Synthesis and ASIC Design Flow

Dr. Sudeendra kumar K

Department of Electronics and Communication Engineering

ADVANCED DIGITAL DESIGN

Synthesis

Unit-III Session -2

Sudeendra kumar K
Department of Electronics and Communication Engineering

Advanced Digital Design

Contents

- ASIC Design flow
- Introduction to Synthesis

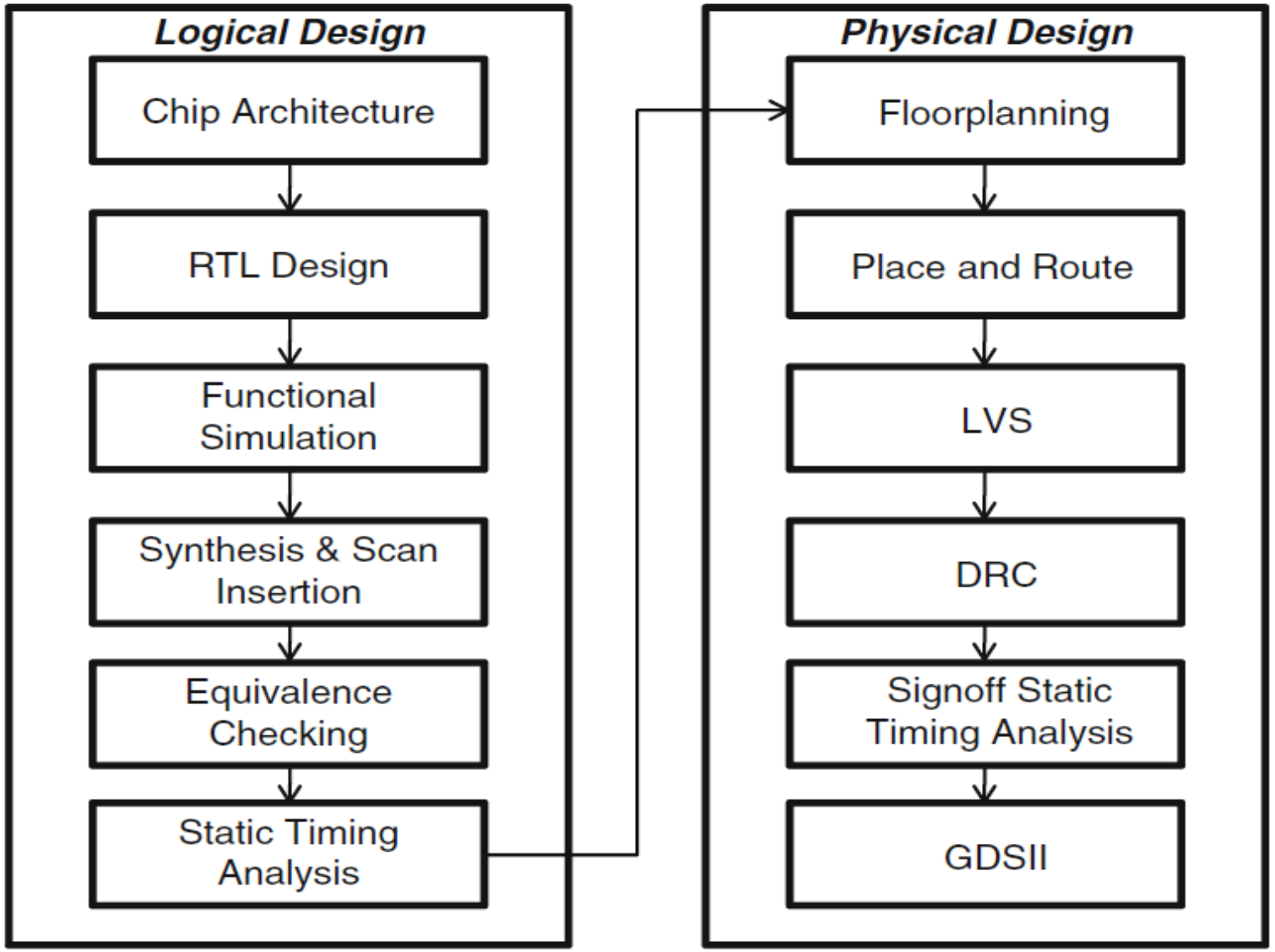
Advanced Digital Design

ASIC Design Flow

- A typical ASIC flow can be broadly categorized into logical design and physical design. Logical design begins with high-level design specification and chip architecture.
- The chip architect captures high-level functional, power (how much power should the design consume) and timing (at what speed should the design operate) requirements.
- Commonly referred to as RTL (register transfer level) (Verilog), this provides an abstraction of the functional behavior of the circuit in terms of how the logical operations on signals enable data to flow between registers (flops) in a design.
- Once the functionality of the design is coded, it is verified using simulation . Simulation is a process where various stimuli are applied to a representation of a design and the response of the design is captured.
- Synthesis (aka logic synthesis) is the step where RTL description is translated to a gate-level representation which is a hardware equivalent implementation of the functionality described in the HDL .

Advanced Digital Design

ASIC Design flow



Synthesis

- Synthesis (aka logic synthesis) is the step where RTL description is translated to a gate-level representation which is a hardware equivalent implementation of the functionality described in the HDL.

- An HDL description is said to be synthesizable RTL, if it can be consumed by industry standard synthesis tools to map to a unique and unambiguous implementation.

- Synthesis in the context of electronic design means realization of a gate-level netlist to achieve a specific functionality. Besides the specific functionality, the process of synthesis might also meet certain other requirements, namely, power, frequency of operation, etc.

Synthesis Process

Design in HDL Description

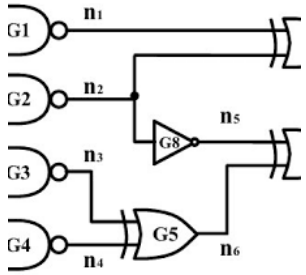
Standard Cell Library

Constraints on Timing, Power & Area

Inputs to Synthesis

Synthesis

Gate-level Netlist



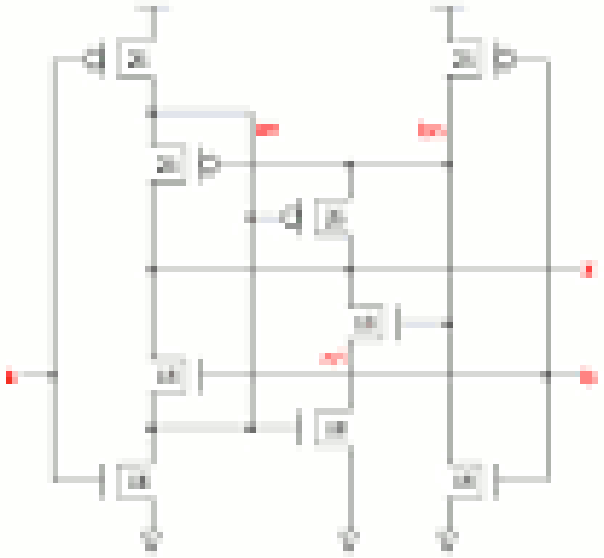
Reports on Timing, Area and Power

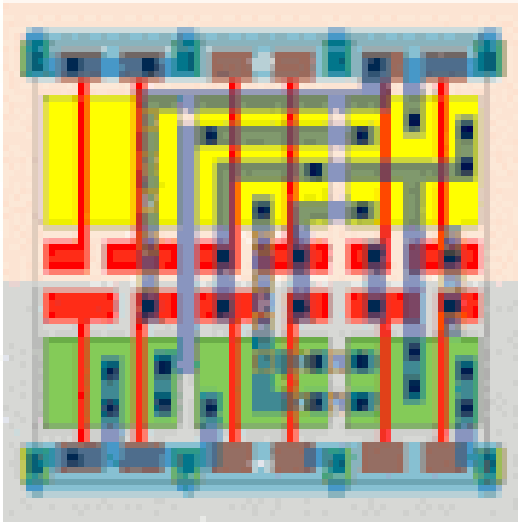
Output of Synthesis

Advanced Digital Design

Standard Cell library

- A standard cell library is a collection of low-level electronic logic functions such as AND, OR, INVERT, flip-flops, latches, and buffers.
- A standard-cell library may also contain the following additional components: -
  - A full layout of the cells
  - SPICE models of the cells
  - Verilog models
  - Parasitic extraction models



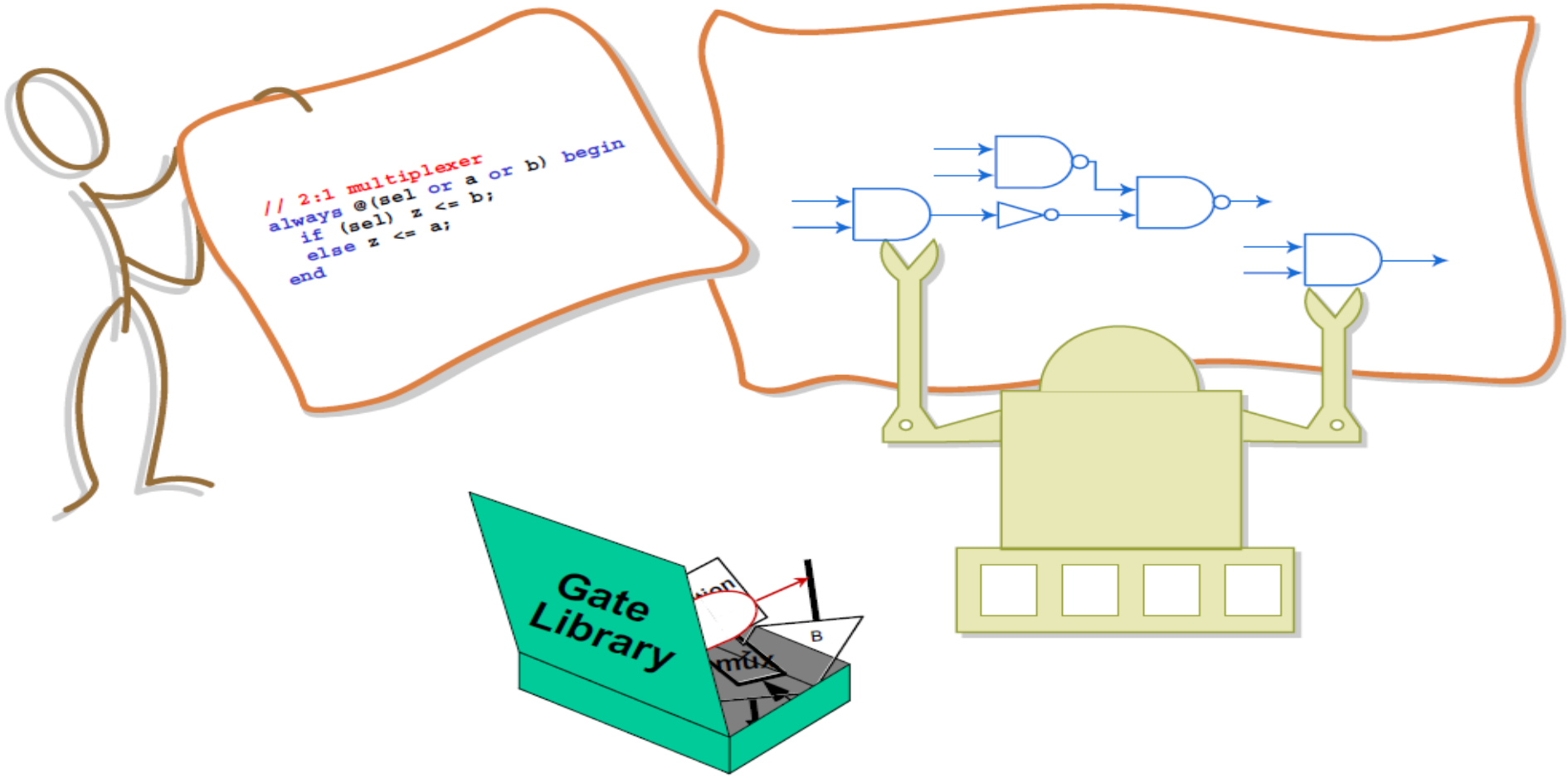


Advanced Digital Design

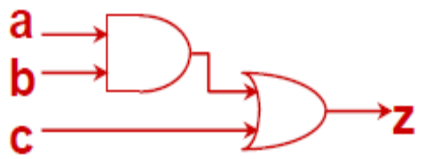
Standard Cell library

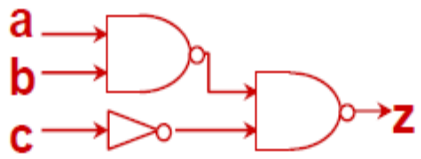
- Commercially available Electronic Design Automation (EDA) tools use the technology libraries to automate synthesis, placement, and routing of a digital ASIC.
- The Logic Synthesis tool performs the process of mathematically transforming the ASIC's register-transfer level (RTL) description into a technology-dependent netlist.
- This process is analogous to a software compiler converting a high-level C-program listing into a processor-dependent assembly-language listing.
- The netlist is the standard-cell representation of the ASIC design, at the logical view level. It consists of instances of the standard-cell library gates, and port connectivity between gates.
- Proper synthesis techniques ensure mathematical equivalency between the synthesized netlist and original RTL description.

Synthesis: Verilog → Gates

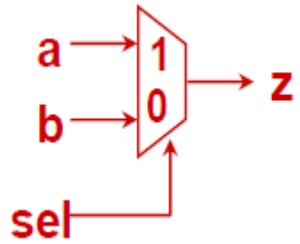


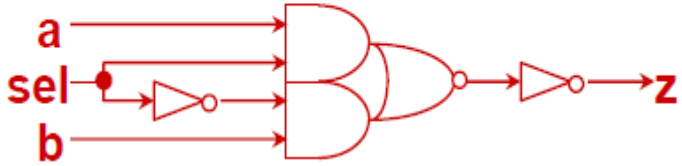
```
assign z = (a & b) | c;
```



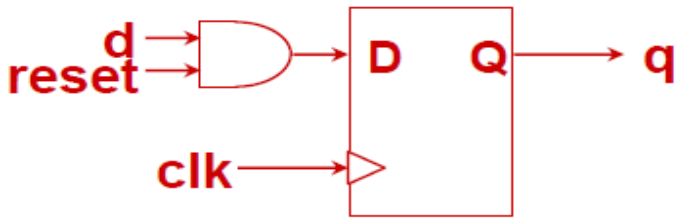



```
// dataflow
assign z = sel ? a : b;
```

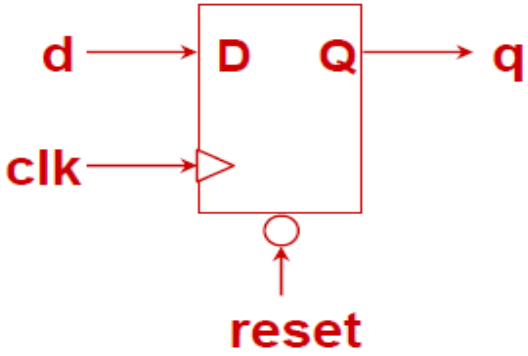




```
reg q;
// register with synchronous clear
always @(posedge clk) begin
  if (!reset)  // reset is active low
    q <= 0;
  else
    q <= d;
end
```




```
reg q;
// register with asynchronous clear
always @(posedge clk or negedge reset)
begin
  if (!reset)  // reset is active low
    q <= 0;
  else         // implicit posedge clk
    q <= d;
end
// warning! async inputs are dangerous!
// there's a race between them and the
// rising edge of clk.
```



Advanced Digital Design
Timing Constraints

- The successful tape out of any chip is measured by a variety of factors. This includes how well the design adheres to the timing, power, and area objectives set by the architect in addition to meeting all the functional requirements.
- From a timing perspective, at the architecture stage the architect will assign block budgets which are handed off to block owners.
- Depending on whether a block is a derivative design or being developed from scratch, the RTL designer will create initial timing constraints or tweak existing ones for synthesis.
- This will form the baseline for all runs in the implementation flow and typically includes defining clock frequency and budgets in the subblocks. This results in an unoptimized netlist with ideal clocks (clock with zero delay).

```
# Constrain clock port clk with a 10-ns requirement

create_clock -period 10 [get_ports clk]

# Automatically apply a generate clock on the output of phase-locked loops (PLLs)
# This command can be safely left in the SDC even if no PLLs exist in the design

derive_pll_clocks

# Constrain the input I/O path

set_input_delay -clock clk -max 3 [all_inputs]

set_input_delay -clock clk -min 2 [all_inputs]

# Constrain the output I/O path

set_output_delay -clock clk -max 3 [all_inputs]

set_output_delay -clock clk -min 2 [all_inputs]
```

Advanced Digital Design

Area, Power and Timing Constraints

Type of information	Commands
Operating conditions	set_operating_conditions
Wire load models	set_wire_load_min_block_size set_wire_load_mode set_wire_load_model set_wire_load_selection_group
System interface	set_drive set_driving_cell set_fanout_load set_load set_port_fanout_number
Design rule constraints	set_max_capacitance set_max_fanout set_max_transition set_min_capacitance set_min_fanout

Type of information	Commands
Timing constraints	create_clock create_generated_clock set_clock_gating_check set_clock_latency set_clock_transition set_clock_uncertainty set_disable_timing set_input_delay set_input_transition set_max_time_borrow set_output_delay set_propagated_clock set_resistance
Timing exceptions	set_false_path set_max_delay set_min_delay set_multicycle_path
Area constraints	set_max_area
Logic assignments	set_case_analysis set_logic_dc set_logic_one set_logic_zero

Important Output files from Synthesis: Reports

```
genus@root:> report_gates
=============================================================
  Generated by:           Genus(TM) Synthesis Solution GENUS15.21 - 15.20-s010_1
  Generated on:           Jun 21 2019  04:22:38 pm
  Module:                 univ_bin_counter
  Technology library:     saed90nm_typ updated 30.Sep.2011
  Operating conditions:   _nominal_ (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=============================================================
```

Gate	Instances	Area	Library
AND2X1	7	51.610	saed90nm_typ
DFFARX1	8	258.048	saed90nm_typ
HADDX1	7	109.670	saed90nm_typ
INVX0	2	11.059	saed90nm_typ
NOR2X0	1	5.530	saed90nm_typ
XNOR2X1	1	13.824	saed90nm_typ
total	26	449.741	

Type	Instances	Area	Area %
sequential	8	258.048	57.4
inverter	2	11.059	2.5
logic	16	180.634	40.2
physical_cells	0	0.000	0.0
total	26	449.741	100.0

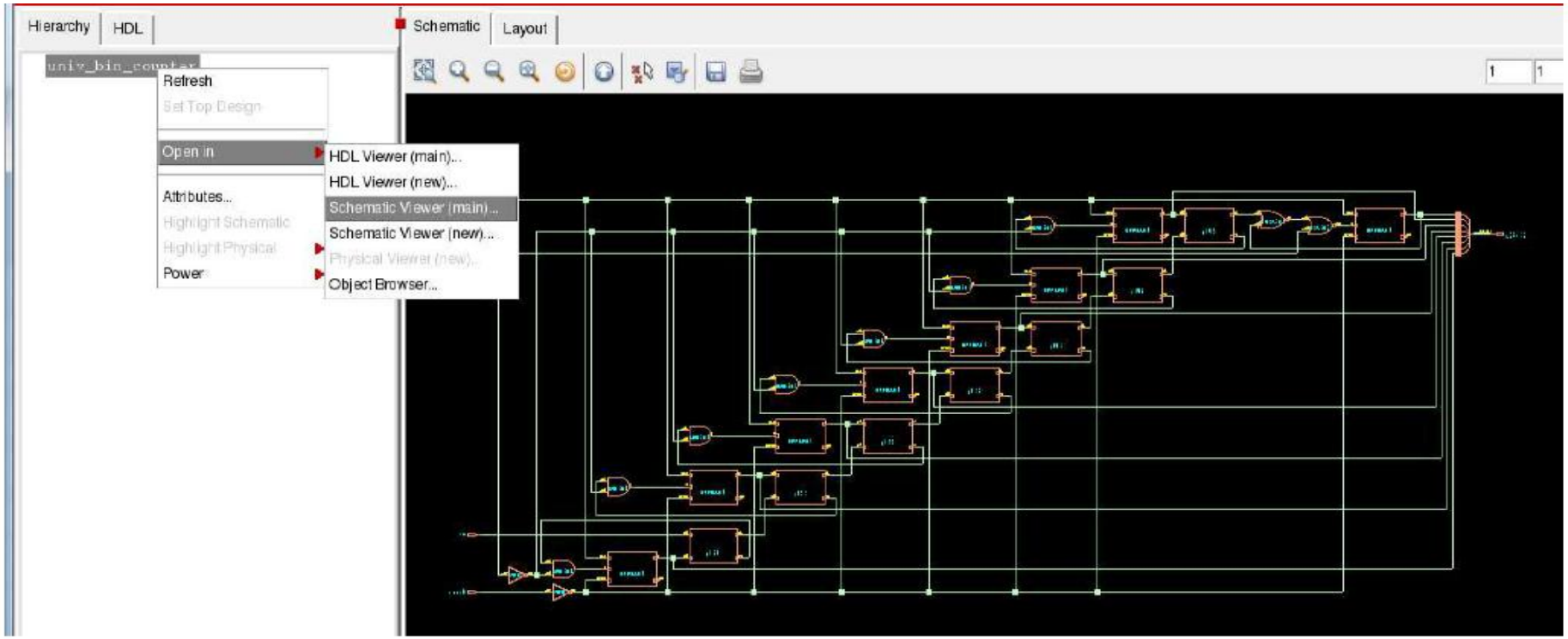
```
genus@root:> report_power
=============================================================
  Generated by:           Genus(TM) Synthesis Solution GENUS15.21 - 15.20-s010_1
  Generated on:           Jun 21 2019  04:41:31 pm
  Module:                 univ_bin_counter
  Technology library:     saed90nm_typ updated 30.Sep.2011
  Operating conditions:   _nominal_ (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
=============================================================

                    Leakage   Dynamic     Total
   Instance   Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------------
univ_bin_counter    26  2203.558 11708.397 13911.954
```

Important Output files from Synthesis: Reports

```
genus@root:> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'univ_bin_counter'.
        : Use 'report timing -lint' for more information.
====================================================================
  Generated by:           Genus(TM) Synthesis Solution GENUS15.21 - 15.20-s010_1
  Generated on:           Jun 21 2019  04:43:34 pm
  Module:                 univ_bin_counter
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

Path 1: MET (8707 ps) Setup Check with Pin r_reg_reg[7]/CLK->D
          Group: clk
     Startpoint: (F) en
          Clock: (R) clk
       Endpoint: (R) r_reg_reg[7]/D
          Clock: (R) clk

                    Capture        Launch
      Clock Edge:+    10000             0
     Src Latency:+        0             0
     Net Latency:+        0 (I)         0 (I)
         Arrival:=    10000             0

           Setup:-       64
     Uncertainty:-      200
   Required Time:=     9736
    Launch Clock:-        0
     Input Delay:-      300
       Data Path:-      728
           Slack:=     8707

#-------------------------------------------------------------------------------------------
# Timing Point    Flags     Arc      Edge    Cell       Fanout Load Trans Delay Arrival
#                                                              (fF)  (ps)  (ps)   (ps)
#-------------------------------------------------------------------------------------------
  en                -         -        F     (arrival)       1  3.6     0     0     300
  g161/C1           -         A0->C1   F     HADDX1          1  3.1    44    83     383
  g158/C1           -         B0->C1   F     HADDX1          1  3.1    44    89     472
  g155/C1           -         B0->C1   F     HADDX1          1  3.1    44    89     561
  g152/C1           -         B0->C1   F     HADDX1          1  3.1    44    89     650
  g149/C1           -         B0->C1   F     HADDX1          1  3.1    44    89     739
  g146/C1           -         B0->C1   F     HADDX1          1  3.1    44    89     828
  g143/C1           -         B0->C1   F     HADDX1          1  2.6    42    88     916
  g141/Q            -         IN1->Q   F     XNOR2X1         1  2.1    33    88    1003
  g139/QN           -         IN1->QN  R     NOR2X0          1  1.5    44    25    1028
  r_reg_reg[7]/D <<<  -       -        R     DFFARX1         1    -     -     0     1028
#-------------------------------------------------------------------------------------------

genus@root:>
```

Advanced Digital Design

Performing Gate-level Simulations: Save the netlist



Save the synthesized gate level netlist: -
Command: - write_hdl > counter_design_netlist.v

```
    flow.cputime  flow.realtime  timing.setup.tns  timing.setup.wns  snapshot
UM:             0          1835              -0 ps         8707.3 ps  synthesize
genus@root:> write_hdl > counter_design_netlist.v
genus@root:>
```

Performing Gate-level Simulations: Command in NcVerilog

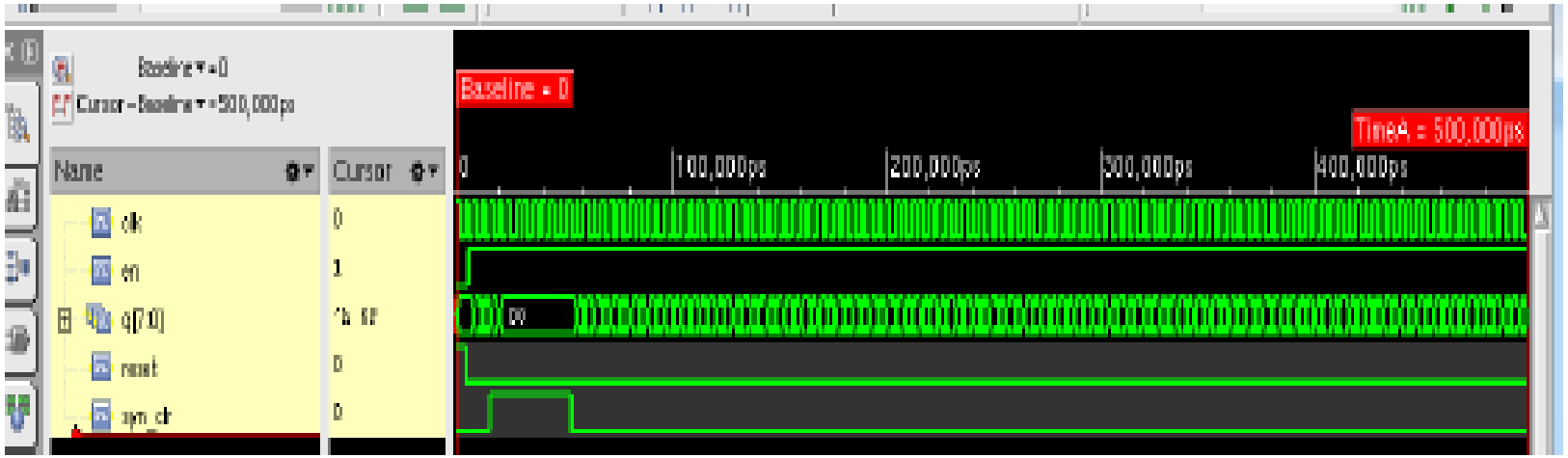
- Save the testbench and perform gate level simulations. The files required to perform gate level simulations are: -
  - Gate level netlist (from synthesis) (*counter_design_netlist.v)*
  - Testbench used in RTL simulation and add SDF instantiation. (*counter_tst.v*)
- Standard cell library in the Verilog format (saed90nm.v) (stored in libs folder, give path)
- The working directory must have all the above files.
- Gate level simulation using Cadence NcSim can be performed as shown below:

  - ***Command: ncverilog testbench.v design_netlist.v -v library.v +access+rw +gui***

```
[sudi@sankh lab1]$ ls
cm.log          constraints2.sdc          counter_design_netlist.sdf.X  counter_tst.v   counter.v  csrc      fv         genus.l
constraints1.sdc  counter_design_netlist.sdf  counter_design_netlist.v      counter_tst.v~  cov_work   INCAfiles  genus.cmd  imc.log
[sudi@sankh lab1]$ ncverilog counter_tst.v counter_design_netlist.v +v ../libs/saed90nm.v +access+rw +gui~
```

Performing Gate-level Simulations: Command in NcVerilog

- *Command: ncverilog testbench.v design_netlist.v -v library.v +access+rw +gui*



Advanced Digital Design

Summary and References

- Introduction to ASIC Design flow
- Basics of Logic Synthesis

References

- This Presentation
- Online Class Lecture Explanation

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu

Linting

Dr. Sudeendra kumar K

Department of Electronics and Communication Engineering

ADVANCED DIGITAL DESIGN

Linting

Unit-III Session -3

Sudeendra kumar K
Department of Electronics and Communication Engineering

Advanced Digital Design

Contents

- Linting
- Linting Tools

Advanced Digital Design

Linting

- Originally, Lint was the name attached to a Unix utility that could flag non-portable or suspicious C code.
- Lint is classified as a static analysis tool in that it does not execute the code, instead examining it based on a set of rules. Today, Lint is applied to many different languages including those used in the design of electronic systems.
- As soon as RTL designers start writing code they will begin to introduce unintended errors. To eliminate these errors, designers will use a variety of tools to ensure the code is correct before hand-off.
- Functional errors are typically caught by a mix of static tools (auto-formal and assertion-based) and dynamic tools such as simulation.
- Linting has the advantages in that it can deliver feedback about troublesome and even dangerous coding styles that would take a much longer time to uncover using simulation.
- With the right lint tool, you can catch the “low-hanging fruit” before tackling functional errors.

