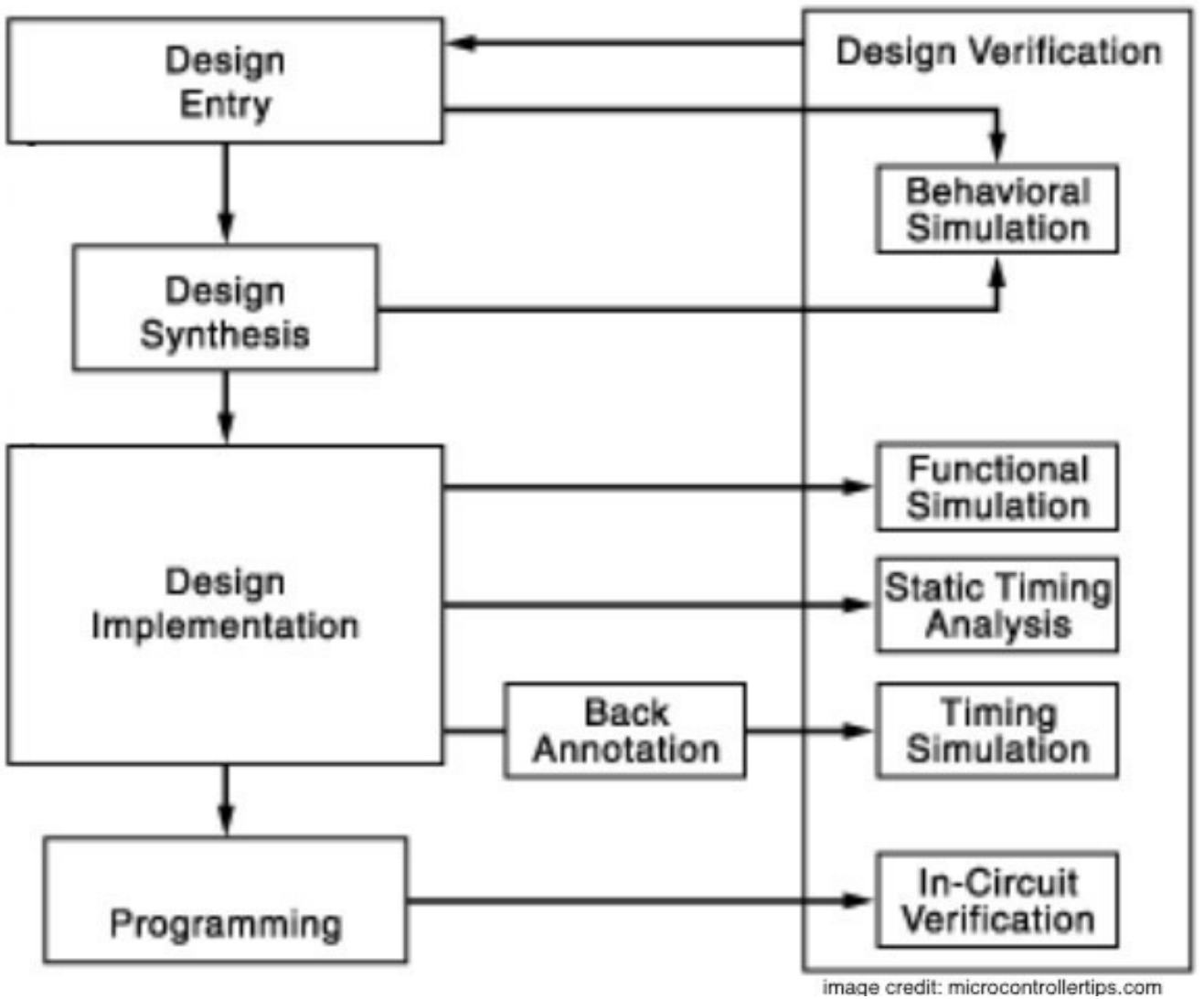
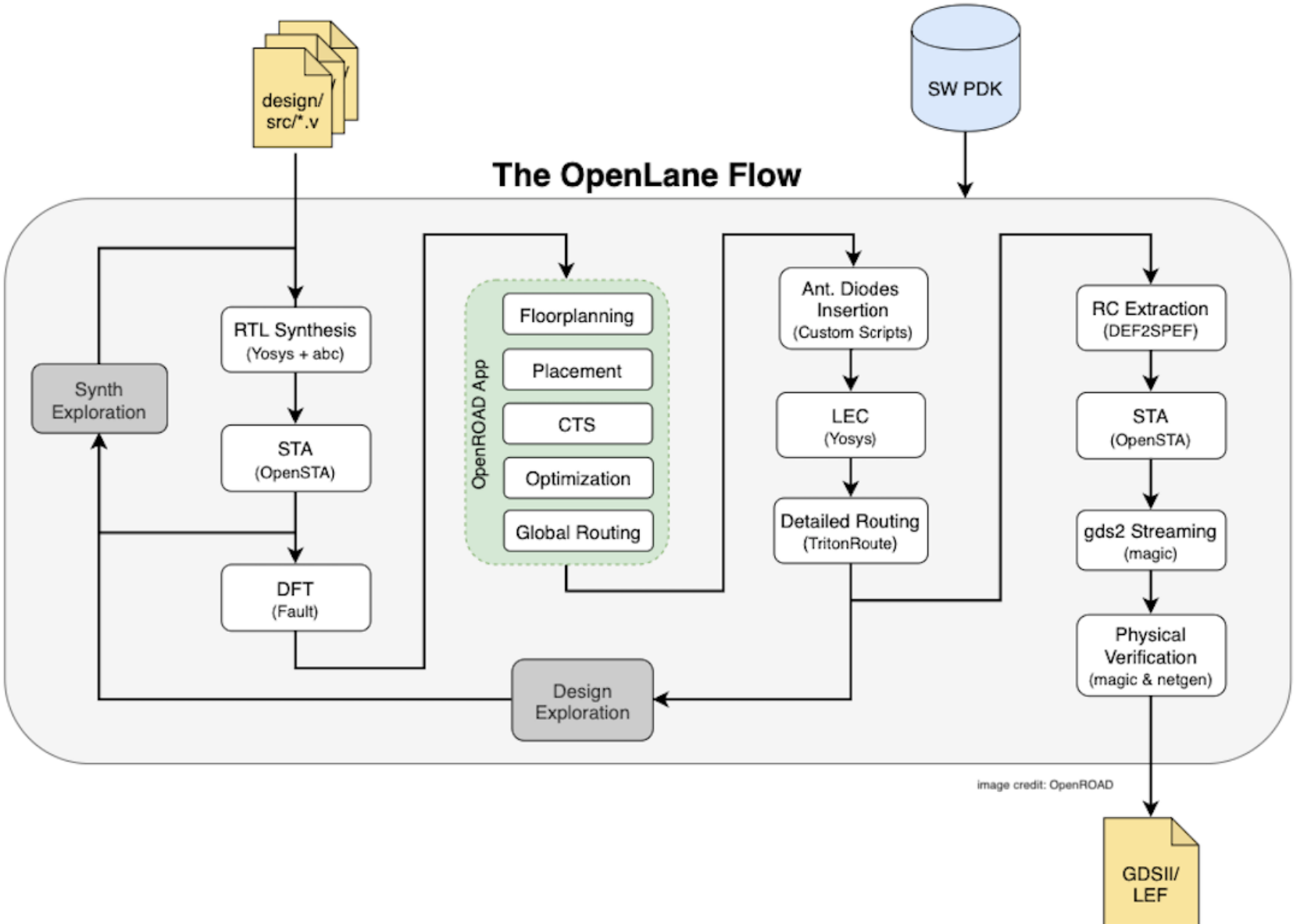