Courtesy: - https://semiengineering.com/knowledge_centers/eda-design/verification/lint/

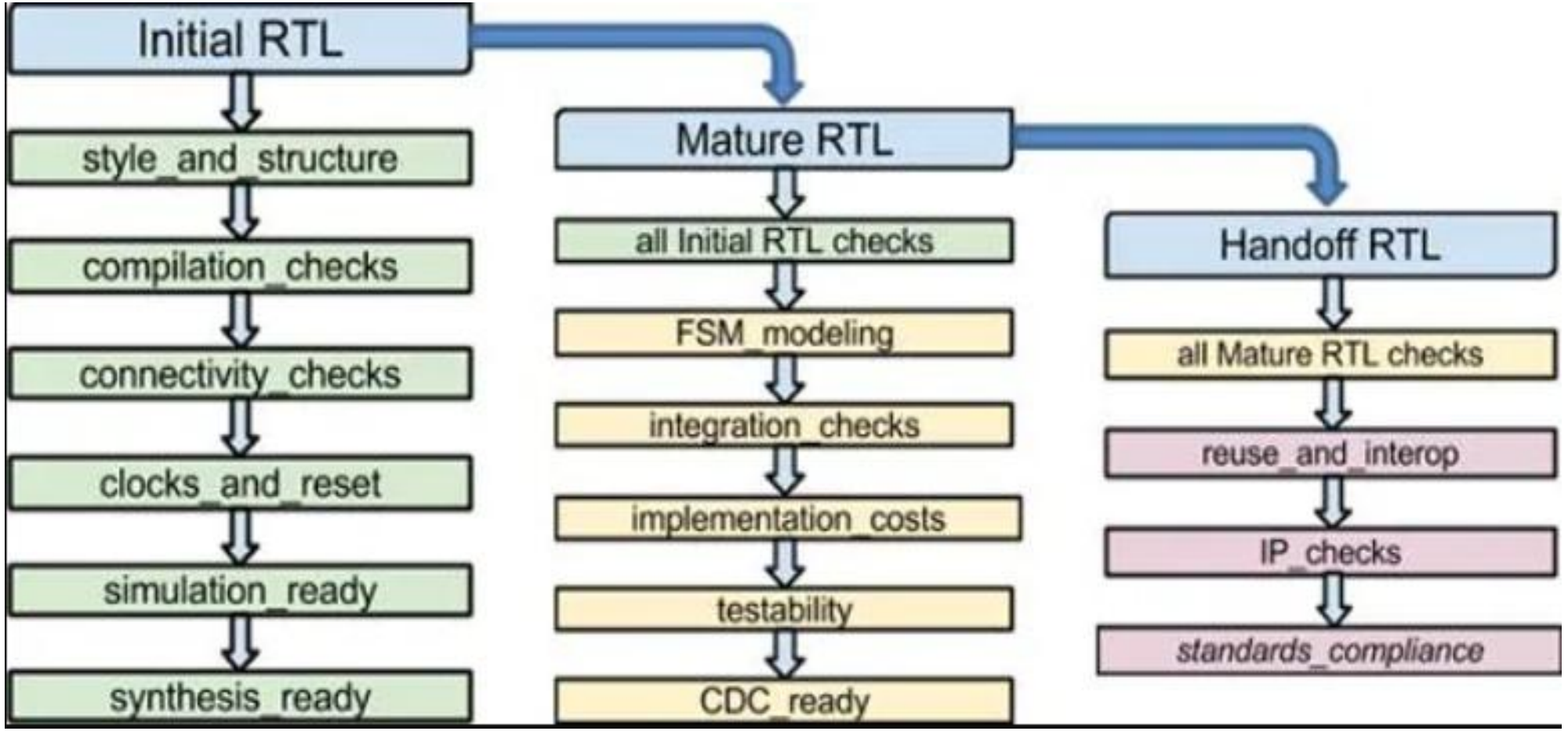
Linting

- Lint tools often use policy files. Each policy file is intended to achieve a significantly greater level of maturity towards achieving quality RTL using a set of Lint rules.
- The policies can be tailored to apply across the broad spectrum of design types, but may be adjusted as needed. Design teams, after careful consideration, may skip individual steps in the flow in keeping with their priorities.
- Additionally, the sequence of policies is optimized to lend itself for early detection, faster debug and low noise. Here again, design teams may choose to re-order the recommendations based on their best practices.

One particular flow classifies HDL maturity into three stages of Initial, Mature and Handoff. The three stages are defined as follows: -

- **Initial RTL –** Initial RTL represents the early phase where the requirements may still be evolving. It ensures that regressions and builds failures are caught early.
- **Mature RTL –** Modeling costs, simulation-synthesis mismatches, FSM complexity, etc. are higher order aspects of freeze-ready RTL that can significantly impact the design quality. The Mature RTL checks ensure necessary conditions for downstream interoperability.
- **Handoff RTL –** At the handoff stage, the checks are geared towards compliance with industry standards or internal conventions, to allow easy integration and reuse.

Usefulness of Linting



Advanced Digital Design

Verilog Linting example

```
// Example: a[1:2]

module top(a, b);
input [2:1]a;
output [2:1]b;
reg [2:1]b;

always @ (a[1:2] or a[2])
     if(a[2])
        b<=1'b0;
     else if(b)
        b<=1'b0;
     else
        b<=a;
endmodule
```

Linting tool detects that the high index of the width range is in the RHS.

```
// Example: test(clk, reset, d, q);

module top (clk, reset, d, q);
input clk, reset, d;
output q;

test (clk, reset, d, q); //E54, no inst

endmodule

module test (clk, reset, d, q);
input clk, reset, d;
output q;
reg q;

always @(posedge clk or posedge reset)
if (reset == 1'b1)
q <= 1'b0;
else
q <= d;

endmodule
```

Linting tools detects that there is no instance name for a module. To solve this problem, name the instance.

Advanced Digital Design

Verilog Linting Examples

```
// Example: input [2:1]a;
...
always @ (posedge a[1:0])
...
module top(clk, reseta, resetb, a, b);
input clk, reseta, resetb;
input [2:1]a;
output [2:1]b;
reg [2:1]b;

always @ (posedge a[1:0])
     if(reseta)
        b<=1'b0;
     else if(resetb)
        b<=1'b0;
     else
        b<=a;
endmodule
```

Range index out of bound

```
reg r;
...
case(r)
   1'b0: ...
   1'b1: ...
endcase

module test(clk,reset,gate,phase);
input clk, reset;
input [1:0] gate;
output [3:0] phase;
wire clk, reset;
wire [1:0] gate;
reg [3:0] phase;

always @(posedge clk) begin
    if (reset) begin
       phase <= 4'b0000;
    end
    else begin
       case (gate)
          2'b00: phase <= 4'b0001;
          2'b10: phase <= 4'b0100;
          2'b01: phase <= 4'b0010;
          2'b11: phase <= 4'b0001;
       endcase
    end
end

endmodule
```

Leda fires for this rule when it finds a case statement that has no default clause, but which appears to cover all cases.

Advanced Digital Design

Linting Examples

```
// Example: out = in1 * in2; out, in1 and in2 are all one bit.

module test(in1,in2,out);
input in1, in2;
output out;
wire in1, in2;
wire out;

assign out = in1 * in2; //W131
endmodule
```

Leda fires for this rule when there is a potential loss of precision in multiplication.

```
// Example:

a = (1'b0) ? b : c;
```

Linting tool detects if the condition in a conditional expression is constant.

Advanced Digital Design

Linting Examples

```
// Example: input x ; assign x = .....;
module w188 ( a, b, ps, bs, x);
input a, b;
input [3:0] ps;
input [1:0] bs;
input x;

assign ps[1:0] = a & b; // Rule W188 is flagged here
assign bs[1] = a | b; // Rule W188 does not flag here
assign x = a & b; // Rule W188 is flagged here

endmodule
```

```
// Example: r1 <= repeat (2) @(posedge clk) inp*2;

module warn(go, clk, inp);
input go, clk;
input [9:0]inp;
reg [15:0]r1;
always @(posedge go)
  r1 <= repeat (2) @(posedge clk) inp*2;
endmodule
```

Repeated assignment in non-blocking.

Advanced Digital Design

Linting Examples

```
// Example:

module top(clk, reset1, reset2, a, b, c);
input clk, reset1, reset2, a;
output b, c;
reg b, c;

always @ (posedge clk or posedge reset1)
   if (reset1)
        b<=1'b0;
   else
        b<=a;

always @ (posedge clk or posedge reset2)
   if (reset2)
        c<=1'b0;
   else
        c<=~a;
endmodule
```

Multiple resets

```
// Example:

module top(clk1, clk2, a, b);
input clk1, clk2, a;
output b;
reg b;

always @ (posedge clk1 or posedge clk2)
      b<=a;
endmodule
```

Multiple Clocks in always block

Advanced Digital Design

Linting Examples

```
// Example:

 resetInt
 module w402 (clk,dataIN,dataOUT,testMode);

   input clk,dataIN,testMode;
   output dataOUT;
   reg    dataOUT;

   wire   resetInt;

   assign resetInt = testMode ? 1'b1 : 1'b0;

   always @ (posedge clk or posedge resetInt)
     begin
        if (resetInt)
          dataOUT <= 0;
        else
          dataOUT <= dataIN;
     end

endmodule
```

Reset is not an input to the module

```
// Example: b = clk;

module top(clk, reset, a, b);
input clk, reset, a;
output b;
reg b;

always @ (posedge clk)
   b = clk;

always @ (posedge clk)
   if(reset)
      b<=1'b00;
   else
      b<=a;
endmodule
```

Clock is used as data

Advanced Digital Design

Linting Examples

```
// Example: if: (if (...) ... else ...)

module test(clk, a, b);
input clk, a, b;

always @(posedge clk)
if (a)
  if (b)
    $display ("a and b");
else  //W527, indentation is wrong. This else belongs to the second if.
$display ("Not a");
endmodule
```

'if' without an 'else' when one may be expected (dangling 'else' for a nested 'if'). Make sure the nesting is correct

Advanced Digital Design

Summary

- Linting and its Importance

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu

Equivalence Checking

Dr. Sudeendra kumar K

Department of Electronics and Communication Engineering

ADVANCED DIGITAL DESIGN

Equivalence Checking

Unit-III Session -3

Sudeendra kumar K
Department of Electronics and Communication Engineering

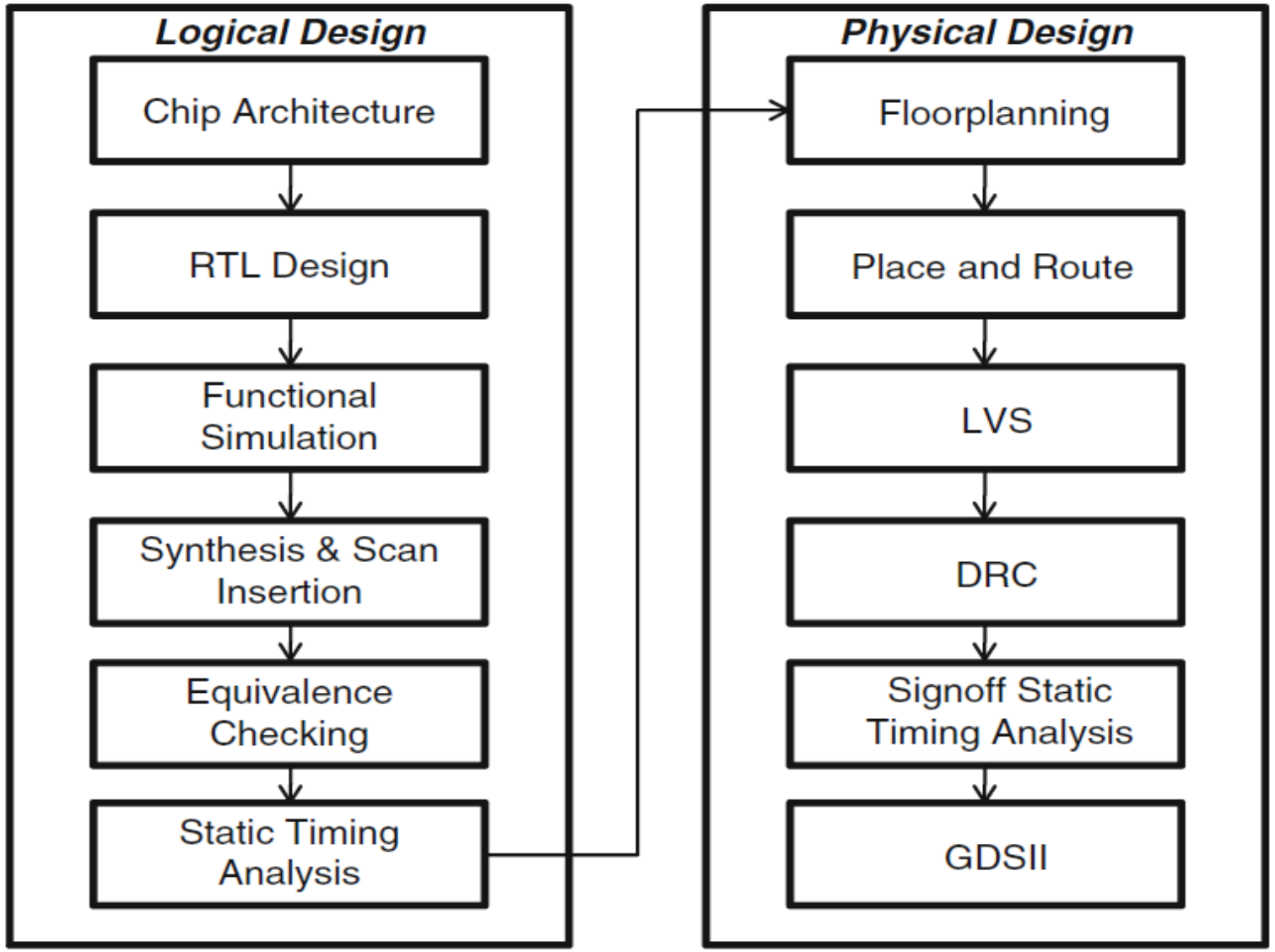
Advanced Digital Design

Contents

- Equivalence Checking

Advanced Digital Design

ASIC Design flow



Formal Equivalence Checking (FEV)

- The need for FEV can be seen as a natural outgrowth of the increased level of design abstraction as the industry has matured over the past half century. During the initial days of chip design (before 1980), the designers used to draw the circuits by hand and work at the transistor level.
- In the following decade (1981-1989), the drawing of such circuits was improved with the aid of computer-aided design (CAD), and circuit design became much simpler, but was not yet optimal.
- The RTL mode of designing started around the early 1990s and is still the major mode of logic design as of this writing. With the advent of RTL, new tools were introduced to automatically synthesize the coded design into functional schematic netlists.
- This allowed the generation of much more complex netlists than were possible with previous methods, and necessitated good RTL-versus netlist FEV tools.

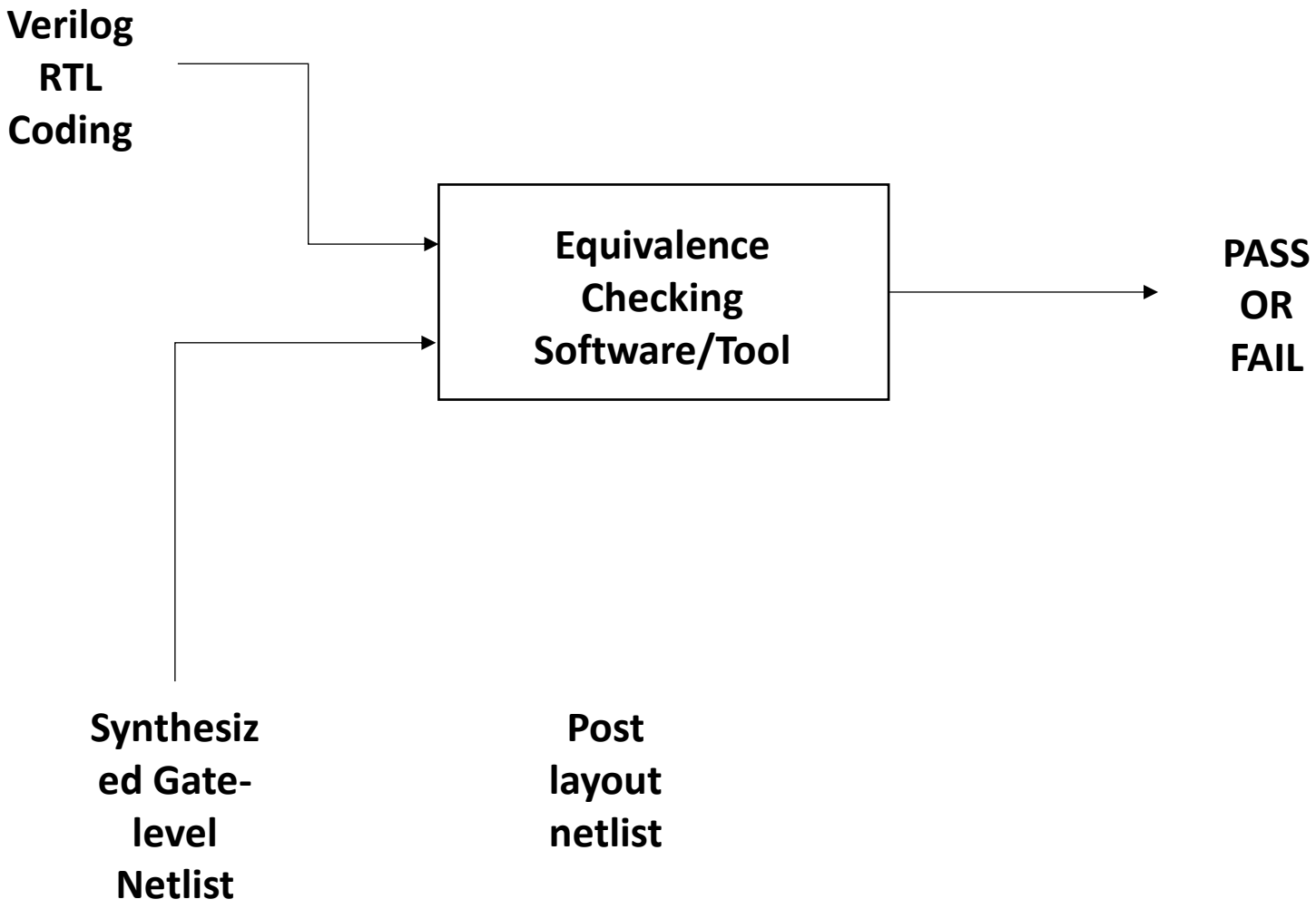
Courtesy: - Formal Verification, Erik Seligman

Equivalence Checking

- Formal equivalence checks have become an even more important requirement to ensure that designs are faithfully translated across the different abstraction levels.
- FEV is one of the most mature FV techniques; it is now considered a standard requirement to ensure the design intent is maintained as designers refine their abstractions into real designs.

Advanced Digital Design

Equivalence Checking block diagram



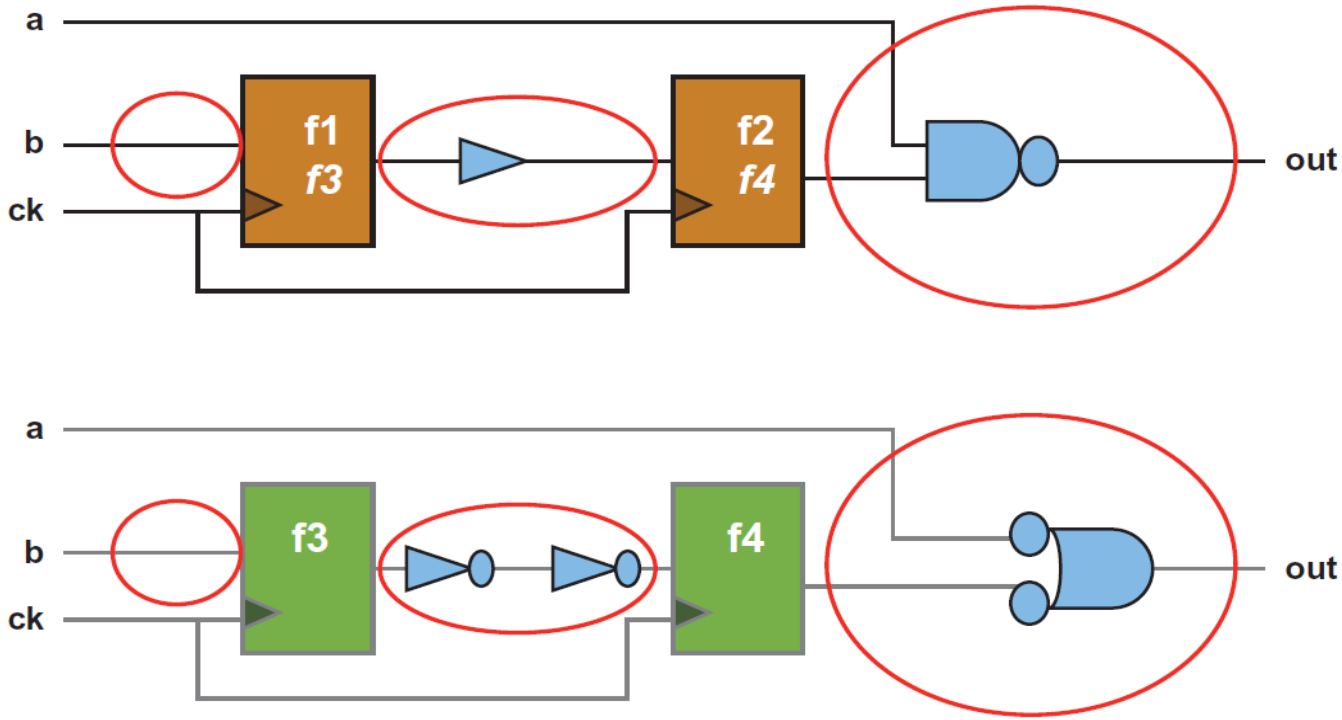
Types of Equivalence to Check

- Before we begin a detailed discussion of FEV techniques, it is important to define what we mean by “equivalence.”
- In a general sense, we define equivalence by choosing a set of key points, points in the two models being compared that are expected to be logically identical given the same set of input stimuli.
- FEV then consists of assuming the input key points receive identical values, and verifying that the logic cones driving the internal and output key points will then be equivalent in the two models. Depending on the equivalence checking technique being used, the key points may include:
  - Primary inputs
  - State elements (latches and flip-flops)
  - Blackboxes
  - Cut points

Types of Equivalence Checking

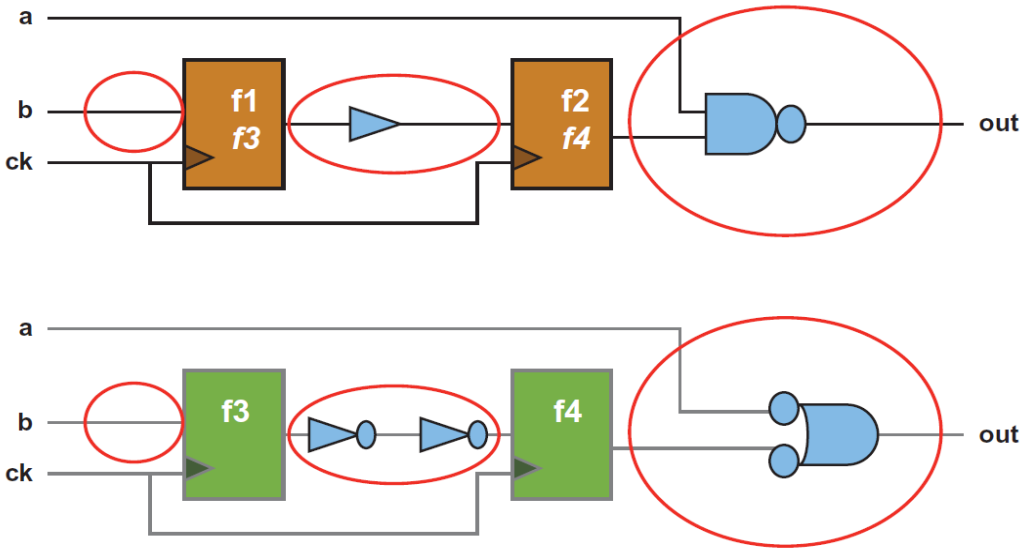
- Figure illustrates the general concept of key points and logic cone checking for FEV. The logic cones being compared are circled.
- Derived from this basic concept, there are several major notions of equivalence that are commonly used by current EDA tools: -

  - Combinational equivalence,
  - Sequential equivalence, and
  - Transactional equivalence.



Combinational Equivalence

- Combinational equivalence is the most mature FEV technique in the EDA industry; using this technique to compare RTL and schematic netlists is considered a requirement at most companies doing chip design.
- In this mode, two designs are claimed to be equivalent when all combinational logic between any pair of corresponding states are logically equivalent. In other words, there is a 1:1 correspondence between the state elements of the two models.
- Whenever the equivalence of a pair of state elements in the two models is checked, the tool uses the assumption that corresponding state elements in their fan-in cones will contain identical values at all points in time.
- In effect, every latch or flop in the two designs is being treated as a cut point, with verification only encompassing the logic between the state elements.



Circuit A	Circuit B	Type
a, b, Ck	a, b, Ck	Input
F1	F3	Flop
F2	F4	Flop
Out	out	Output

Combinational Equivalence

- This might seem like a huge limitation—but it arose in tandem with logic synthesis technology that has a similar structure. In other words, most EDA tools that synthesize netlists from RTL are also state-matching, guaranteeing (mostly) that each state element in the netlist will correspond to a state element in the RTL.
- In addition, the state-matching compromise makes the FEV problem much more tractable: FEV tools that use this method just need to analyze Boolean expressions, with no need to account for value changes over time. Thus, combinational FEV is the technique used in most RTL to netlist logic checkers today.
- If state elements are present, then not only are the outputs compared, but the logic cones driving each pair of corresponding states is checked individually. Since the RTL and netlist state elements directly correspond to each other, this check guarantees that the netlist is truly an implementation of the RTL.

Combinational Equivalence

- Some major limitations of this combinational equivalence type of equivalence. For example, what happens if you want the implementation to recode a finite state machine (FSM), using a different set of intermediate states to ultimately generate the same output?

- The application of combinational equivalence to FSMs is limited to comparing models that have equivalent sets of state elements. If this condition is not met, the combinational FEV tool will report the two models as nonequivalent.

- But in general, the state-preserving requirement is a stringent one, and there are many FEV scenarios in which it must be relaxed for effective verification.

Sequential Equivalence

- Sequential equivalence is also referred to as cycle-accurate equivalence by some vendors.
- With sequential FEV tools, we ask the question of whether two models will ultimately generate the same outputs at the same times based on an equivalent set of inputs, without requiring that internal state elements fully correspond.
- Instead of asking the question, “Is the property RTL model **output == reference** model output always true?” we ask the similar question, “Is the RTL model always equivalent to the reference model?”

Advanced Digital Design
Sequential Equivalence

- In fact, because good commercial sequential FEV tools have only appeared on the market very recently, it was common for many years for engineers to use FPV tools as simple sequential FEV tools in this way.
- They would create an enclosing module with both the SPEC (RTL) and IMP (gate level or RTL) model inside, create assertions that the outputs are equivalent, and run a standard FPV tool.
- However, now that there are tools available that specialize in sequential FEV, if you do have a reference model that is complete enough that you can just check equivalence instead of needing to write properties, it is usually better to use sequential FEV.
- This method can take advantage of engine improvements targeted at equivalence checking.

If you have both FPV and sequential FEV tools available, you should usually use the sequential FEV tools when trying to solve a non-state-matching model equivalence problem.

Sequential Equivalence

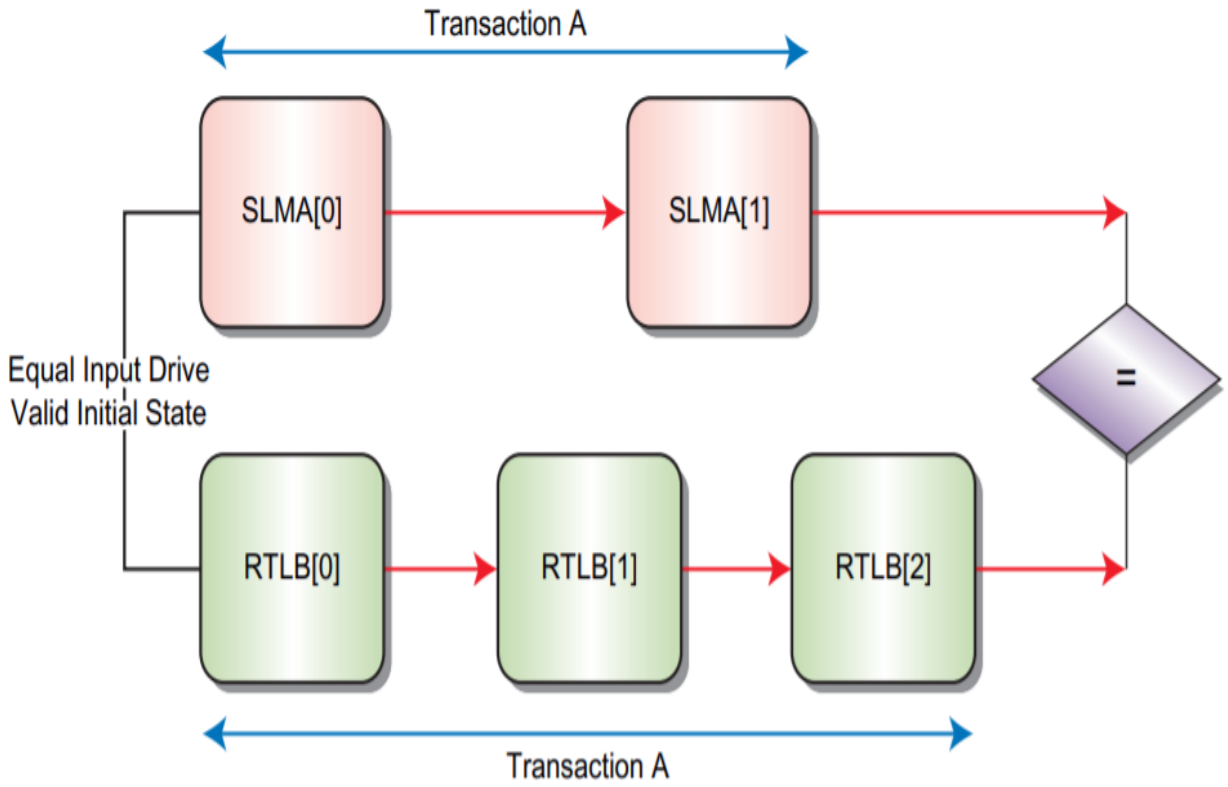
- Unlike combinational checkers, sequential equivalence checkers can verify designs with common functionality despite differences in: -
  - State Representation
  - Pipeline depth
  - Interface timing and protocols
  - Resource scheduling and allocation
  - Differences in data types
  - Clock gating and other power-saving features
- For example, when using sequential FEV, one of the designs under comparison might be a pipelined version of the other. We could prove the equivalence of the pure behavioral model of RTL to the complete pipeline implementation, and satisfy ourselves that they do indeed implement equivalent functionality.
- One common application of this methodology is during the equivalence check of an unpipelined and pipelined model in RTL. In the simplest case of comparison, one can assume that the same input patterns are applied to the corresponding inputs of both the designs, and the outputs are compared with some known delay after the RTL pipeline is completely filled.

Sequential Equivalence

- For example, when using sequential FEV, one of the designs under comparison might be a pipelined version of the other. We could prove the equivalence of the pure behavioral model of RTL to the complete pipeline implementation, and satisfy ourselves that they do indeed implement equivalent functionality.
- One common application of this methodology is during the equivalence check of an unpipelined and pipelined model in RTL.
- In the simplest case of comparison, one can assume that the same input patterns are applied to the corresponding inputs of both the designs, and the outputs are compared with some known delay after the RTL pipeline is completely filled.

Advanced Digital Design
Transaction Based Evidence

- This is another variant of sequential FEV. In some cases, one model will be highly abstract compared to the other, and its outputs will generally not match cycle-by-cycle, except for certain well-defined transactions.
- This means we need a looser notion of equivalence, based on checking that legal transactions are handled equivalently. The equivalence can be after a fixed number of RTL clock cycles, which could signify the transaction, or a completely unpredictable number based on the transaction completion.
- Hence, this notion is more generic and is a superset of all the equivalence modes: models that are not combinationally or sequentially equivalent may still be able to demonstrate transaction-based equivalence.



Transaction based Equivalence Check

Advanced Digital Design
Transaction Based Evidence

- Figure illustrates a high-level view of a transaction-based equivalence check. To further clarify what FEV is doing, let's examine a conceptual view of the "transaction space" of a typical design.
- This notion of equivalence is quite general and encompasses the previous notions of equivalence described in this section. This model of equivalence:
  - Does not assume that the amount of time (or the number of state transitions) required to complete one transaction will be the same for the RTL and the system-level design (SLM).
  - Denotes the end of a transaction by either a fixed number of RTL cycles or some kind of "data ready" handshaking protocol.
  - Assumes the user is able to specify a set of initial states (a partial assignment of values to the state-holding elements) for both the RTL and system-level design, which represents the model state at the start of a valid transaction.
  - Assumes the user is able to specify a set of states for both designs that correspond to the end of a transaction.

Transaction Based Evidence

- At the end of a transaction, the outputs of the two designs are compared and a check is made to ensure the ending state of both designs still satisfies the state invariants and constraints.
- Thus, this form of verification can handle cases where changes in the design abstraction level have only preserved behavior for certain well-defined transactions, rather than following the more stringent matching requirements of sequential FEV.

If a design change involves complex sequential changes that are beyond the capability of your sequential FEV tool, you may be able to verify it with transaction-based FEV.

Transaction-based FEV tools are a relatively new style of FV, and not very mature in the EDA marketplace

Advanced Digital Design

Commercial Tools for Equivalence Checking

- Cadence Conformal (LEC) (Logic Equivalence Check)
- Synopsys Magnellan

Advanced Digital Design

Demonstration

- Commercially available Electronic Design Automation

References

- This Presentation
- Online Class Lecture Explanation

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu

Advanced Digital Design

Dr. Sudeendra kumar K

Department of Electronics and Communication Engineering

ADVANCED DIGITAL DESIGN

Unit-III: PVT Conditions

Sudeendra kumar K
Department of Electronics and Communication Engineering

Advanced Digital Design

Contents

- PVT Conditions and Power Analysis

Operating Conditions

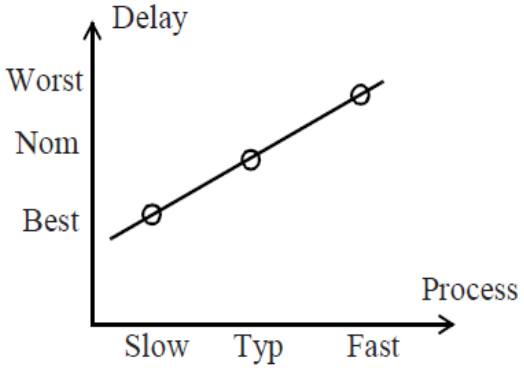
- STA and Power Analysis is typically performed at a specific operating condition.
- An operating condition is defined as a combination of Process, Voltage and Temperature (PVT).
- Cell delays and interconnect delays are computed based upon the specified operating condition.
- There are three kinds of manufacturing process models that are provided by the semiconductor foundry for digital designs: slow process models, typical process models, and fast process models.

Operating Conditions

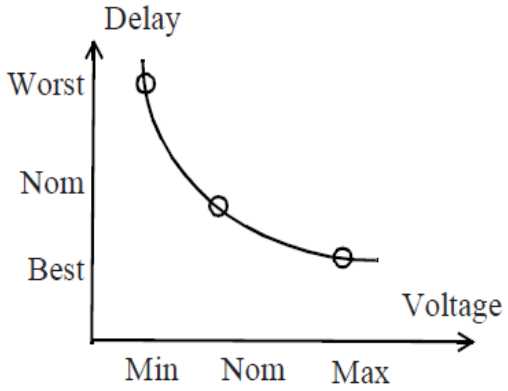
- The slow and fast process models represent the extreme corners of the manufacturing process of a foundry.

- For robust design, the design is validated at the extreme corners of the manufacturing process as well as environment extremes for temperature and power supply.

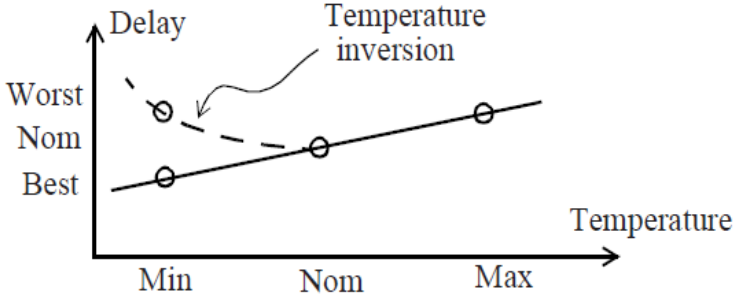
Advanced Digital Design
Operating Conditions



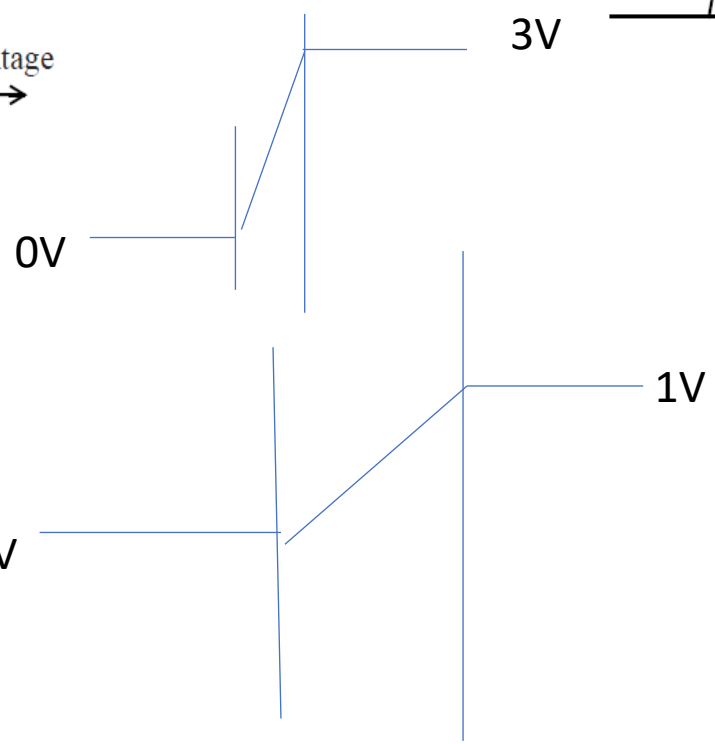
(a) Delay vs process.

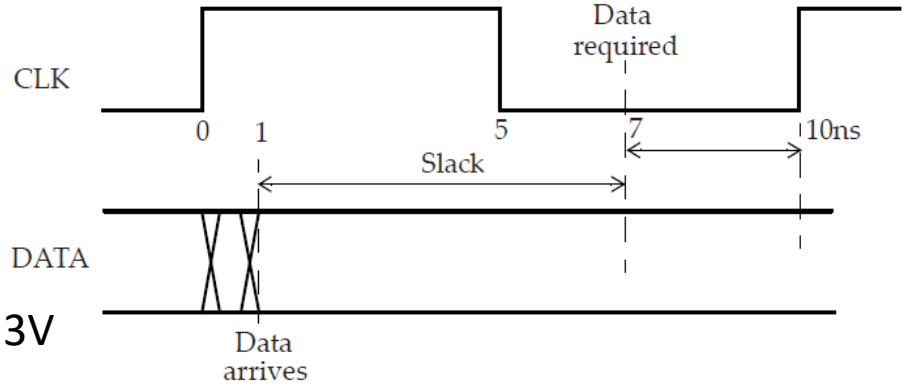


(b) Delay vs voltage.



(c) Delay vs temperature.





Slack Calculation: -
As per the diagram above:
Slack is 7-1 = 6 (Slow)
Slack is 7 – 3 = 4 (Typ)
Slack is 7 – 6 = 1 (fast)

Typical Standard Vdd = 1.2V
Min Voltage = 0.9V
Max Voltage = 1,4V

Operating Conditions: NBTI

- **WCS (Worst-Case Slow):** Process is slow, temperature is highest (say 125C) and voltage is lowest (say nominal 1.2V minus 10%). For nanometer technologies that use low power supplies, there can be another worst-case slow corner that corresponds to the slow process, lowest power supply, and lowest temperature.

- The delays at low temperatures are not always smaller than the de-lays at higher temperatures. This is because the device threshold voltage (Vt) margin with respect to the power supply is reduced for nanometer technologies.

Operating Conditions: NBTI (Negative Bias Temperature Inversion)

- In such cases, at low power supply, the delay of a lightly loaded cell is higher at low temperatures than at high temperatures.

- This is especially true of high Vt (higher threshold, larger delay) or even standard Vt (regular threshold, lower delay) cells.

- This anomalous behavior of delays increasing at lower temperatures is called temperature inversion.

Operating Conditions: Leakage Power Measurements

- The environment conditions for power analysis are generally different than the ones used for STA. For power analysis, the operating conditions may be:
- **ML (Maximal Leakage):** Process is fast, temperature is highest (say 125C) and the voltage is also the highest (say 1.2V plus 10%).
- This corner corresponds to the maximum leakage power. For most designs, this corner also corresponds to the largest active power.
- **TL (Typical Leakage):** Process is typical, temperature is highest (say 125C) and the voltage is nominal (say 1.2V). This refers to the condition where the leakage is representative for most designs since the chip temperature will be higher due to power dissipated in normal operation.

Different Corners cont'd

(a) *Process*: *Fast*—normally referred as FF (*FastN*, *FastP*) which refers to each MOS device (NMOS or PMOS) being at the fastest possible corner within the manufacturing process. At the FF process, both the *device-on* as well as the -*device-off* currents are high. The *on* currents map to the active currents and the *off* currents map to the leakage currents. Thus, both active and leakage power are high for the FF process condition.

(b) *Power supplies*: Maximum allowed for the design. In many cases, this may correspond to 10% above the nominal supply values. This refers to all the power supplies used for the design. In general, the power supply values are uncorrelated; however any correlation should be considered for analysis. The active power as well as leakage power increases with power supply.

(c) *Temperature*: Maximum junction temperature allowed for the design. In many cases, the maximum temperature allowed is 125°C. The leakage power has a superlinear (almost exponential) dependence on temperature. In most digital designs, the active power also increases with temperature though the increase is much smaller than that of leakage power.

Based upon the above, the power analysis using the above PVT corner will correspond to the worst (or maximum) power dissipation for the ASIC. A designer may also want to obtain the nominal power dissipation in which case, additional power analysis using nominal PVT libraries can be performed.

References

- This lecture slides and explanation given in online class

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu

Digital System Design

Dr. Sudeendra kumar K

Department of Electronics and Communication Engineering

DIGITAL SYSTEM DESIGN

Introduction to Standard Cell Libraries

Sudeendra kumar K
Department of Electronics and Communication Engineering

What you expect from the foundry for semicustom design

- **DDK (Digital Design Kit): -** A digital design kit is a collection of physical views of certain functions that a fabrication house is capable of processing at a given technology node.
- Coupled to this are the other physical (read place and route) views as well as the logical, temporal, power, noise, and test views that represent these various functions for (usually some, maybe most) of the various engineering design automation tools available on the market.
- Moreover, the typical DDK release will be for what has become known as a “Frankenstein” flow, with Cadence logical and LEF (library exchange format) views and Synopsys timing, power, and noise views and supported in the physical design kit (PDK), by Mentor verification decks, although there are exceptions to each instance, with Cadence, Synopsys, and Mentor being “the big three” EDA companies.

At the very minimum, a DDK will contain the following

- Graphical Database System Stream-file format (GDS) (physical), which are views of one or more standard cell (std-cell) libraries, usually including IO-specific libraries, which can be used on designs that are destined to be processed in a particular fabrication house.
- These views are a binary format, viewable in any one of several GDS viewers. ASCII based human readable formats of GDS can be created through various EDA tools.
- Gate-level SPICE (simulation program with integrated circuit emphasis) netlist of each cell in the various libraries, SPICE being a cross between a physical view (which it represents) and a simulatable view, which is one of its two major purposes, and a verification source, which is its other major purpose. SPICE is ASCII based and easily human readable.

What you expect from the foundry for semicustom design

- Some further physical representation (possibly a physical LEF, an ASCII-based and human-readable description of the place and router pertinent parts of the GDS).
- This further physical representation is used by automatic place and route tools, in conjunction with a logical LEF, usually part of the PDK that contains information on layer definitions that the router can use during design place and route.
- Timing, power, and noise views, which are typically but not necessarily referred to as liberty files, a name that comes from the extension that is most often used (“lib”) on versions of these ASCII files that were originally purely for Synopsys tools. Being ASCII, they are easily readable.

What you expect from the foundry for semicustom design

- More so, most liberty writers tend to output highly structured versions of the format, which further aids human understanding on reading.
- The Backus-Naur Format (BNF) for these files have since been “open sourced”—that is, they can usually be read and understood by most EDA timing tools.
- One aside: Synopsys has not frozen the BNF for its liberty files and thus care must be taken with every new Synopsys release in order to keep previous versions of liberty files from becoming antiquated.

What you expect from the foundry for semicustom design

- Logical and test views, typically being slightly adjusted versions of the same logical format, but with one geared toward aiding logical simulations and the other geared to representing actual internal nodes of a stdcell (or IO) function that allows for accurate tracking of “reachability and observability” for internal fault tracking.
- These views may come in one or both of two major logical tool formats (Cadence’s open-source VERILOG file or IEEE’s open-source VHDL format). They are both ASCII based and human readable.
- There may be other views, especially white papers, verification or validation documentation, and release-tracking documents, all in some form of human readable format. Sometimes other, more EDA-tool-specific views can also be included.

What you expect from the foundry for semicustom design

- Along with the preceding major views, support for special tools may be added to a DDK release by a fabrication house. These tools can include RAM/ROM memory compilers.
- Once installed on a system, RAM/ROM memory compilers, at the press of a few buttons, automatically generate all of the preceding formats for a specific RAM/ ROM configuration supported by the fabrication house.
- The timing, power, and noise view is initially a machine-estimated version. Typical fabrication-house vendors, however, offer the option of characterizing the specific RAM/ROM in order to allow cleaner design closure of a design that uses that specific RAM/ROM, and these fabrication house usually offer this either free or for a small fee.

What you expect from the foundry for semicustom design

- The reason that RAM/ROM compilers exist as opposed to a long list of usable RAM and ROM instances is that there is such a larger list of possible combined number of words, bits, and column-multiplexer (MUX) instances that could be chosen by design teams, all of which would therefore otherwise need to be supported.
- Finally, there may be certain specific circuits (or analog subcircuit pieces, such as various layer resistors of multilayer capacitors or diode or PNP or NPN structures that can be used to build analog function in the particular technology node and in the specific fabrication house).
- Some of these, which are dependent on the technology node, will be custom and uneditable or only marginally editable structures (the more complex and deep the technology node, the more likely that this is so). Some will be just Boolean layer definitions.
- Most will have rather pared-down subsets of the previously mentioned views, just enough to allow some amount of simulation (usually in SPICE) and design-level verification (including netlist capability).

What you expect from the foundry for semicustom design

- Deeper technology nodes will tend to have stricter requirements for the physical views than higher technology nodes.
- The real reason for this is that as geometries get smaller, they tend to go from drawn silicon shapes being transferred to the mask through lithographic treatments such as serifs at corners of polygons through optical-proximity correct techniques, through phase shifting, and to exact set widths and spaces.
- For the deepest technologies, what is drawn on the mask can bear little visible relation with what is desired on silicon. In those instances, the operative order is “Don’t touch.”
- Trust that the fabrication house understands better than the design team what certain polygons need to be, where they need to be, and why they need to be.