image credit: microcontrollertips.com

ASIC Design Flow



Necessity of Logic Synthesis

```
1  // We don't want to write
2  // code like this
3  module test (
4    input wire X, Y, Z,
5    output wire F, G, H);
6
7    wire Xn, Yn, Zn, N, M;
8
9    not n0 (Xn, X);
10   not n1 (Yn, Y);
11   not n2 (Zm, Z);
12
13   and a0 (N, Xn, Z);
14   and a1 (M, X, Zn, Y);
15   xor x0 (F, N, M);
16
17   or  o0 (H, Xn, Zn);
18 ...
```

```
1  // We'd rather write code like this
2  module branch_controller (
3    input wire [31:0] i_rs1, i_rs2,
4    input wire [2:0] btype,
5    output logic taken);
6
7  always_comb begin
8    case (btype[2:0])
9      BR_EQUAL: begin
10       taken = (i_rs1 == i_rs2);
11     end
12     BR_LT_SIGNED: begin
13       taken = ($signed(i_rs1) < $signe
14     end
15     BR_LT_UNSIGNED: begin
16       taken = (i_rs1 < i_rs2);
17     end
18 ...
```

Intro to Yosys

- Logic-synthesis and manipulation multitool
  - Verilog frontend (among other input formats)
  - Design manipulation & optimizations
  - Inference of FSMs, memories, and other hard logic
  - Techmapping (to FPGA tech, ASIC PDK, etc.)
  - Formal verification passes (limited)
  - Visualizations
  - Various backends (to other software)

Levels of Abstraction for Digital Logic

- System-Level / High-Level (block diagrams, software models)
- Behavioral Level (Verilog/VHDL/etc)
- Register-Transfer Level (registers and logical connections)
- Gate-Level (single-bit registers, basic logic gates)
- Physical-Gate Level / Switch-Level (underlying physical process)
  - An example of “physical gates” in an ASIC PDK

Synthesis Demo: Verilog handling

Synthesis Demo: CXXRTL

Yosys Optimization Passes

- `opt_muxtree` - Remove dead inputs in multiplexers
- `opt_reduce` - Consolidate AND/OR/MUX trees
- `opt_merge` - Combine redundant cells (same connections)
- `opt_dff` - Combine enable, and reset into DFF (convert to SDFF, DFFE, SDFFE, usually available in the physical process)
- `opt_clean` - Remove unused or unconnected cells
- `opt_expr` - Constant-folding on internal cells
- **`opt` - Run all of the above, in a loop, repeatedly**

Synthesis Demo: Constant folding

`synth_X` commands

- Manually scripting Yosys is very powerful, but also time-consuming
- Fortunately, Yosys comes with built-in `synth_X` targets:
  - `synth` / `synth -lut` - Basic logic-gates or LUTs
  - `synth_ice40`, `synth_ecp5`, `synth_xilinx`, ... - Targets for many different FPGA targets
  - Other flows (i.e. ASIC toolchains) come with their own synth scripts
  - Example of the basic `synth` command

Synthesis Demo: LUT mapping

Synthesis Demo: SAT solving