What you expect from the foundry for semicustom design

- However, for the higher nodes, even those in the 90-nanometer range, which includes some phase-shifted mask and some strong optical proximity correction, it is worth pursuing with the fabrication house when a question on a physical view arises.
- For higher technology nodes than that, there is a good chance that you can find optimizations that the fabrication house has left “on the table.” In addition, it is highly advantageous at these higher technology nodes to attempt to do just that.
- The reasoning is that these nodes have been “in the marketplace” for such a long time that every design house that wants to compete in a market will have access to the same generic package.
- Assuming that all design centers make the same decision on technology node (which is a valid assumption given that they would be privilege to the same marketing information), they will produce roughly the same size product in the same rough timeline.

Standard Cell Library

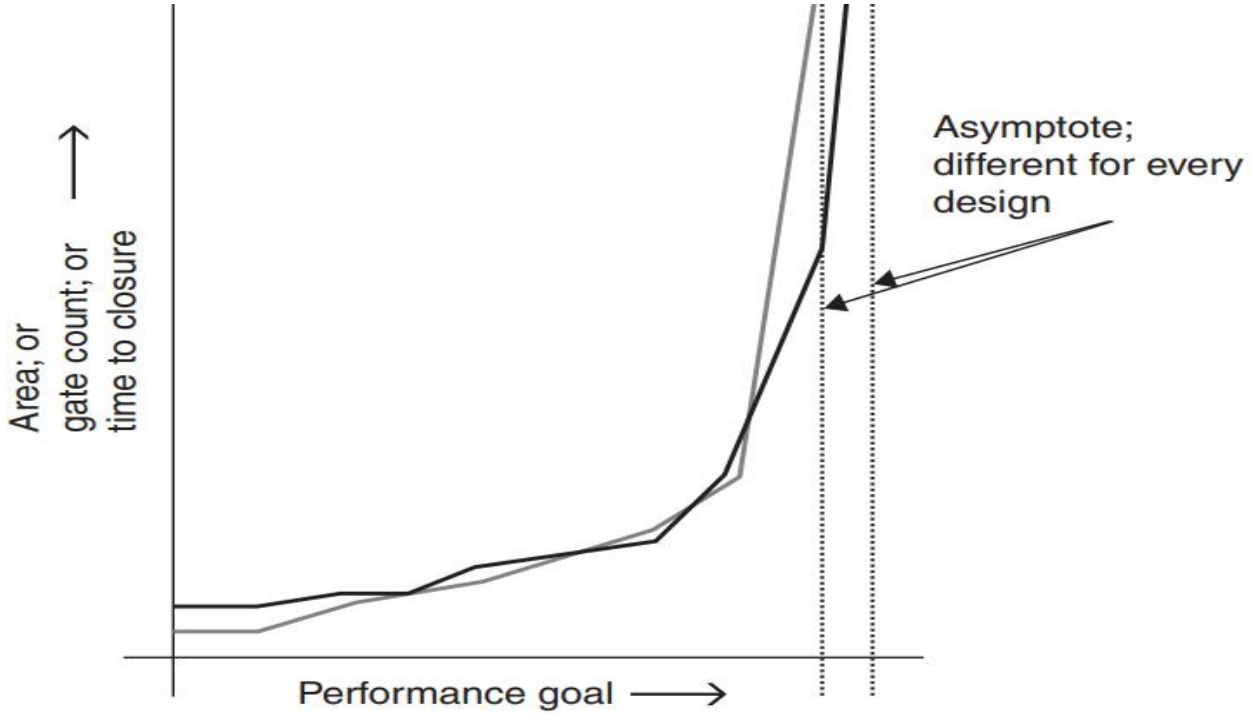


FIGURE 2.1 Showing the Asymptotic Nature of Performance Limits of Libraries and Technologies. For many applications that do not stretch the limits of a library's abilities, the ability of the library to achieve closure (in terms of time or density or power) is not significantly different from any other library. Only at the extreme edges of the ability of libraries do they show dramatic differences during design closure. Typically, such extreme limits are not nearly the same limits of other libraries designed for other design goals, although they are shown here to be nearly coincident.

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu

Digital System Design

Dr. Sudeendra kumar K

Department of Electronics and Communication Engineering

DIGITAL SYSTEM DESIGN

Introduction to Standard Cell Libraries

Sudeendra kumar K
Department of Electronics and Communication Engineering

Standard Cell

- Digital design kit (DDK) vendors, most often representing fabrication houses, usually offer them in a family of several hundred of the various functions and strengths.
- The actual choosing of which stdcells in a family to use in a design and the actual placing of these at various locations on a piece of silicon, both of which can be done manually, is usually done by software provided by various engineering design automation (EDA) vendors.

Standard Cell

- It will be beneficial to understand that the actual stdcell layouts, as represented by the various rectangles in Figure 2.2, are DDK view.
- The clear rectangle surrounding the shapes in the figure can be described as the stdcell outline.
- Although there may be shapes associated with the stdcell that extend beyond the rectangle (such as the NWELL), that rectangle represents the regular placeable structure of the stdcell.

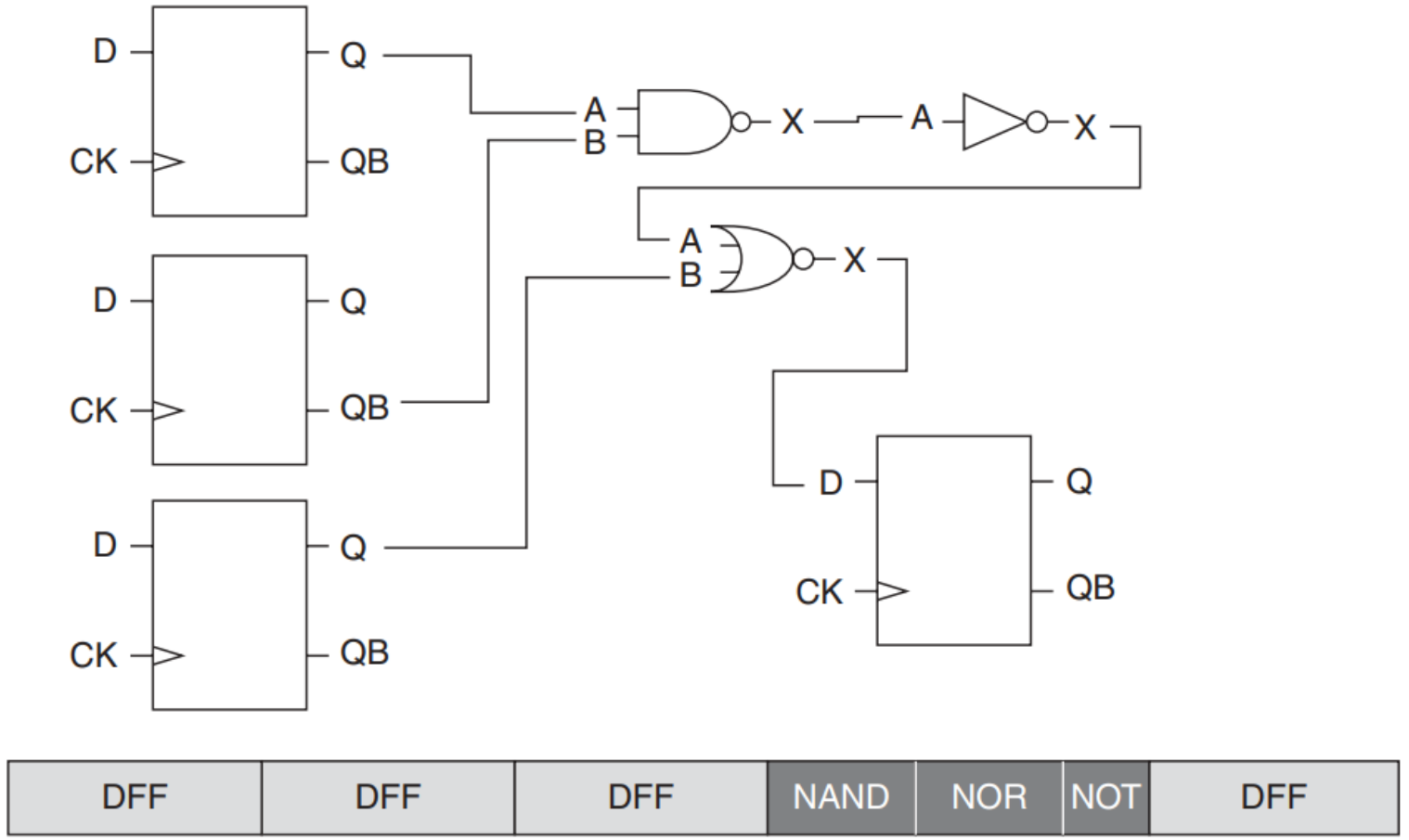


FIGURE 2.2 A Small Circuit of Stdcells and a Possible Layout of Same in a Single Row. Note that the relative sizes of the rectangles that represent the actual physical views of the various elements in the circuit are fairly close to actual ratios for those functions. Also note that the cells are not necessarily placed in any given order similar to the logic circuit.

Standard Cell

- Such rectangles (or cell outlines or boundaries) are of consistent height (or multiples thereof) and consistent width (or multiples thereof).
- Other, more complicated stdcells will have more P-channel and N-channel transistors within their cell boundaries, but those cell boundaries will always be of given multiples of the basic stdcell boundary height and width.
- This feature allows them to “stack” in both the horizontal and vertical grid on a chip during place and route.

Combinational Cells: Single Stage

- The leading EDA synthesis tool over the last two decades, Synopsys's Design Compiler, requires at least one storage element, one inverting element, one combining element, and one tri-stating element (even if the RTL does not need a tristate function) to be in the stdcell library so that the tool can properly synthesize the netlist.
- The combining and inverting element could be combined as a NAND function or a NOR function. Therefore, the smallest stdcell offering, as illustrated in Figure 2.4, must contain a flip-flop, a tri-state, and a NAND or NOR.

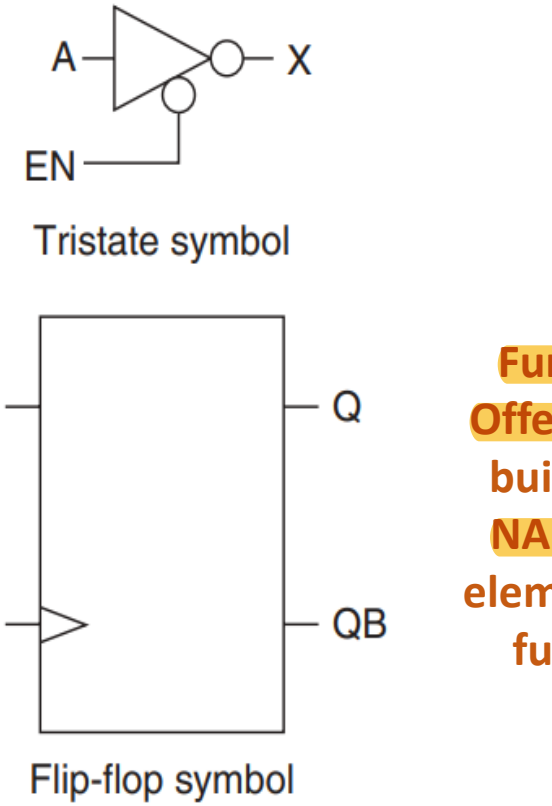


FIGURE 2.4 Minimum Set of Functions Required in a Synthesis Offering. Any logical function can be built with various combinations of NAND (or NOR) gates. The storage element and the tristate element are further required by the industry standard synthesis tool.

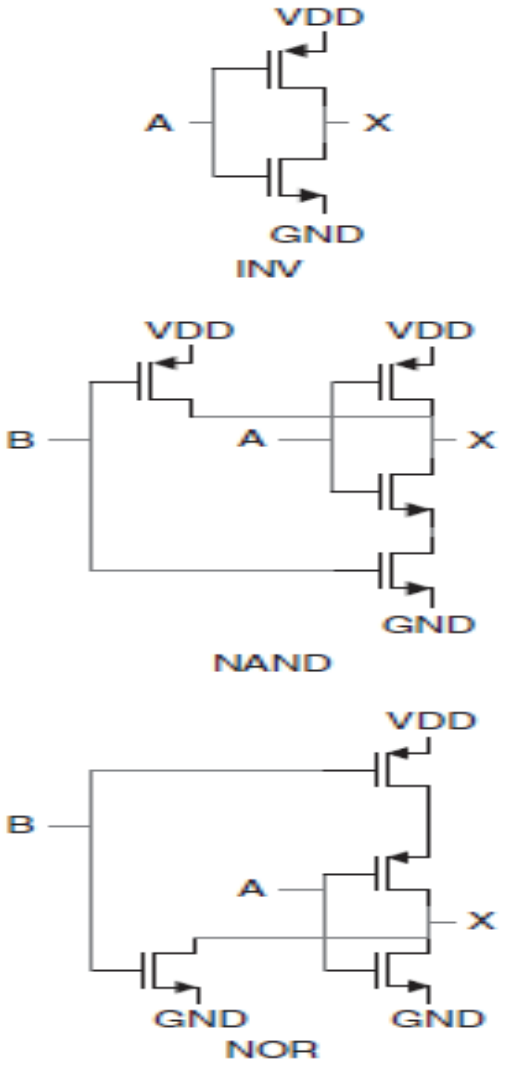
Standard Cell

- Such stdcell offerings can be artificially built either by removing other elements in a more extant library offering or by adding the tokens “Don’t touch” and “Don’t use” to them.
- These are Synopsys constructs that prevent stdcells that are present in a Synopsys liberty file from being modified after the fact (the “Don’t touch” construct), or instantiated before the fact (the “Don’t use” construct), in a synthesized netlist, and then to watch Design Compiler synthesize some representative (for your design environment) circuits.
- The results of an experiment, doing just as described previously for a four-bit adder circuit, has been added.
- However, no viable stdcell library on the market contains just these three elements.

Standard Cell

- Instead, most stdcell offerings on the market today contain 300–400 elements representing 3–4 drive strengths for somewhere in the realm of 70–80 different combinational logical functions (and a set of sequential storage elements).
- These numbers, however, can be widely variable in some offerings. Some libraries offer more than 1,000 elements, a half dozen drive strengths, or 120 or more logical functions singly or in combination.
- In addition, many current stdcell library offerings contain high-performance or low-power options that either are their own separate sub-libraries or are part of the offering as a whole. These can be “high-Threshold Voltage (Vt)” (slower, less-power-hungry versions of the library elements) and “low-Threshold Voltage (Vt)” (hotter but faster versions).
- Some of these versions are designed so that they can be used intermingled with cells of the same basic architecture but of different Vt.

Standard Cell



- The number of signals that are being combined in a logic function is limited by the number of transistors that can be stacked between a power or ground rail and the output.
- Stacking too many will not allow sufficient gate drain potential to allow the transistors to properly turn on or off completely (or efficiently).
- In addition, as stack reaches the edge of this potential, the last transistor out will tend to become an extremely slow transitioning timing arc.
- For modern deep submicron technologies, a three stack (as many as three transistors between a rail and an output) is typically the most that can be handled. This limits the number of functions that can be handled.

Standard Cell

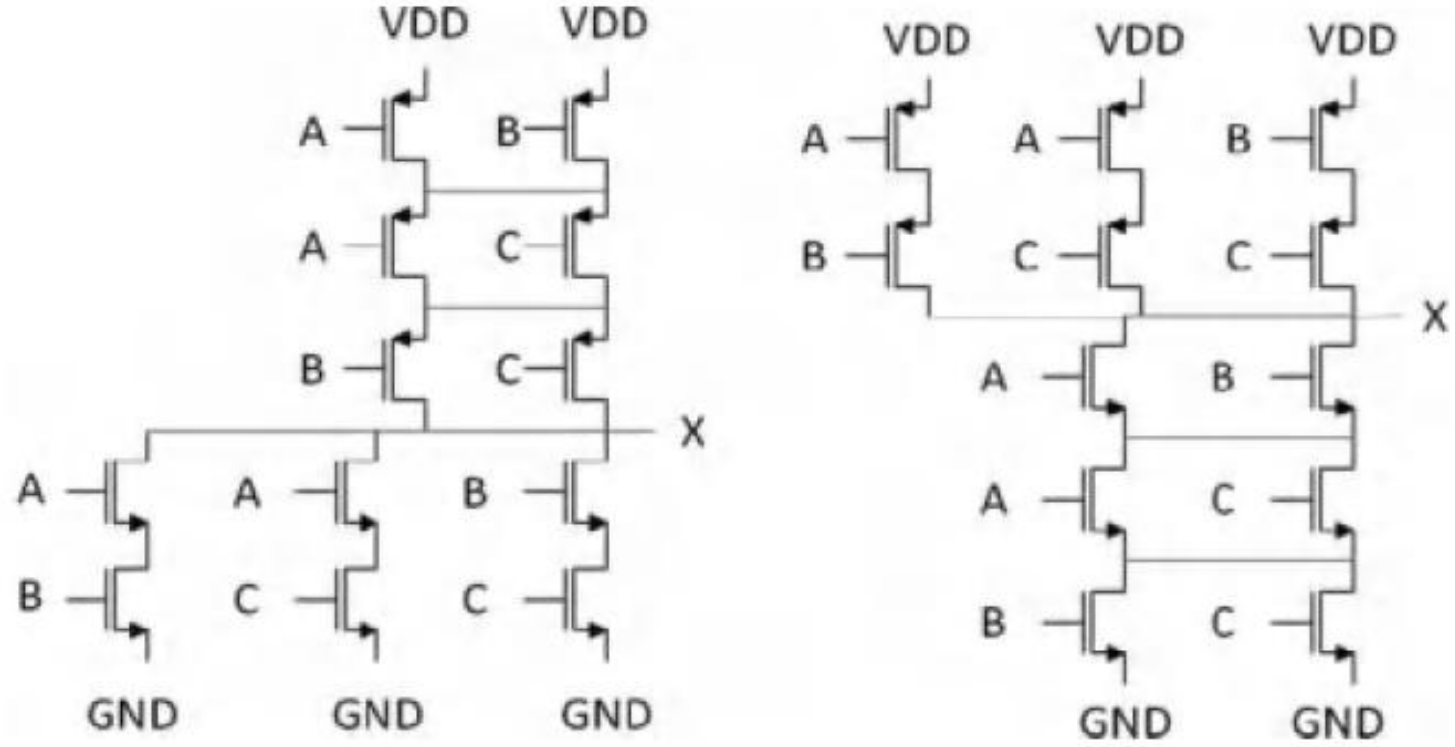


FIGURE 2.7 AOI222 (Left) and OAI222 (Right) Implementations of MAJ2 Function. Note that the one circuit has a three-stack P-channel and a two-stack N-channel pattern whereas the other circuit has the exact opposite.

Standard Cell

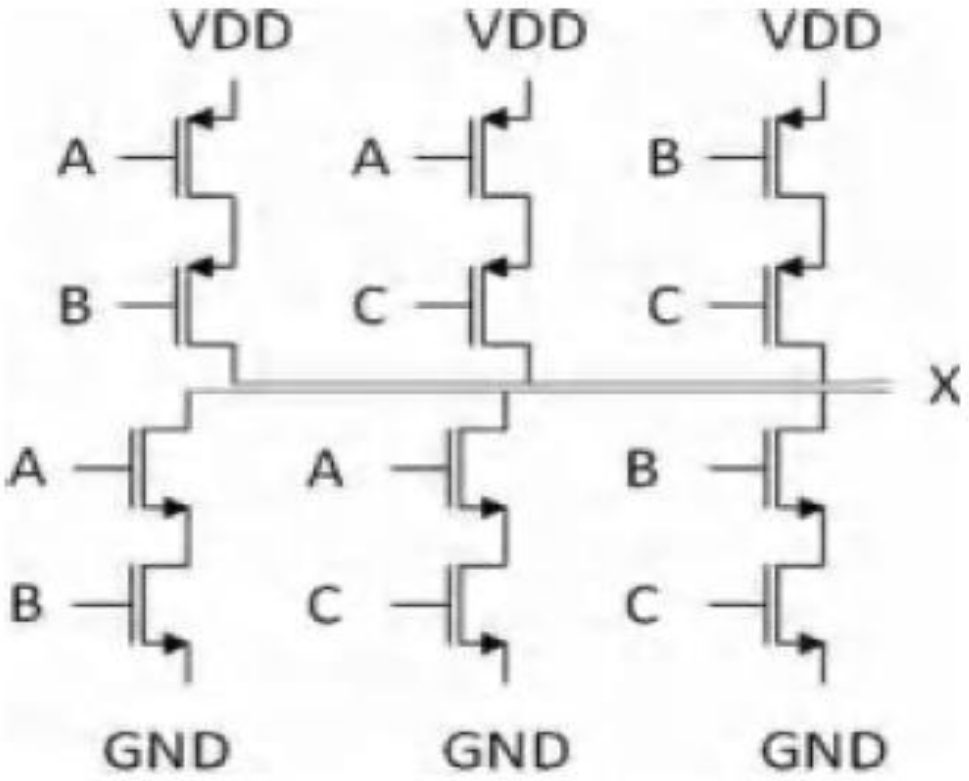


FIGURE 2.8 AOI222 N-Channel and OAI222 P-Channel Combined MAJ2 Function (Two-Stack). Replacing the three stack with the similar function two stack in only one or the other of the P-channel or N-channel while leaving the opposite two-stack side alone.

Standard Cell

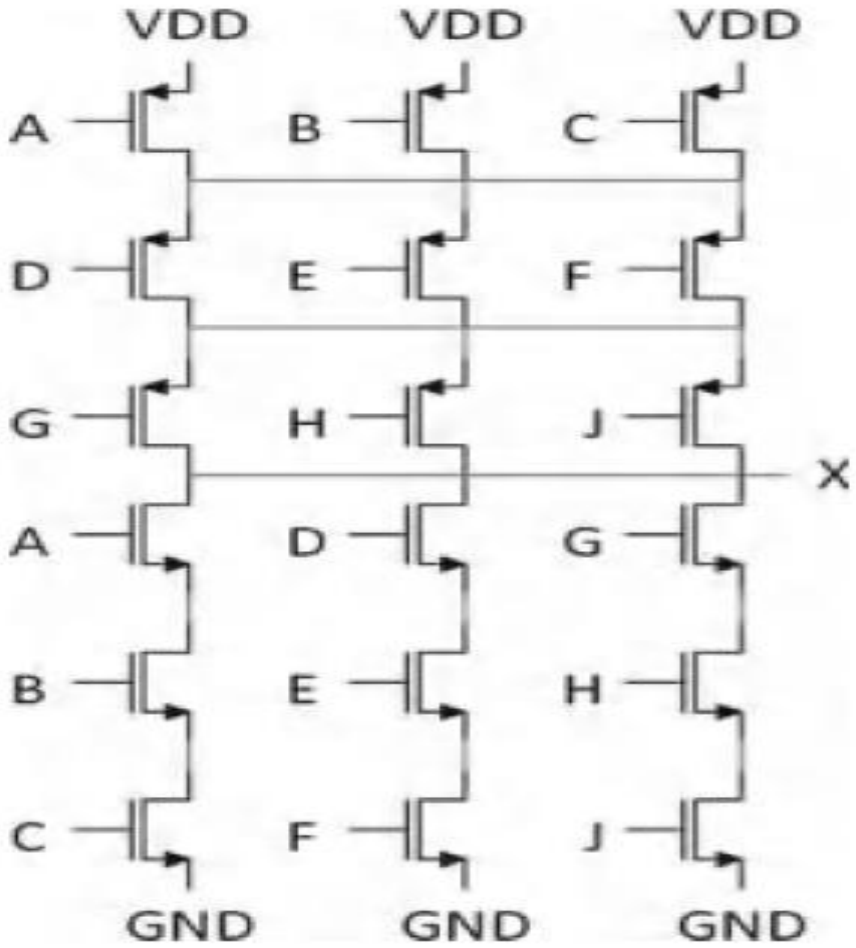


FIGURE 2.10 AOI333: Usually Present in a Stdcell Library and the Basis of a Large Family of Potential Additional CMOS Functions. This is one of the two largest "safe" three-stack Boolean functions from which multiple families of more complex logic but simpler circuit functions can be derived. Note the serial N-channel but serial and parallel P-channel pattern.

Standard Cell

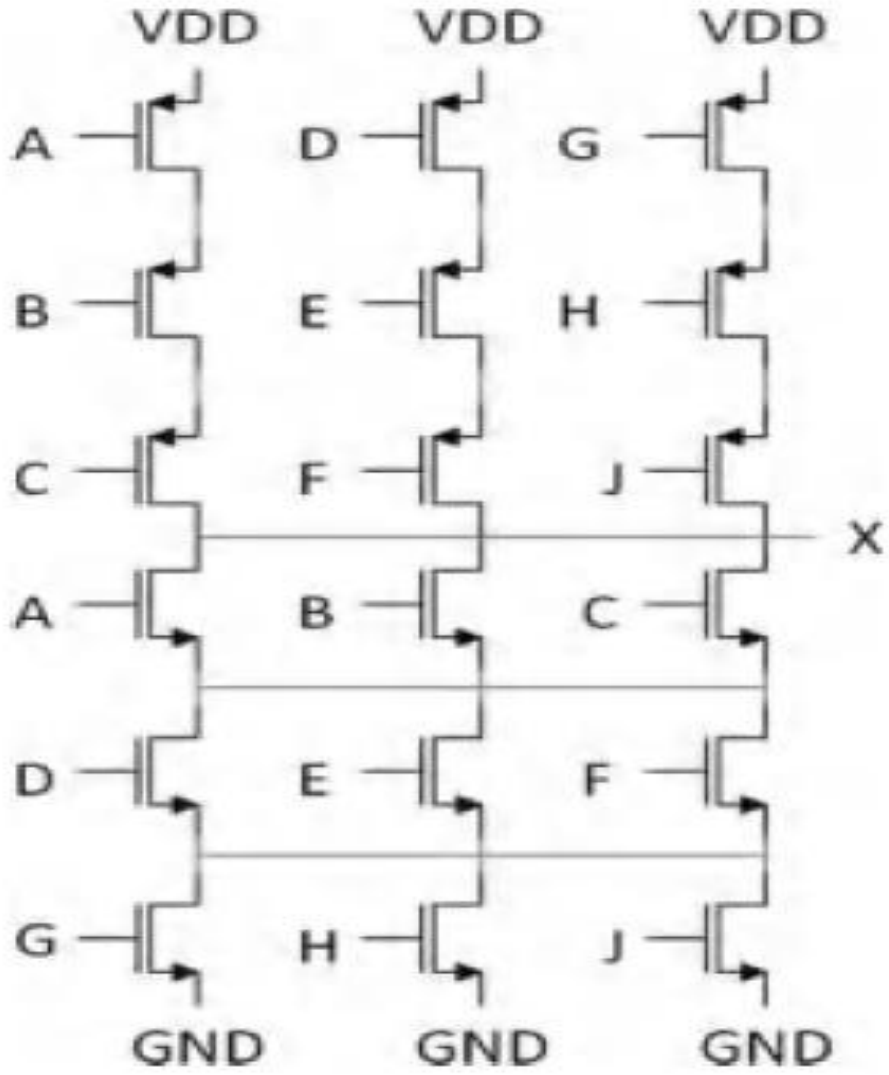


FIGURE 2.11 OAI333: Usually Present in a Stdcell Library and the Second Basis of a Large Family of Potential Additional CMOS Functions. This is the second of the two largest "safe" three-stack Boolean functions from which multiple families of more complex logic but simpler circuit functions can be derived. Note the serial P-channel but serial and parallel N-channel pattern.

Digital System Design

Standard Cell

TABLE 2.1 Basic One-, Two-, and Three-Stack CMOS Functions. These can be accomplished using single-stage implementation of either the basic NAND or basic NOR pattern.

Name	Function	Note
INV	A_	
NAND2	(AB)_	
NAND3	(ABC)_	3 stack
NOR2	(A + B)_	
NOR3	(A + B + C)_	3 stack
AOI21	(AB + C)_	
AOI22	(AB + CD)_	Can be used as a decoded MUX.
AOI211	(AB + C + D)_	3 stack
AOI221	(AB + CD + E)_	3 stack
AOI222	(AB + CD + EF)_	3 stack
AOI31	(ABC + D)_	3 stack
AOI32	(ABC + DE)_	3 stack
AOI33	(ABC + DEF)_	3 stack
AOI311	(ABC + D + E)_	3 stack
AOI321	(ABC + DE + F)_	3 stack
AOI322	(ABC + DE + FG)_	3 stack
AOI331	(ABC + DEF + G)_	3 stack
AOI332	(ABC + DEF + GH)_	3 stack
AOI333	(ABC + DEF + GHI)_	3 stack
OAI21	((A + B)C)_	
OAI22	((A + B)(C + D))_	
OAI211	((A + B)CD)_	3 stack
OAI221	((A + B)(C + D)E)_	3 stack
OAI222	((A + B)(C + D)(E + F))_	3 stack
OAI31	((A + B + C)D)_	3 stack
OAI32	((A + B + C)(D + E))_	3 stack
OAI33	((A + B + C)(D + E + F))_	3 stack
OAI311	((A + B + C)DE)_	3 stack
OAI321	((A + B + C)(D + E)F)_	3 stack
OAI322	((A + B + C)(D + E)(F + G))_	3 stack
OAI331	((A + B + C)(D + E + F)G)_	3 stack
OAI332	((A + B + C)(D + E + F)(G + H))_	3 stack
OAI333	((A + B + C)(D + E + F)(G + H + J))_	3 stack
TBUFE	A_ if EN=1 and EN_=0, tri-state if EN=0 and EN_=1, undefined otherwise	Admittedly, can be used as an open-source or an open-drain function as well, depending on other combinations of en and en_; rarely seen as stand alone in a commercially available library.

Standard Cell

TABLE 2.2 Two Special Yet Simple Two- and Three-Stack CMOS Functions That Demonstrate the Boolean Flexibility of the Technology. These are special cross combinations of both the basic NAND pattern and basic NOR pattern as explained in the text.

Name	Function	Note
MAJ3	(AB + BC + AC)_	Multiple implementations
MAJ5	(ABC + ABD + ABE + ACD + ACE + ADE + BCD + BCE + BDE + CDE)_	Only one valid implementation in 3 stack.

- These are the two-of-three majority vote function, sometimes known as the MAJ3 function; and the three-of-five majority vote function, sometimes known as the MAJ5 function.
- The first can be implemented using an AOI222 or an OAI222 or a cross of the two functions, but the second can only be done using the combination.

Standard Cell

- Some creative design usages for MAJ5 cells—for instance, in data-communication error detection and correction techniques and possibly in some low-power bus-inversion circuits and techniques—but typical stdcell libraries do not have MAJ5 functions. These cells require 60 transistors that further require 30 polyfingers in order to implement.
- Unless they are added for specific majority vote or data-stream validation reasons, their addition should be questioned.

Digital System Design
Standard Cell: Multistage

TABLE 2.5 Basic Two-Stage CMOS Functions. These are the usual two-stage families on which the most popular synthesis engines rely.

Name	Function	Note
BUF	A	Double inversion
AND2	AB	
AND3	ABC	
OR2	A + B	
OR3	A + B + C	
XOR	AB_ + A_B	
XNR	AB + A_B_	
MUX	AS + BS_	
MXB	(AS + BS_)_	
NAND3i	(ABC_)_	Can be done with 3-stack NOR (or less) into a 3-stack NAND (or less).
NAND4ii	(ABC_D_)_	Can be done with 3-stack NOR (or less) into a 3-stack NAND (or less).
NAND5iii	(ABC_D_E_)_	Can be done with 3-stack NOR (or less) into a 3-stack NAND (or less).
NOR3i	(A + B + C_)_	Can be done with 3-stack NAND (or less) into a 3-stack NOR (or less).
NOR4ii	(A + B + C_ + D_)_	Can be done with 3-stack NAND (or less) into a 3-stack NOR (or less).
NOR5iii	(A + B + C_ + D_ + E_)_	Can be done with 3-stack NAND (or less) into a 3-stack NOR (or less).
NAND2i	(AB_)_	Can be done with 3-stack NOR (or less) into a 3-stack NAND (or less).
NAND3ii	(AB_C_)_	Can be done with 3-stack NOR (or less) into a 3-stack NAND (or less).
NAND4iii	(AB_C_D_)_	Can be done with 3-stack NOR (or less) into a 3-stack NAND (or less).
NOR2i	(A + B_)_	Can be done with 3-stack NAND (or less) into a 3-stack NOR (or less).
NOR3ii	(A + B_ + C_)_	Can be done with 3-stack NAND (or less) into a 3-stack NOR (or less).
NOR4iii	(A + B_ + B_ + D_)_	Can be done with 3-stack NAND (or less) into a 3-stack NOR (or less).

Standard Cell

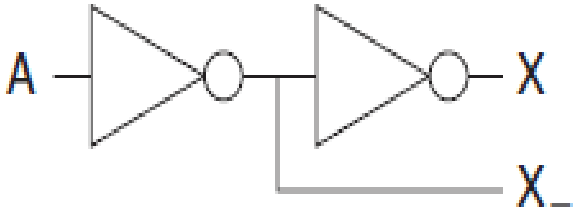


FIGURE 2.14 Once Common but Now Fairly Forbidden Two-Stage CMOS Stdcell Function. Because the delay to the last output is dependent on the loading of the intermittent stage, it is best to not have such cells in a non–"expert use only" library release. However, this can be eliminated by the general rule of only allowing multiple outputs on sequential cells.

Standard Cell: Sequential Cells

- For pure synthesis purposes, the flip-flop offering can be limited to D-type flip-flops only (Design Compiler does not synthesize JK flip-flops and uses D-type flip-flops for toggle flipflops).
- However, every design team eventually comes across a legitimate need for a JK or toggle flip-flop and can add one. In addition, design teams usually require many integrated functions to be added to the otherwise limited flip-flop offering.
- With the preceding description in mind, Table 2.6 gives a useful but not necessarily function-priority-ordered complete set of flip-flop stdcells.
- For typical design practice, where all flip-flops need to be scan capable, this list can be further paired down to just the scan flip-flops. However, legitimate reasons exist to have flip-flop circuits that do not contain the added MUX-D.

Standard Cell

TABLE 2.6 Fairly Complete Set of Common Flip-Flops and Latches (Both Scan and Nonscan Versions), LSSD Functions Excluded. Although other sequential functions certainly do exist, these make up the more useful set.

Name	Function	Note
DFFP	Positive clock D flip-flop	
DFFRP	Positive clock D flip-flop with asynchronous reset	
DFFSP	Positive clock D flip-flop with asynchronous set	
DFFRSP	Positive clock D flip-flop with asynchronous set and reset	There is an inherent race condition between asynchronous set and asynchronous reset: designs need to be warned not to rely on coming out in a known state if they are both going inactive at the same time.
DFFCP	Positive clock D flip-flop with synchronous clear	Synchronous clear also works as a data enable.
DFFEP	Positive clock D flip-flop with clock enable	
SDFFP	MUX-D positive clock D flip-flop	
SDFFRP	MUX-D positive clock D flip-flop with asynchronous reset	
SDFFSP	MUX-D positive clock D flip-flop with asynchronous set	
SDFFRSP	MUX-D positive clock D flip-flop with asynchronous set and reset	There is an inherent race condition between asynchronous set and asynchronous reset: designs need to be warned not to rely on coming out in a known state if they are both going inactive at the same time.
SDFFCP	MUX-D positive clock D flip-flop with synchronous clear	Synchronous clear also works as a data enable.
SDFFEP	MUX-D positive clock D flip-flop with clock enable	
DFFN	Negative clock D flip-flip	
SDFFN	MUX-D negative clock D flip-flip	
JKFFP	JK positive clock flip-flop	
JKFFN	JK negative clock flip-flop	
LATP	Positive nonscan lat	
LATN	Negative nonscan lat	
SLATP	Positive scan lat	
SLATN	Negative scan lat	

Standard Cell: Sequential Cells

- For all of these sequential cells, several possible circuit designs exist.
- As an example, a flip-flop can be designed as a “transmission gate” flip-flop, with the MUX at the front of the master and slave latches being accomplished by some combination of transmission gates and tri-statable inverters, or it can be designed as a “Boolean” flip-flop, with the MUX at the front of the latches being accomplished with Boolean functions.
- The particular circuit design that is used should depend on the primary goal of the library of which the sequential cell is part

Standard Cell: Sequential Cells

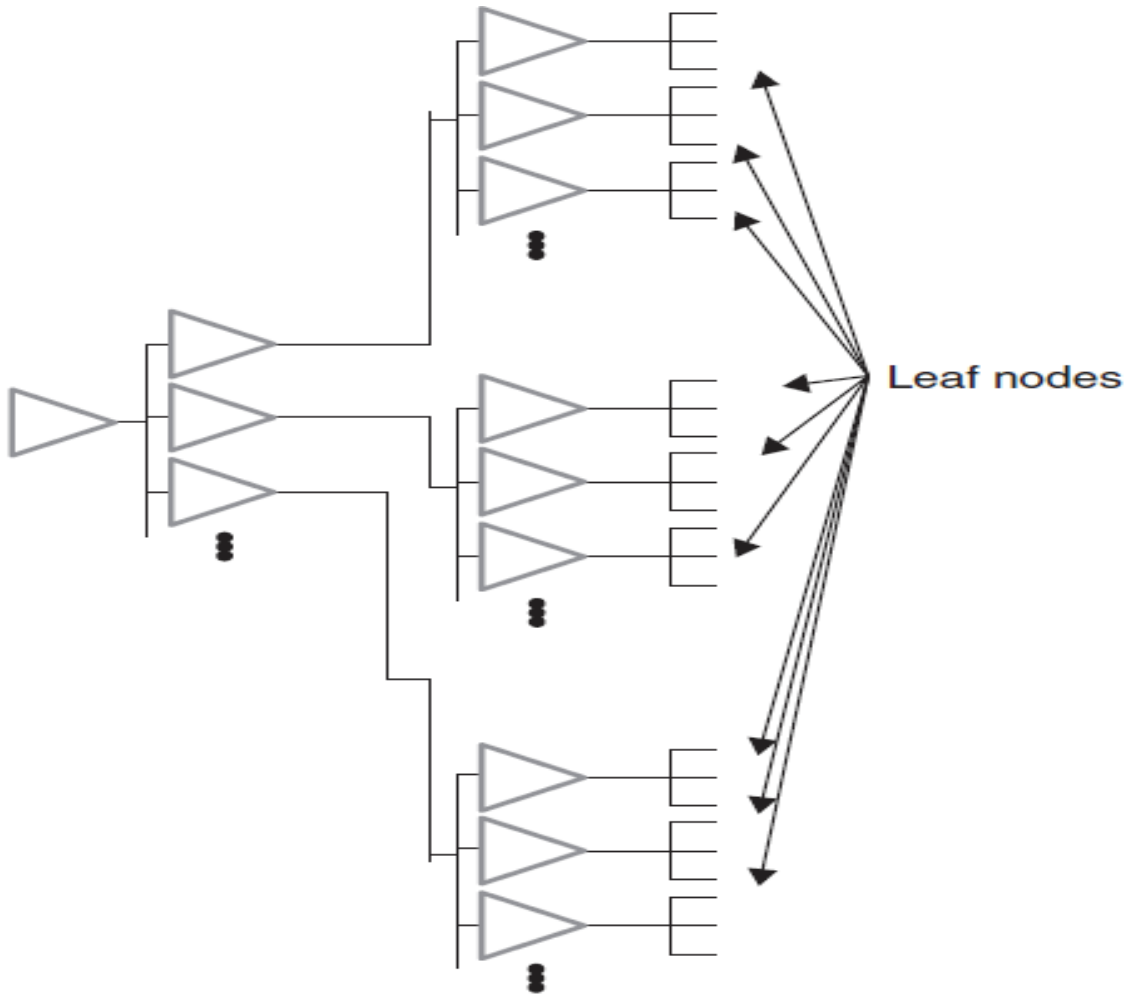
- A transmission-gate flip-flop will tend to be faster (with minimal setup and hold constraints) but will take more space to layout (because of the extra routing required to accomplish the opposing P-channel and N-channel gating within the transmission gates).
- A Boolean flip-flop does not suffer from this need for the more complicated routing of the transmission gates, but it tends to be slower with longer setup and hold constraints.
- Hence, it may make sense to have examples of both types within the same library. The reason for this is that few of the various cones of logic tend to be true long paths in a typical ASIC, requiring the fastest setup constraints and propagation delays.
- As a result, if both types of flip-flops are available, then most cones of logic can use the denser Boolean flip-flops, whereas only the true long paths that truly need them will use the transmission-gate flip-flops.

Standard Cell: Metastability

- Certain stdcell design techniques can make the flipflops and latches less prone to having this occur.
- These include using significantly different strengths on one of the two inverters in the loop as opposed to the other or using significantly different P-channel versus N-channel lengths in the two transistors of at least one of the inverters, thus changing the voltage when it switches well away from the usual halfway to rail.
- However, metastability happens, and no storage element is immune to it. The best way to handle it is to ensure that when the cell is used in a given design, the signals feeding the data input of the storage element occur significantly far away from the clock edge and are stable for enough time that metastability does not occur.

Standard Cell: Clock functions

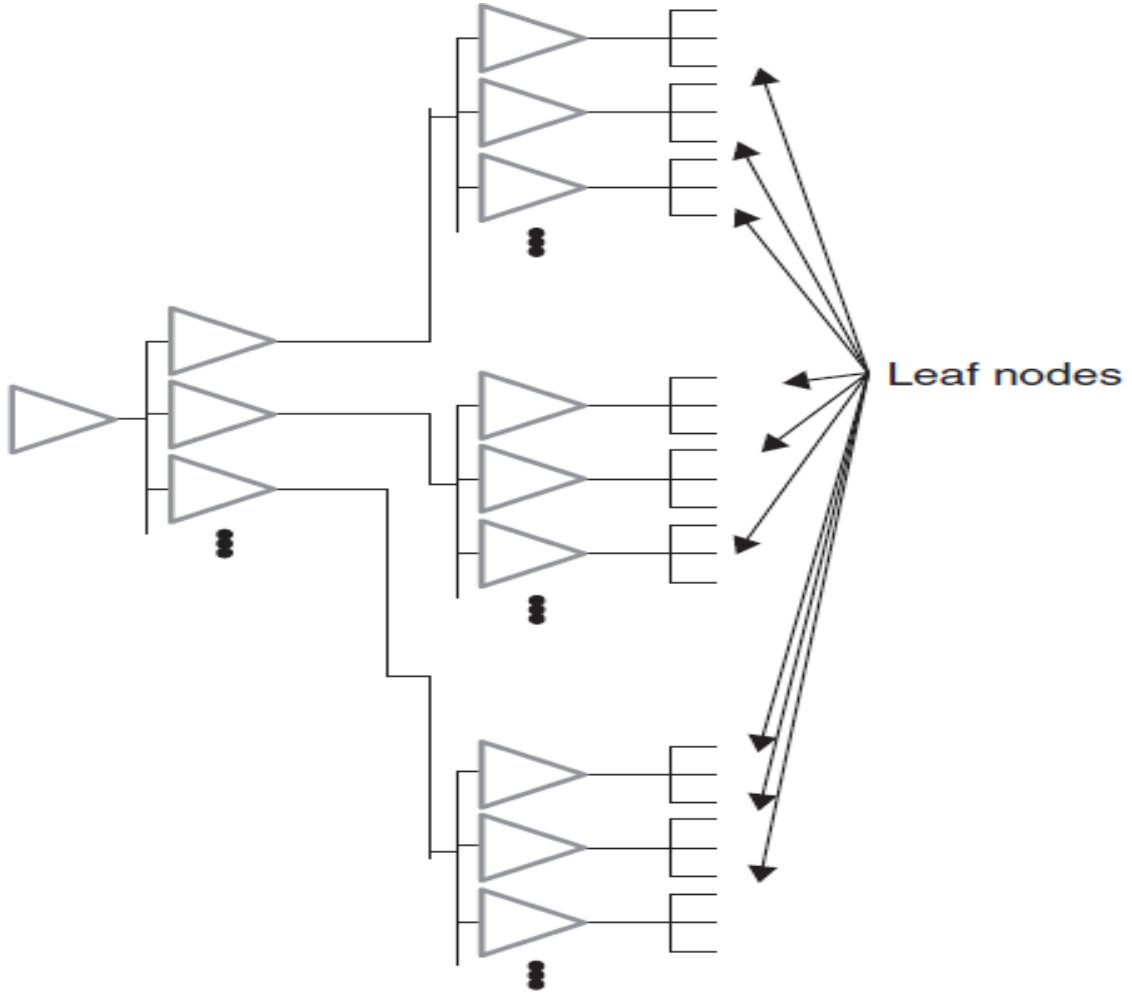
- The clock functions described here fall into two categories. First, there are the buffering cells. These are used to increase the drive strength of the clocking (or control) signals that are distributed across the various chip designs. Second, there are the clock-gating cells. These are used to cleanly enable and disable clock signals.
- There is no set number of levels for such a clock tree. Sometimes, just one level, that of a large buffer (or inverter dependent on the flavor of the clock signal), suffices; sometimes the number could be four or five or more levels, although even for the largest designs, this number tends to be no more than six. In addition, the figure shows a regular fan out of more than three.



A Basic Clock Tree. Also a useful design technique for any high fan-out signal

Standard Cell

- There is no fixed form with this either. Typically, fan out is at the least two, causing a regular bifurcation of the clock signal network, but it could be larger, requiring fewer levels in order to generate a large number of leaf nodes.
- It is possible that there will even be different fan outs at each node in the clock tree. Beyond all of that, there are several variations of this. If the number of flip-flops or storage elements that need to be driven is small, then it could be as simple a clock tree as a signal buffer.
- If the requirements for synchronicity are tight enough in a particular technology, then it could require a messing together of various stages after some or all of the buffering stages in the tree.



A Basic Clock Tree. Also a useful design technique for any high fan-out signal

Standard Cell: Clock functions

- Each stage tends to drive larger stages, cumulatively (and sometimes absolutely) as the clock tree fans out. At the leaf nodes of the tree, the individual clock signal leaves drives out to a subset of the total storage element loading the clock signal.
- The brunt force comes into play at these nodes and at each of the internal levels of the clock tree. Each succeeding level drives large capacitive loads. At the leaf nodes, the edges of the signal have to be sharp. Hence, the solution: make the buffers (or inverters) used in the clock tree large.
- We will find that most stdcell libraries have a set of CLKBUF buffers and CLKINV inverters that are “reserved” for the clock tree (or other signal tree, typical of control signals such as RESET and TEST_MODE). The real reason they are reserved is that they tend to be the largest drive strengths of any buffer or inverter in the stdcell library. Hence, they take up larger footprints on silicon and consume more power during operation.

Standard Cell: Clock functions

- There any other characteristic of these large buffers and inverters that they have in common? Yes. Most stdcells tend to have P-channel to N-channel ratios that are geared to “fastest possible performance” (or “lowest possible power consumption,” or “smallest possible footprint”).
- None of those three “best possible” bear directly to another basic requirement of clock trees: the need for “equal propagation.”
- Typically, along with these cells being larger, the typical P-channel to N-channel ratio will be adjusted in order to allow for equal rising-edge and falling-edge propagations and equal rising and falling edge rates.

Standard Cell: Clock functions

- Realizing this and knowing that it might make sense to allow for NAND, NOR, and MUX usually leads to the addition of several additional functions in the “clock buffer and inverter” list.
- Specifically, a typical stdcell library has several drive strengths of “equal rise and fall and propagation,” CLKNAND, CLKNOR, and CLKMUX2. These usually do not get the largest strengths that the CLKBUF and CLKINV do because they are intended to be used as early in the clock tree as possible (which makes sense because doing so later in the clock tree would require repeating the NAND or NOR function on the signal appropriately).

Standard Cell: Low Power Support

- By the way, if the stdcell library, along with the fabrication house that supports it, does have multi-Vt capabilities, then it is important to be able to place different Vt cells next to each other in an automatically place and routed design.
- This is because design centers, wanting to minimize power, will most likely synthesis with the highest Vt library (lowest power, least performance) available and then to accomplish timing closer, after automatic place and route, by substituting the cells in the failing long-paths with lower Vt cells (assuming that the same cell is available in each Vt).
- Figure 2.18 shows before and after views of automatically placed and routed stdcells having long-path cells substituted for by lower Vt stdcells.

Standard Cell: Low Power Support

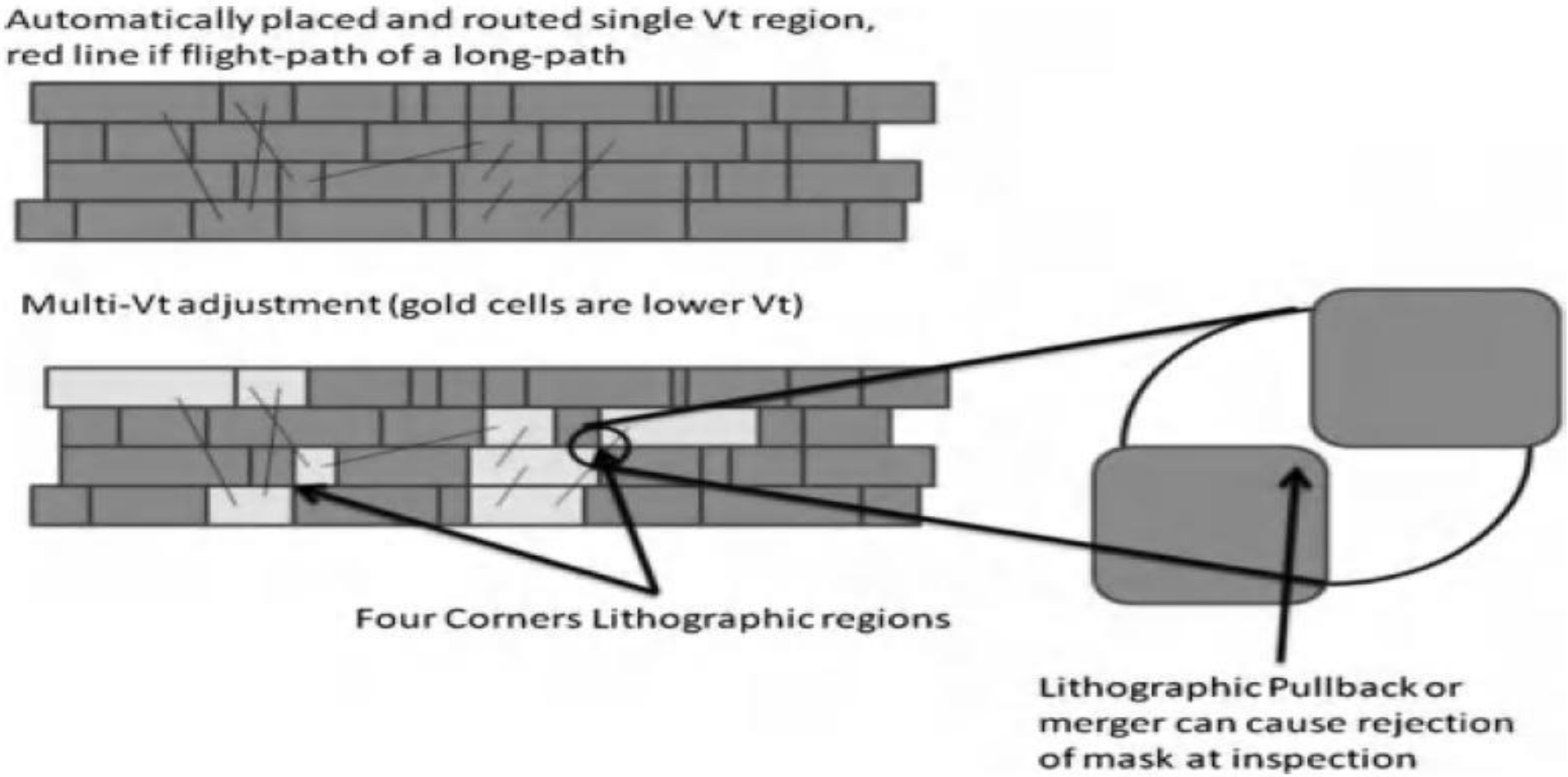


FIGURE 2.18 "Per Cell" Automatic Placement of Different Vt Cells Leading to a "Four Corners" Lithographic Issue. In actuality, cells can be designed such that this is a nonissue (by ensuring that no transistor source or drain falls within the region where such region lithographic pullback can occur).

Standard Cell: Low Power Support

- The lithographic issue is that the masking plate for the two large regions of antireflective chrome comes together at a point.
- In reality, these two regions either will have to be separated by a small gap (the regions having been pulled back from each other) or will have to be merged.
- During incoming mask inspection into the fabrication house (or outgoing inspection out of the mask house), these gaps or mergers can cause some to reject the mask. This is truly a lithographic issue and not an electrical issue.

Standard Cell: Boutique Library Offerings

- An example of a boutique library would be one for a technology shrink of an extant ASIC design (or design block) that is a known good (read functionally correct). The design center may not wish to resynthesis from a RTL, which would require some amount of formal validation of the resultant netlist, or the original RTL may have been lost, or too significant an amount of post-synthesis manual manipulation of the gate-level netlist may have taken place.
- Perhaps another company that had been using a different stdcell library supplier has been bought and the task is to transfer (known as port) a netlist into the correct design environment for use by other internal design centers in the same technology node.

Standard Cell: Topics of Interest

- **Multistage, two-stack versions of the stdcells :** As technology nodes continue to shrink, the need for reducing the number of transistors stacked between rails and outputs will increase. It might be useful for the student or interested engineer to attempt to replicate as many of the given functions in two-stack logic with increased staging depth.

- **Transmission-gate–based MUX versions of the various cells:** All of the MUX logic was implicitly designed with a Boolean-based MUX function in mind. However, there are times when a true transmission-gate–based (or tri-statable inverter) MUX would be beneficial (with significantly faster performance). The interested engineer might attempt to design such MUX (and explore pin-order benefits in such designs).

Standard Cell: Topics of Interest

- **Boolean-based FLIP-FLOP versions of the various cells:** Almost all FLIP-FLOP circuits designed over several generations of technology nodes have used some amount of transmission-gate–based MUX circuitry inside the actual stdcell (usually, but not always, three tri-state inverters and one true transmission gate).
- However, as technology nodes head toward 20 nanometer, efficiently handling the significant amount of signal crossing within the cell that is required by such a circuit topology can become cumbersome, One alternative is to use Boolean constructs within the stdcell as opposed to transmission-gate constructs in order to MUX the appropriate signals together.
- The interested engineer might find it interesting to attempt to design a Boolean-based FLIP-FLOP for each FLIP-FLOP listed in the chapter. One caveat: Boolean-based FLIP-FLOPS tend to be slower in both output propagation and synchronous input constraint.

Standard Cell: Topics of Interest

- **Pulsed versions of the various cells:** Several design centers have attempted to use pulsed logic where a logic function does not resolve until an active pulse is recorded on a transistor (or more than one transistor) within the cell. Such designs have been implemented and used, with varying degrees of success, over multiple technology nodes.
- **Majority vote usage:** As mentioned in the chapter, the two-of-three majority vote MAJ3 and the three-of-five majority vote MAJ5 stdcells are usually not included within a stdcell library offering. The interested engineer might be inclined to develop some RTL, perhaps for a communication channel error detection and correct circuit and see how the synthesized netlist work change as a function of the inclusion versus exclusion of these two stdcells.

Standard Cell: Topics of Interest

- **Minimal versus maximal stdcell library synthesis:** Synthesis of a four-bit full adder with a minimum number of stdcells versus a richer offering of the same.
- The interested engineer might be inclined to repeat this exercise using various other RTL in order to understand why a rich Boolean offering is important to allow smaller and faster synthesized netlist.

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu

Semi-Custom Layout

Dr. Sudeendra kumar K

Department of Electronics and Communication Engineering

ADVANCED DIGITAL DESIGN

Semi-Custom Layout

Sudeendra kumar K
Department of Electronics and Communication Engineering

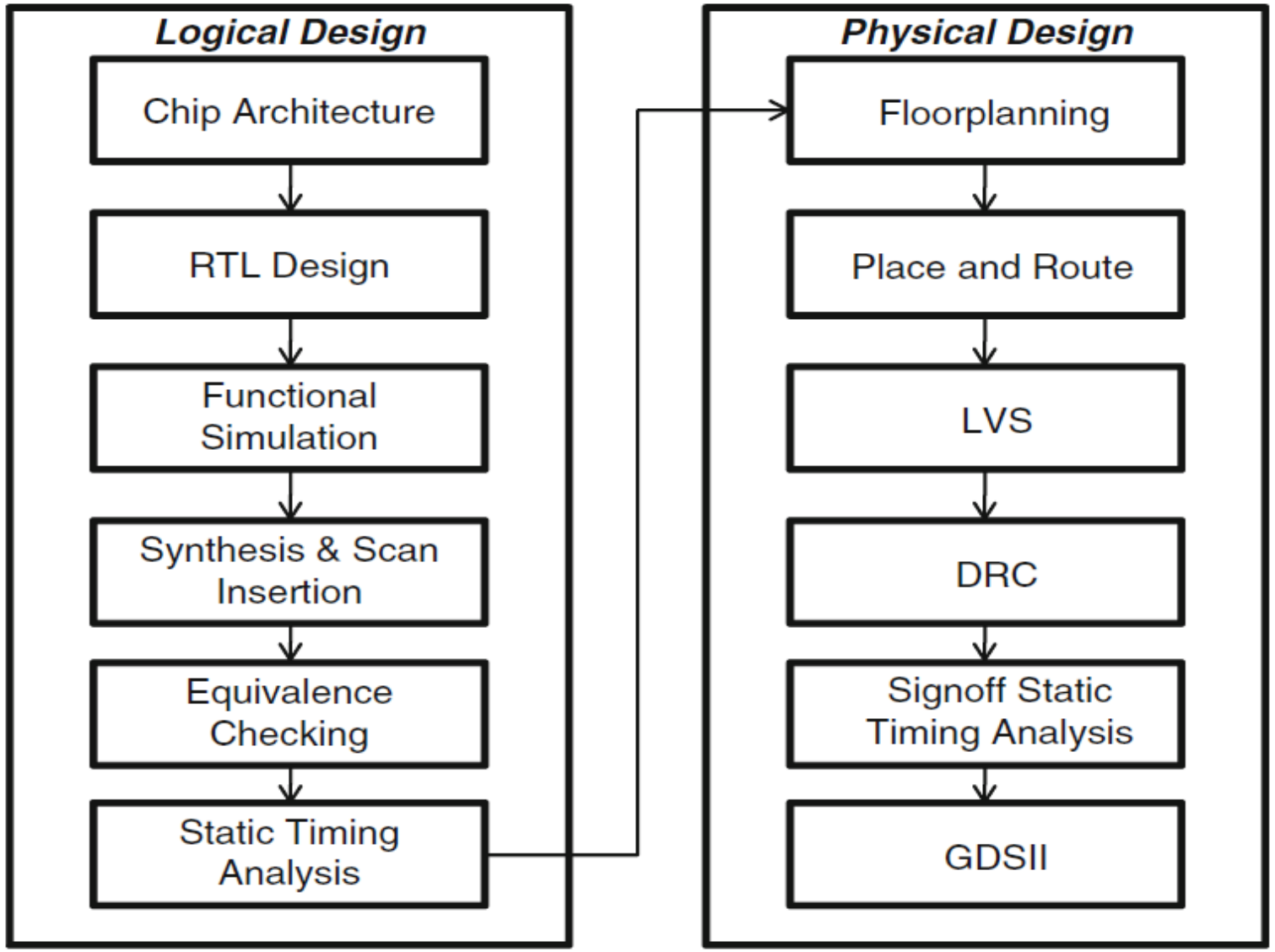
Advanced Digital Design

Contents

- Layout
- Full Custom Layout
- Semi-Custom Layout
  - Placement
  - Routing

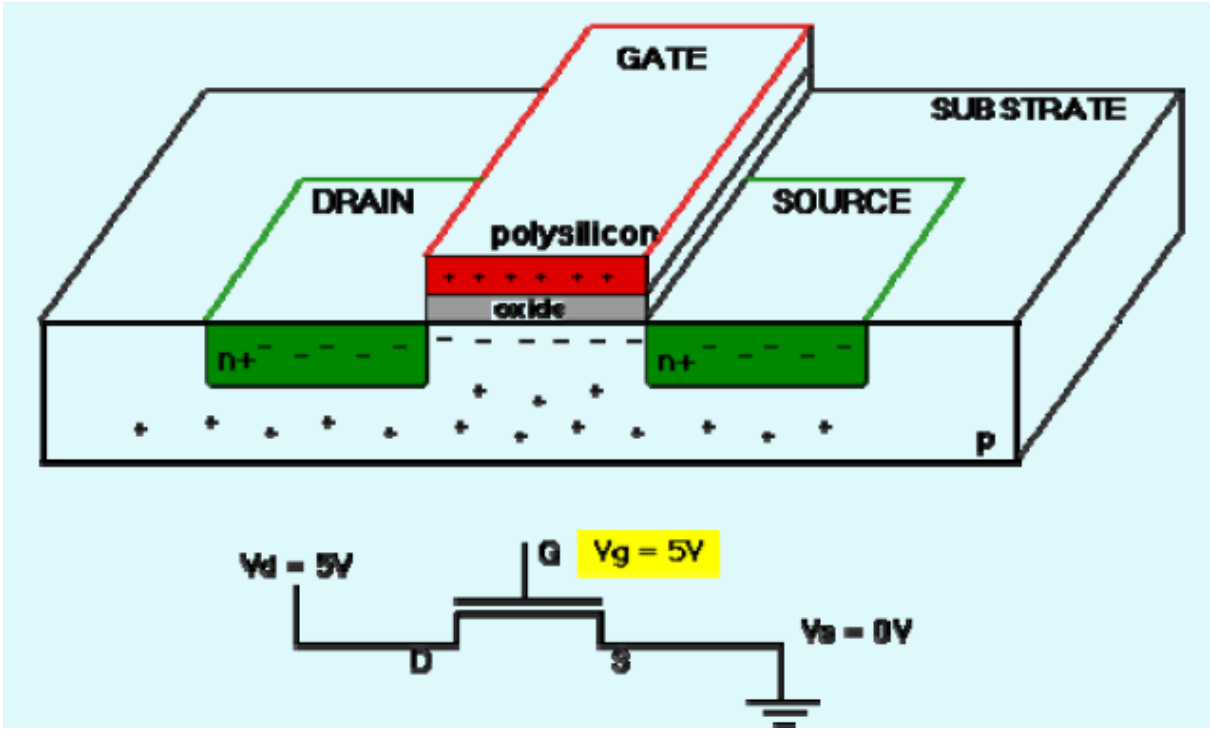
Advanced Digital Design

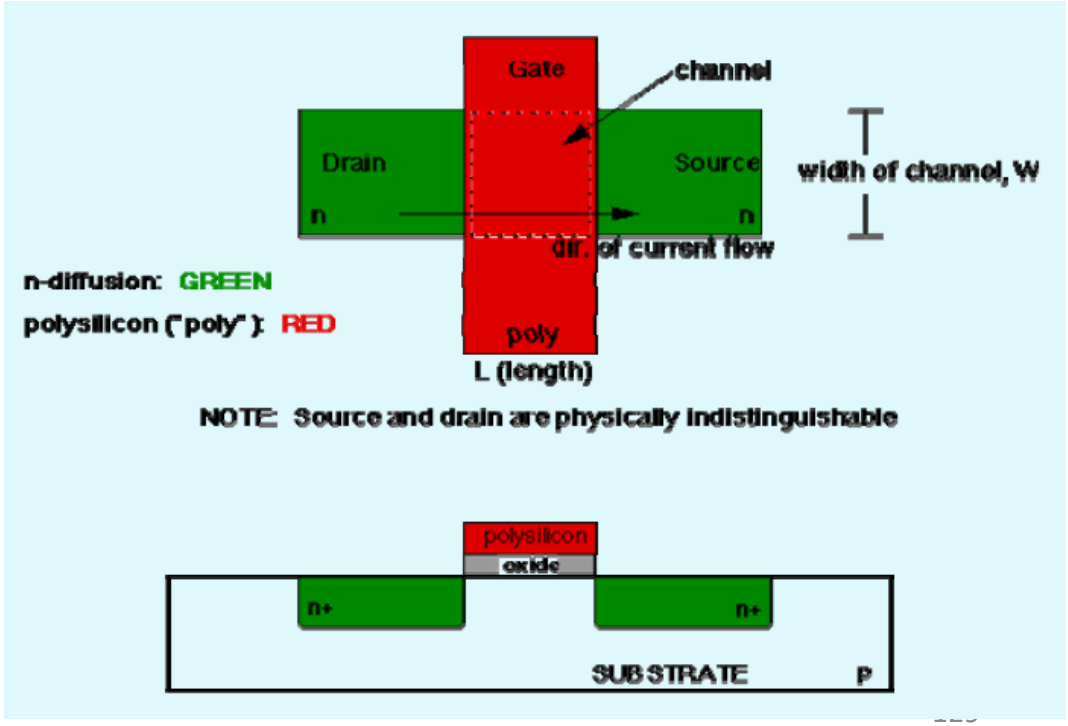
ASIC Design flow



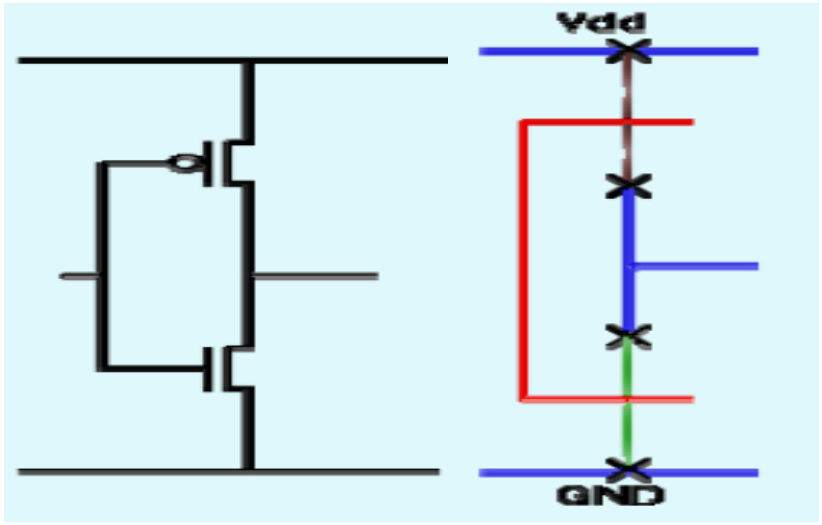
Layout

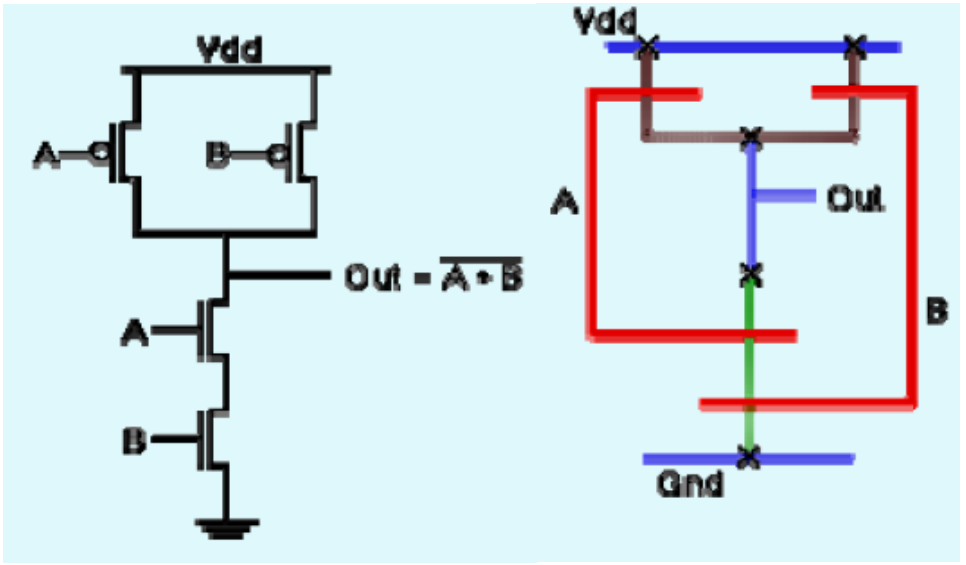
- In integrated circuit design, physical design is a step in the standard design cycle which follows after the circuit design.
- At this step, circuit representations of the components (devices and interconnects) of the design are converted into geometric representations of shapes which, when manufactured in the corresponding layers of materials, will ensure the required functioning of the components.
- This geometric representation is called integrated circuit layout.
- Typically, the IC physical design is categorized into full custom and semi-custom design.
- **Full-Custom:** Designer has full flexibility on the layout design, no predefined cells are used.
- **Semi-Custom:** Pre-designed library cells are used, designer has flexibility in placement of the cells and routing.

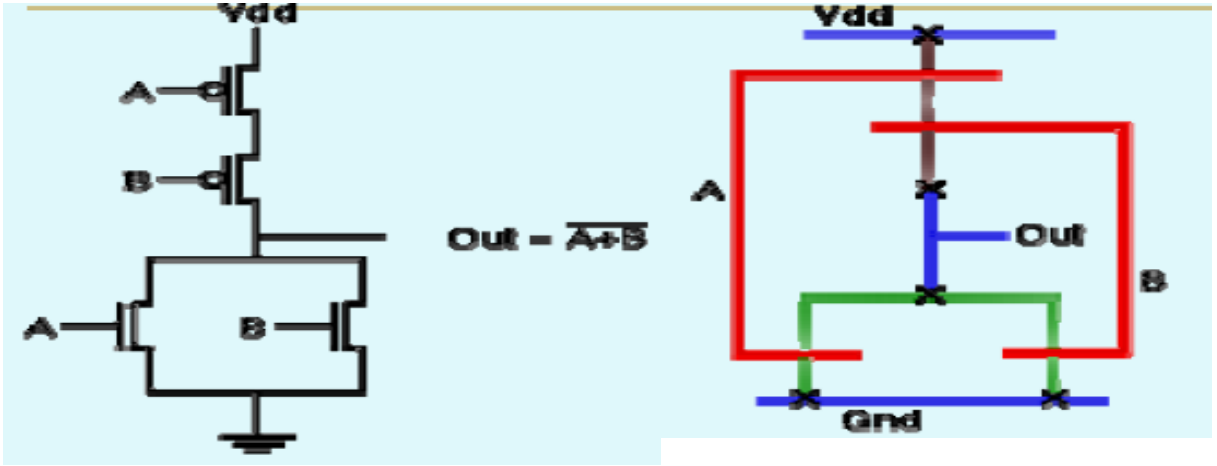

GATE
SUBSTRATE
DRAIN
polysilicon
SOURCE
oxide
n+
n+
p
Vd = 5V
G
Vg = 5V
Vs = 0V
D
S


Gate
channel
Drain
Source
n
n
width of channel, W
dir. of current flow
n-diffusion: GREEN
polysilicon ("poly"): RED
poly
L (length)
NOTE: Source and drain are physically indistinguishable
polysilicon
oxide
n+
n+
SUBSTRATE
p

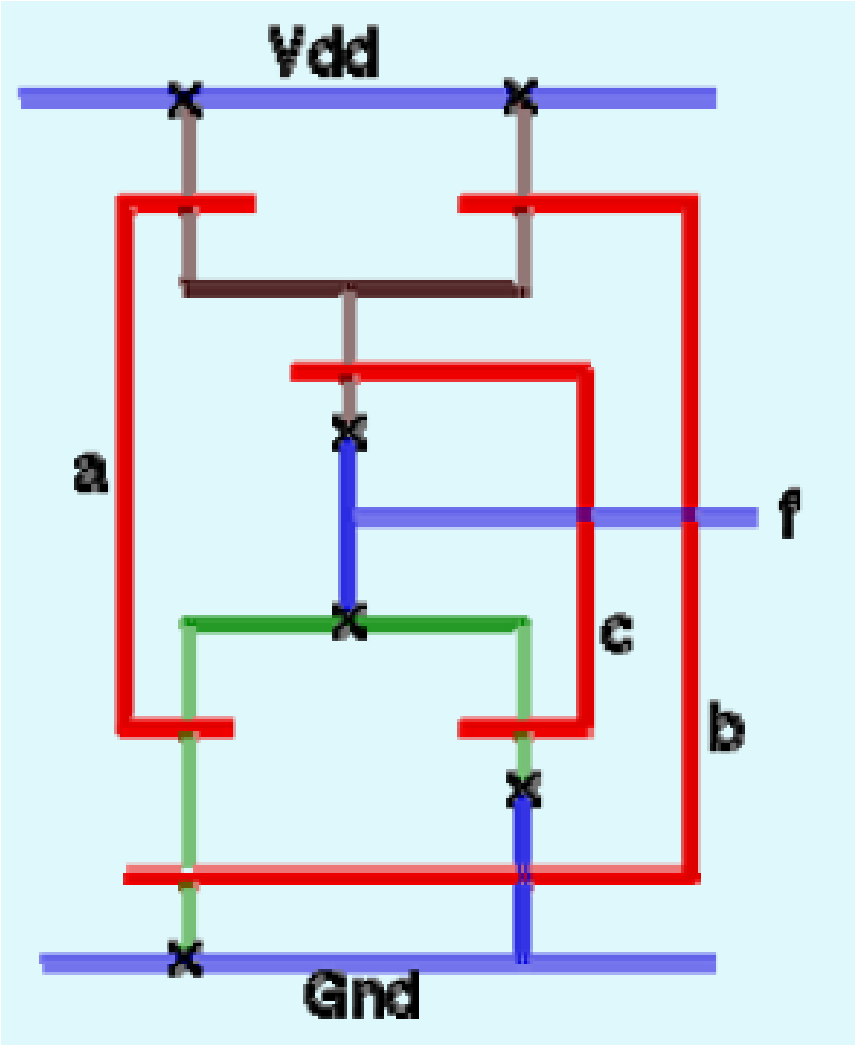
Layout of Circuits

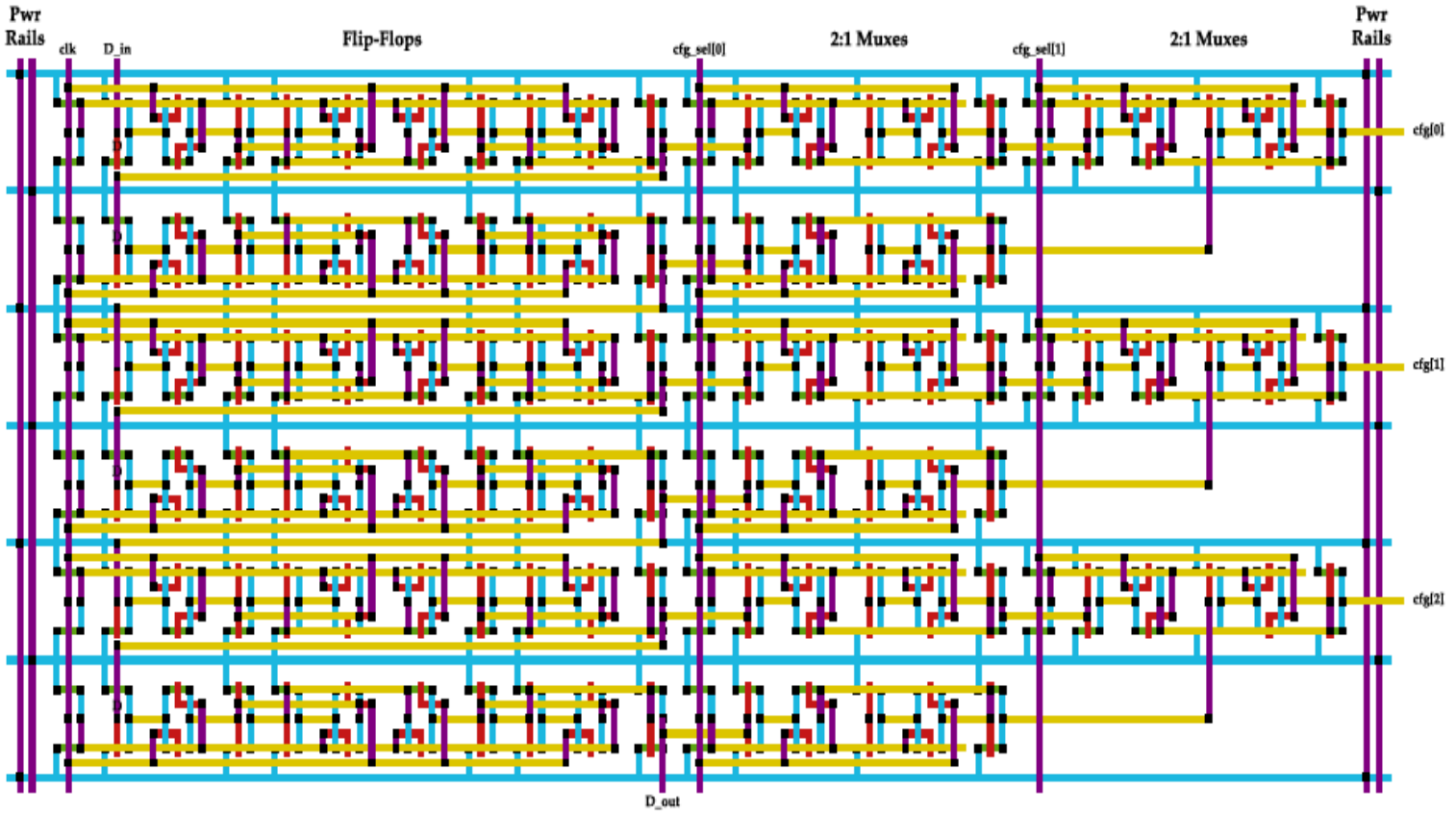






Advanced Digital Design
Layout of the circuits

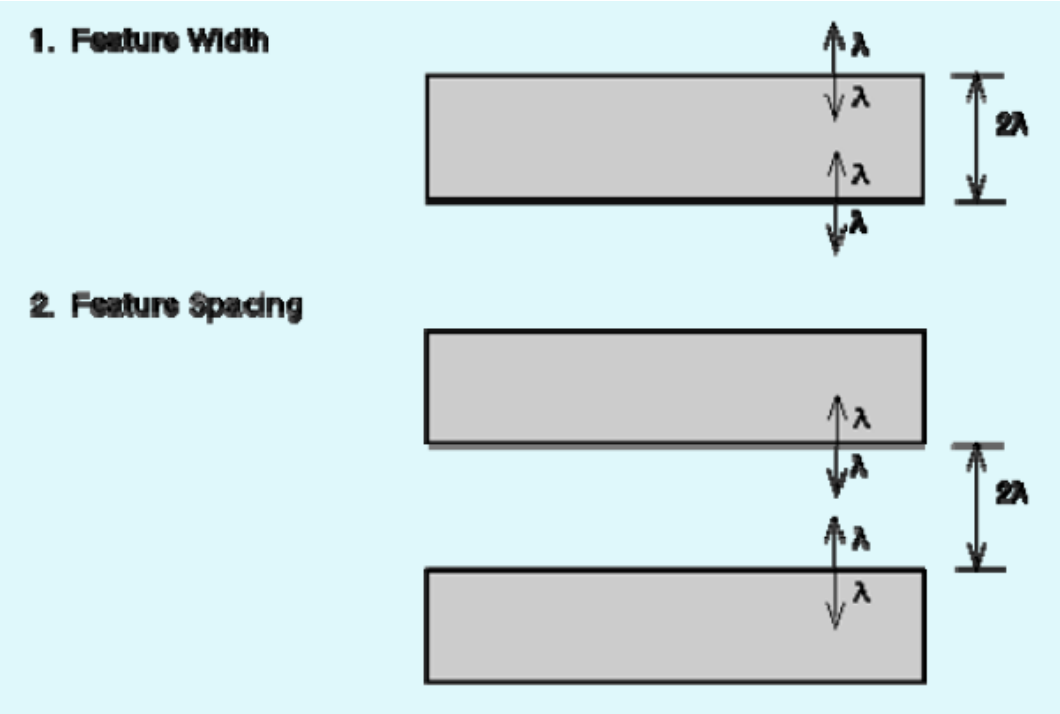


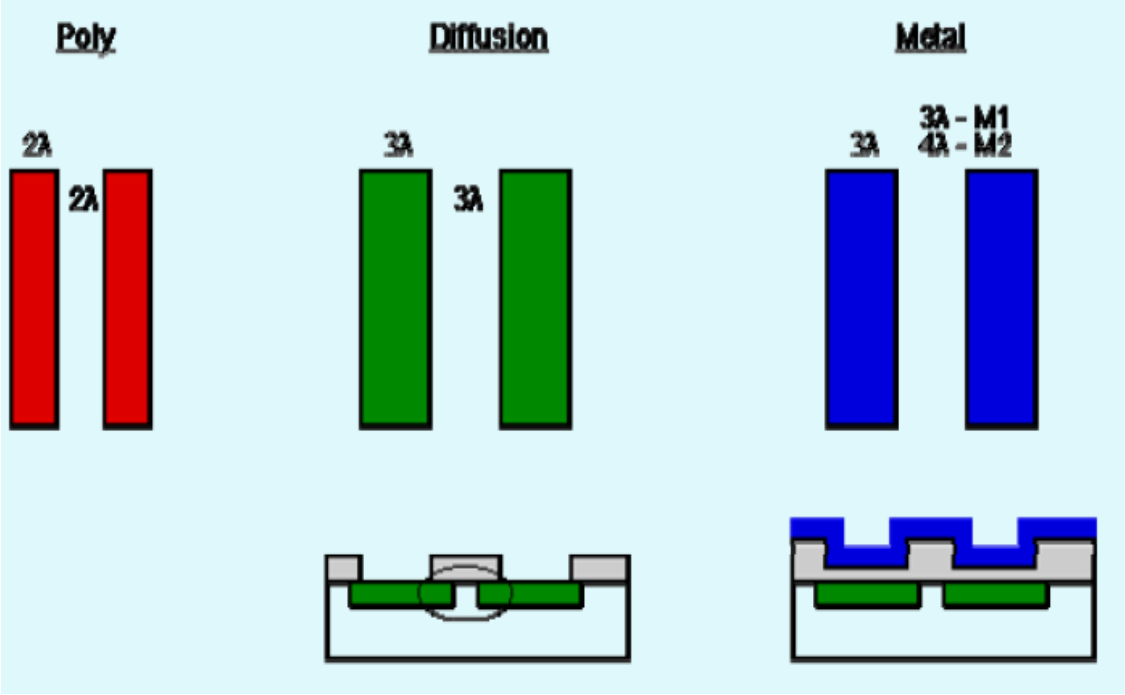


Example: $f = \overline{a \cdot b + c}$

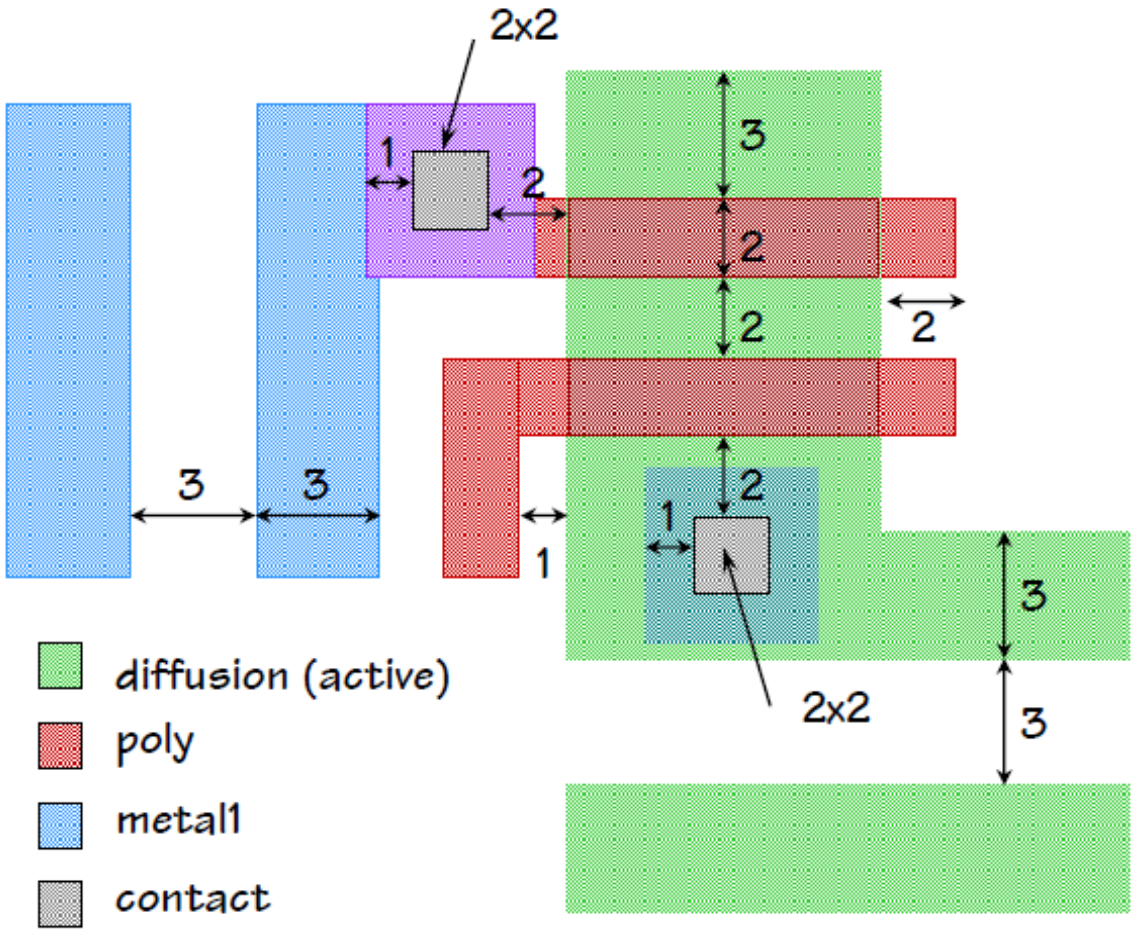
Layout of the circuits

- Lambda-based design rules are based on the assumption that one can scale a design to the appropriate size before manufacture.
- The assumption is that all manufacturing dimensions scale equally, an assumption that "works" only over some modest span of time.



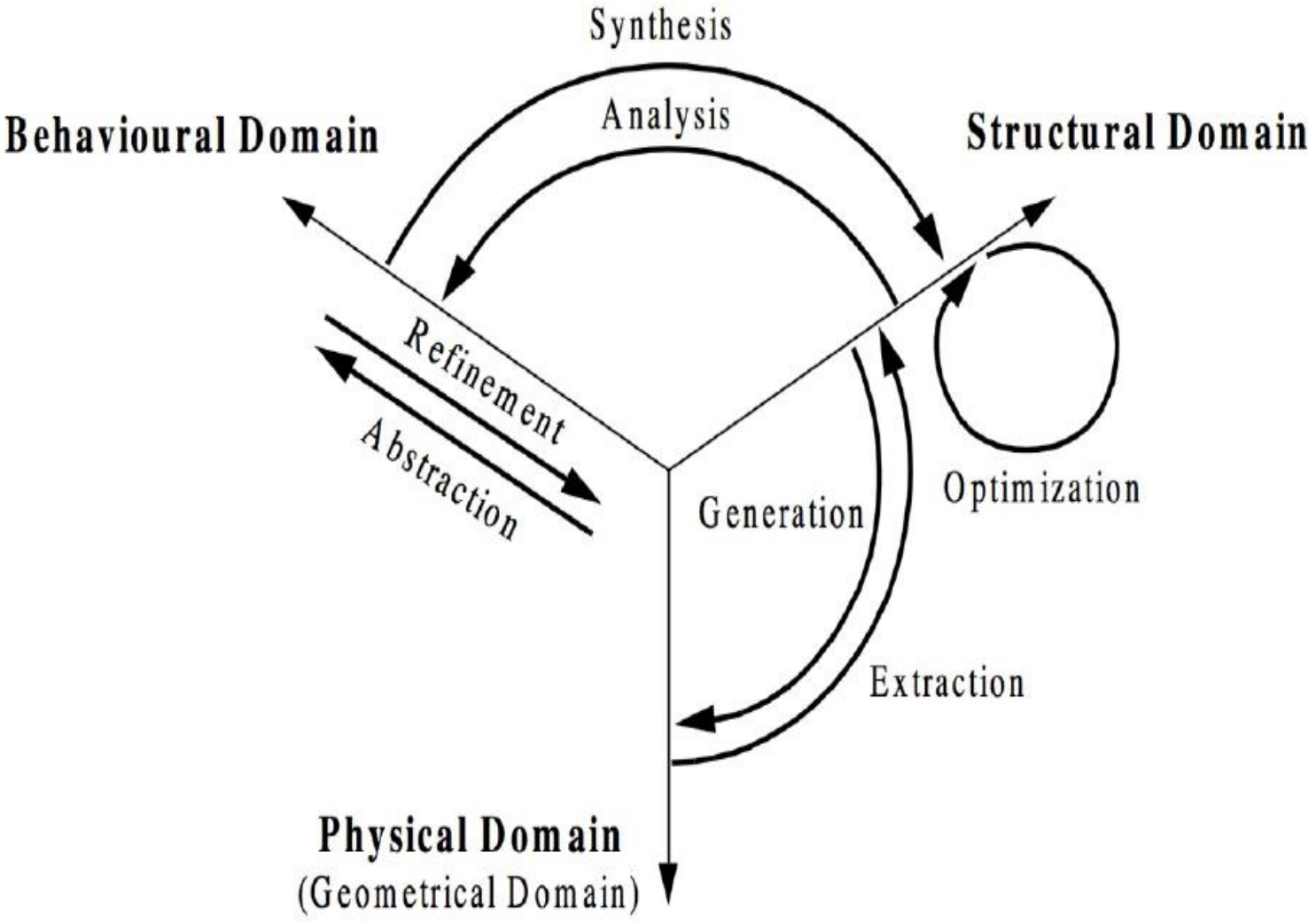


One lambda = one half of the "minimum" mask dimension, typically the length of a transistor channel. Usually all edges must be "on grid", e.g., in the MOSIS scalable rules, all edges must be on a lambda grid.



lambda rule	lambda = 0.5u
1λ	0.5μ
1λ	0.5μ
3λ	1.5μ

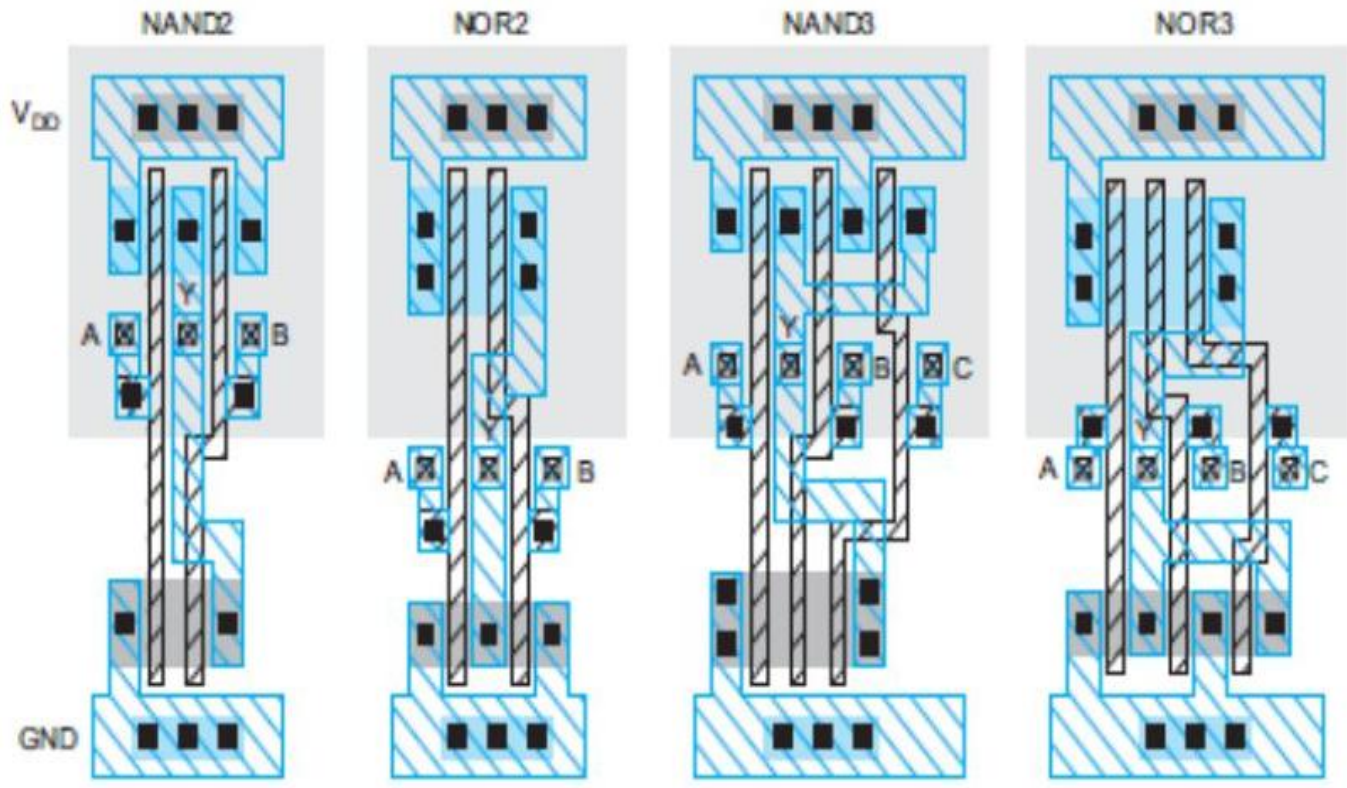
Different abstractions (domains) of Physical Design



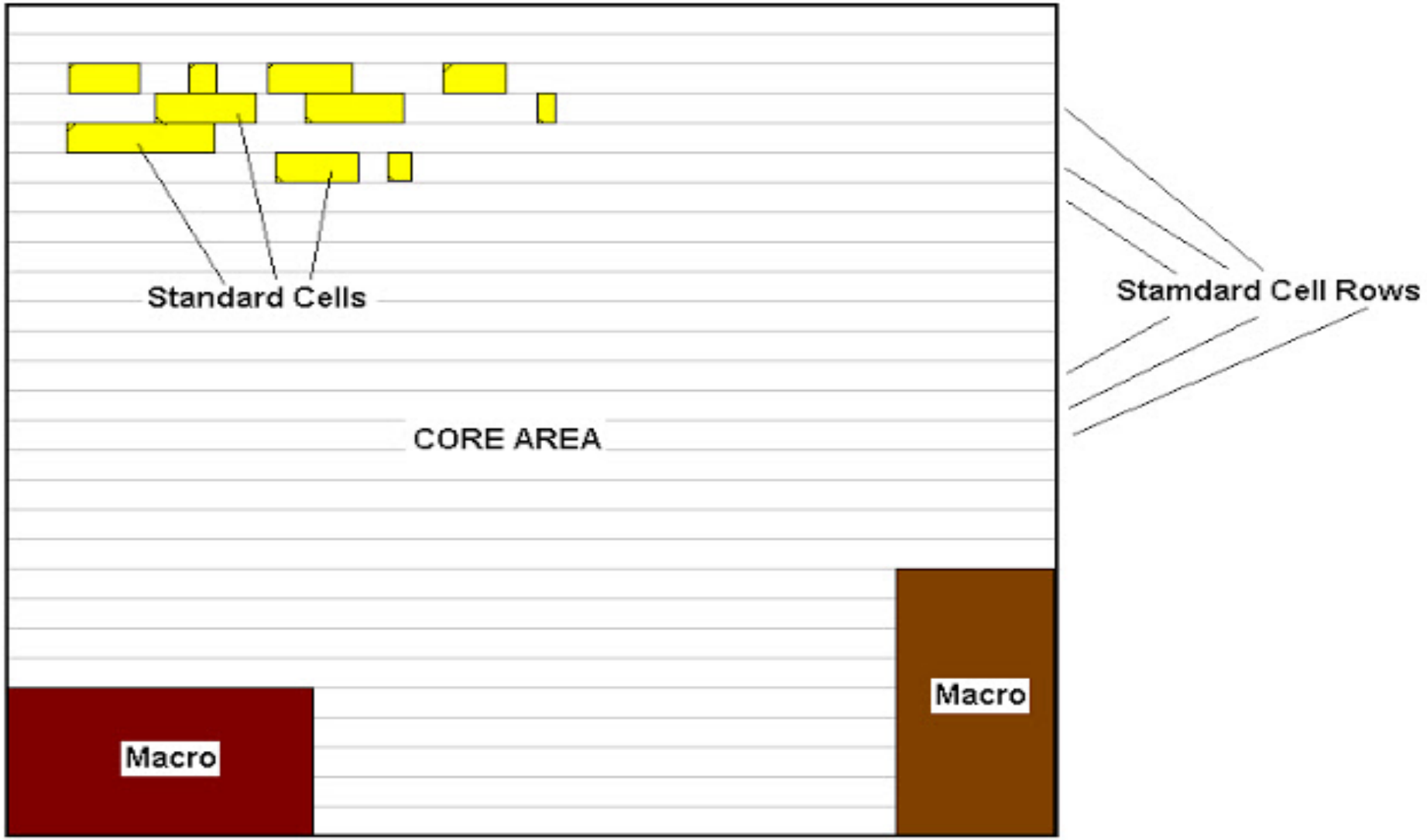
Standard Cells

Basic idea:

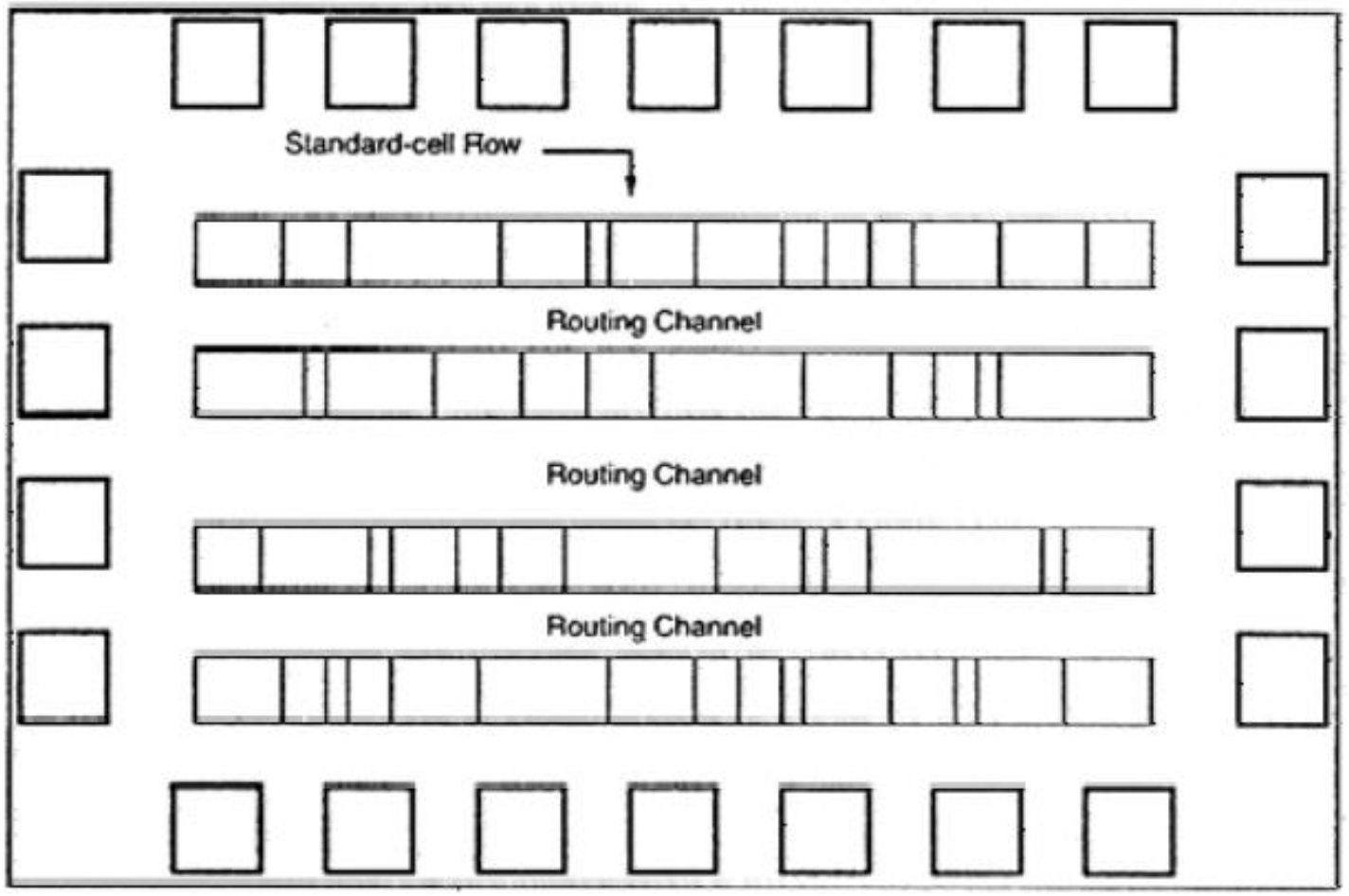
- All of the commonly used logic cells are developed, characterized, and stored in a standard cell library.
- A typical library may contain a few hundred cells.
- Inverters, NAND gates, NOR gates, complex AOI, OAI gates, D-latches, and flip-flops.
- **Each cell is designed with a fixed height.**
- To enable automated placement of the cells, and Routing of inter-cell connections.
- A number of cells can be abutted side-by-side to form rows.
- The power and ground rails typically run parallel to upper and lower boundaries of cell.



Standard Cell Placement and automated layout creation

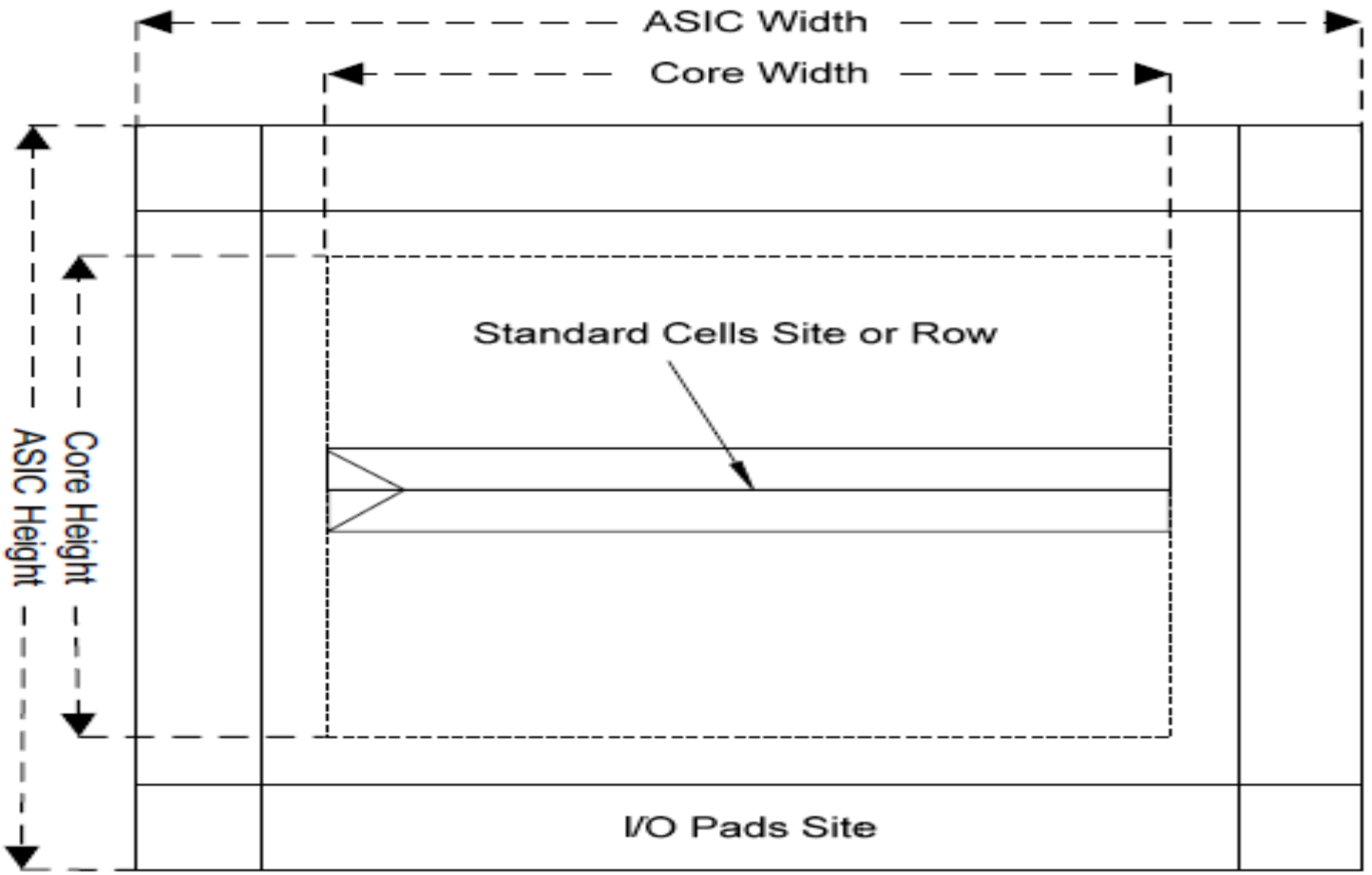


Standard Cell Placement and automated layout creation

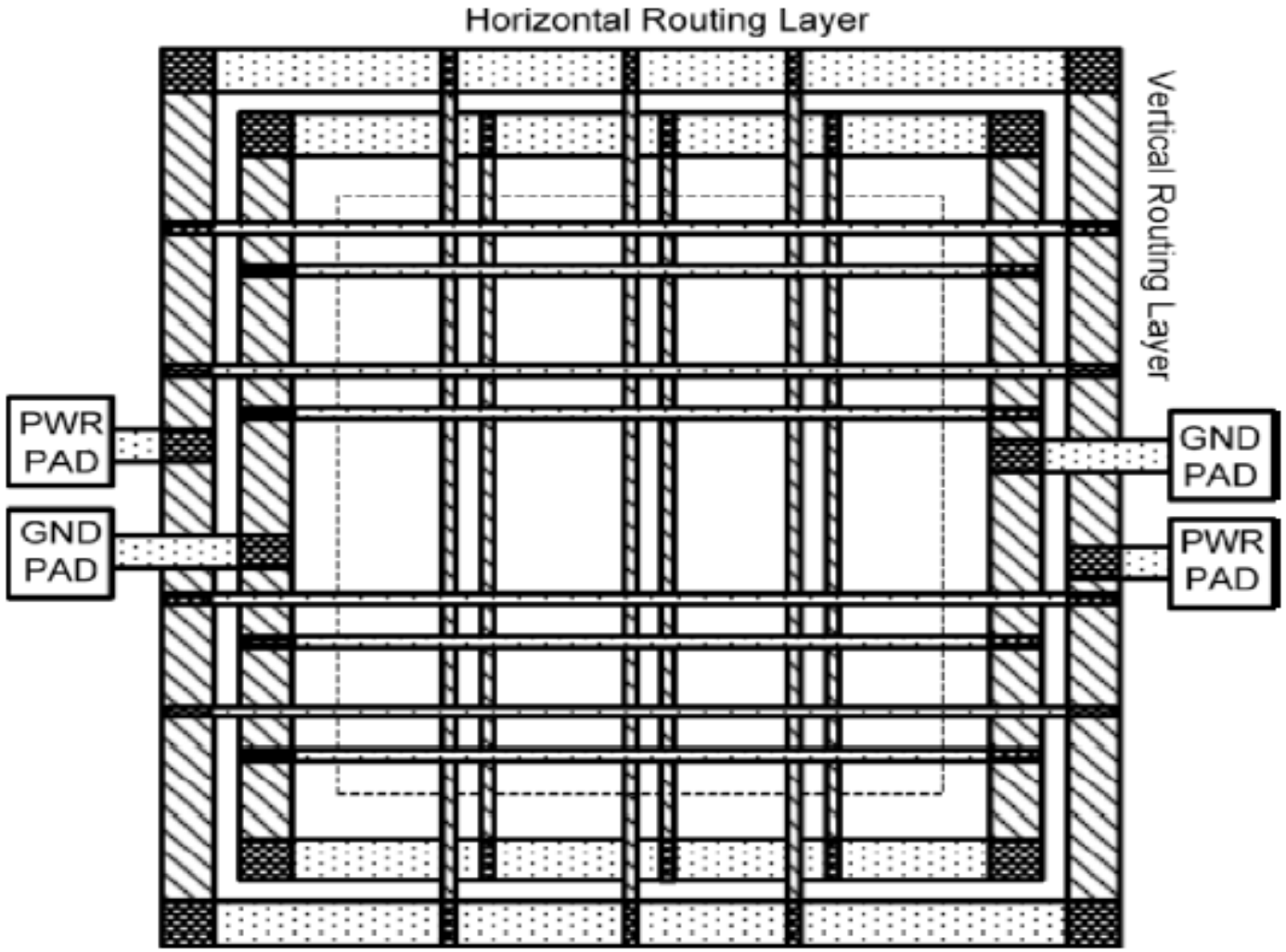


Standard Cell Layout: Floorplanning

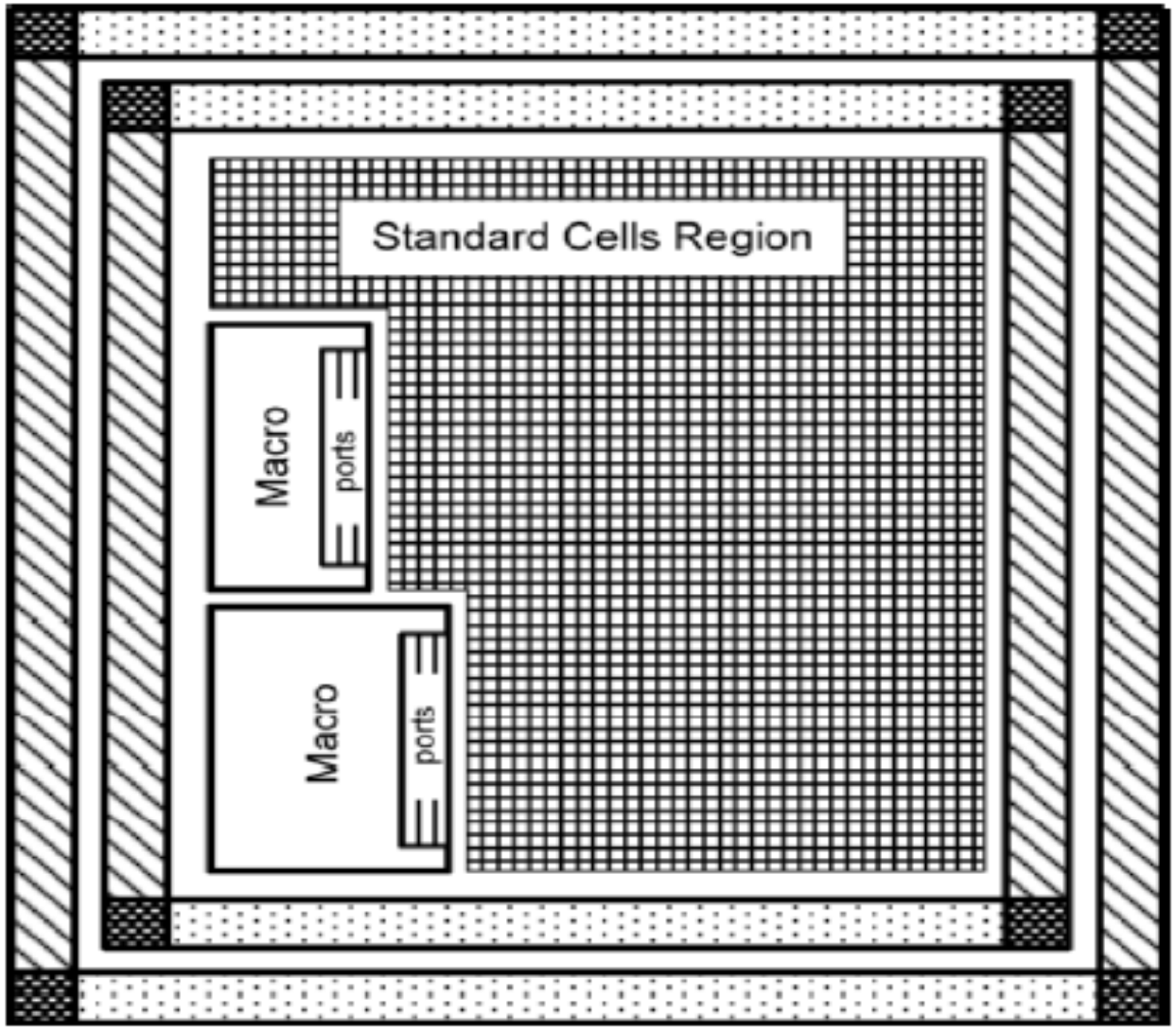
- Floorplanning is the art of any physical design. A well thought-out floorplan leads to an ASIC design with higher performance and optimum area.



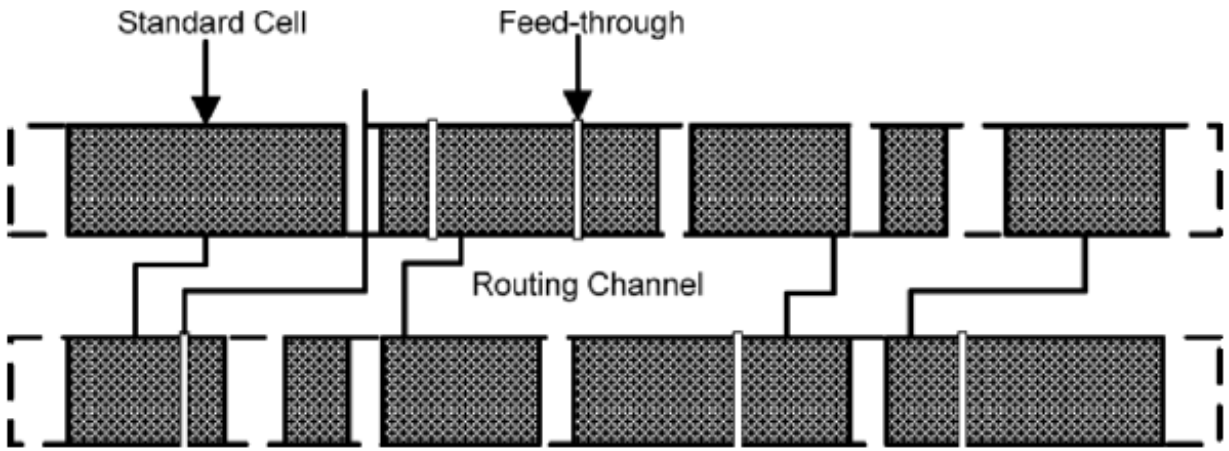
Advanced Digital Design
Semi-Custom Layout: Power Planning



Advanced Digital Design
Semi-custom Layout: Cell View

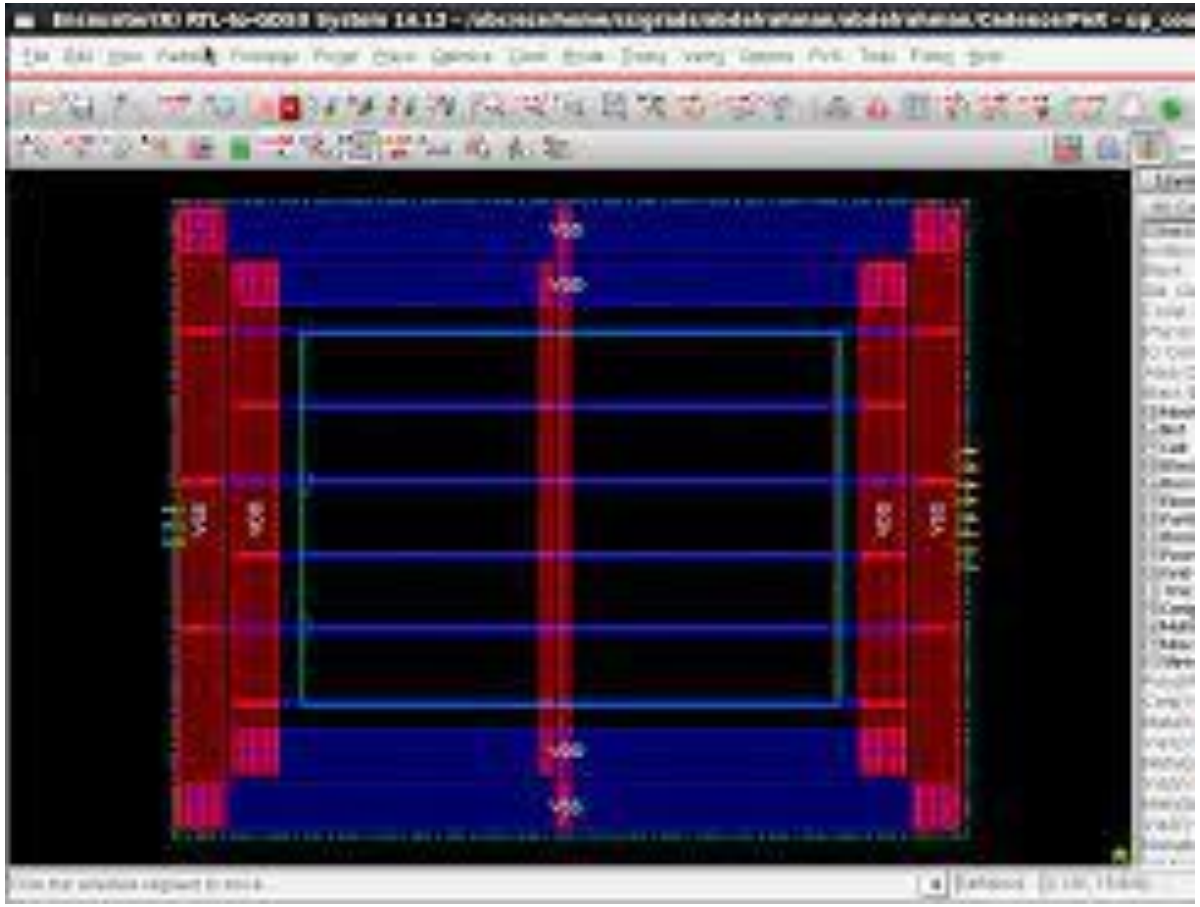


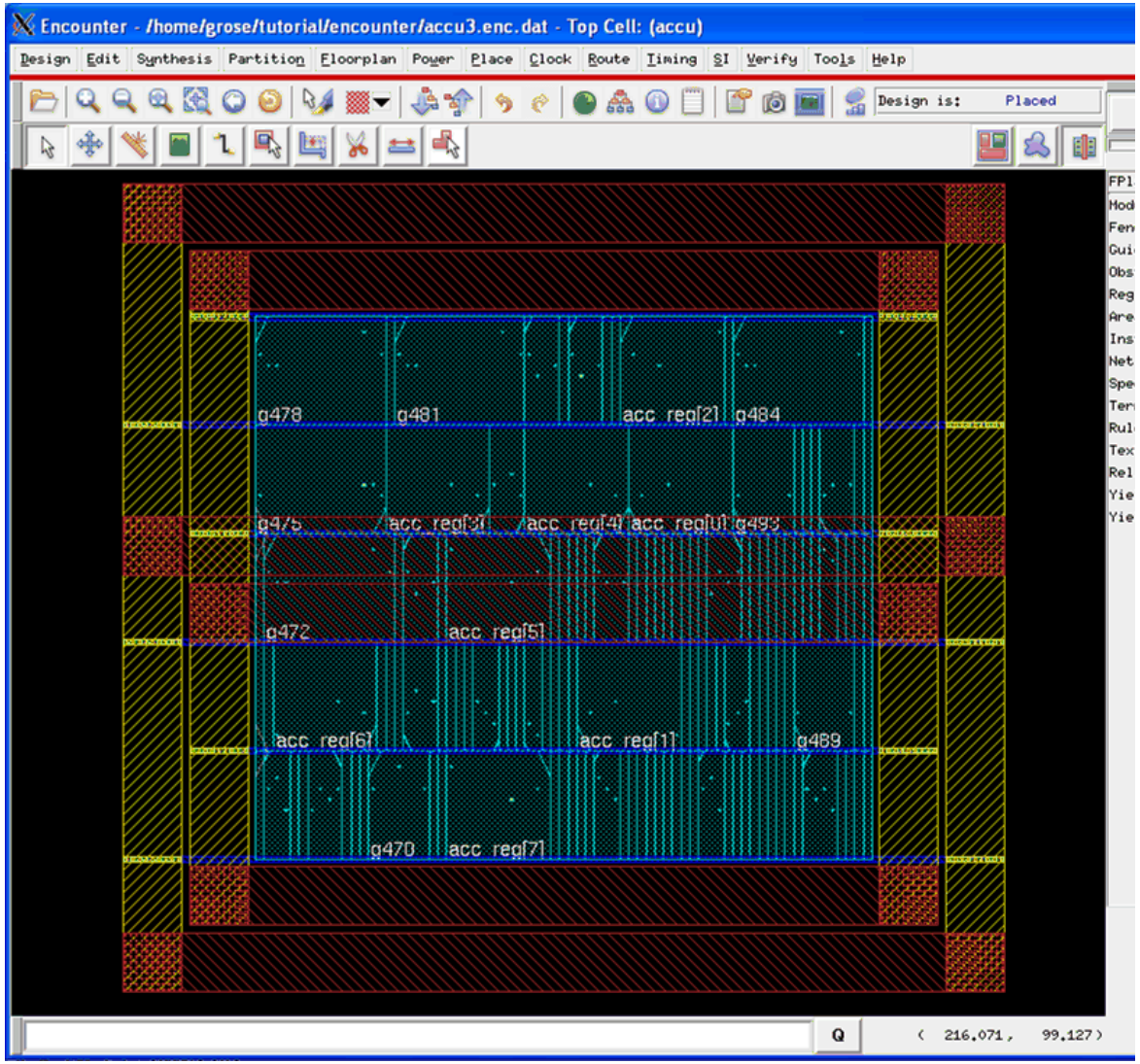
Advanced Digital Design
Semi-custom Layout : Routing



Advanced Digital Design

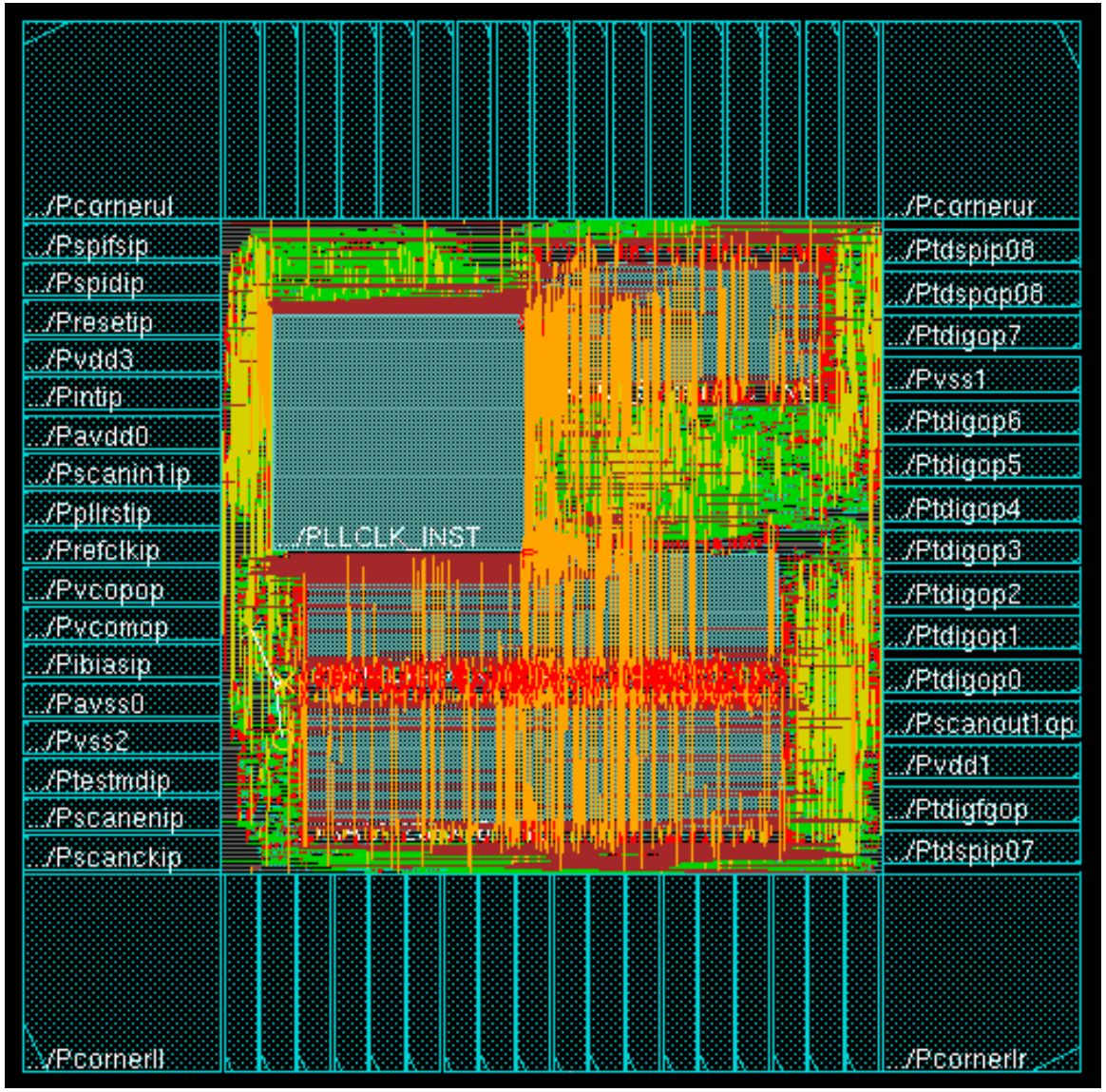
Semi-custom Layout: in Cadence Encounter/Innovus





Advanced Digital Design

Semi-custom Layout: Full chip tape out in Cadence Innovus



Semi-custom Layout: Input and Output files

- Input files: -
  - Verilog Gate level netlist from Synthesis
  - Layout exchange format (LEF) file from library
  - Standard cell library (.lib) (for ex: saed90nm.lib)
  - Timing Constraints file (.sdc) (Standard Design Constraints)
  - Configuration file (in some cases optional)
- Output files: -
  - Layout of the circuit in different formats like DEF (Design exchange format) etc
  - GDS-II (Binary format of Layout) : which will be sent to Foundry/fabrication facility to manufacture the chip.
  - Parasitics extraction files (SPEF) (Standard Parasitics Extraction Format)

Advanced Digital Design

Summary and References

- Semi-Custom Layout: A Brief Introduction

References

- This Presentation
- Online Class Lecture Explanation
- For more details : Physical Design Essentials, Springer Publications.

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu

Digital System Design

Dr. Sudeendra kumar K

Department of Electronics and Communication Engineering

DIGITAL SYSTEM DESIGN

Floorplanning

Sudeendra kumar K
Department of Electronics and Communication Engineering

Introduction to Floorplanning

- Floorplanning is the art of any physical design. A well thought-out floorplan leads to an ASIC design with higher performance and optimum area.
- Floorplanning can be challenging in that it deals with the placement of I/O pads and macros as well as power and ground structures.
- Before one proceeds with physical floorplanning one needs to make sure that the data used during the course of physical design activity is prepared properly.
- Proper data preparation is essential to all ASIC physical designs in order to implement a correct-by-construction design.

Introduction to Floorplanning

- The types of data that are required to start a physical design are: -
- Related technology and library files
- Circuit description of the design in the form of netlist representation
- Timing requirements or design constraints
- Floorplan

Floorplanning: Technology File

- Almost all physical synthesis and place-and-route tools operate based on the technology file.
- Technology files contain information or commands that are used to configure structures, parameters (such as physical design rules and parasitic extractions), and limits of an ASIC design targeted to specific process technology.
- These commands are used at different stages of ASIC design implementation by physical design tools. One of the objectives is to make sure that all such as standard cell placement, routing quality, and accuracy of parasitic extraction, should be carefully analyzed. Based upon the final inspection, technology files may require further refinement for optimal performance.

Floorplanning: Technology File

- Technology rule basics are as follows:
  - Manufacturing grid
  - Routing grid
  - Standard cell placement tile
  - Routing layer definition
  - Placement and routing blockage layer definition
  - Via definition
  - Conducting layer density rule
  - Metal layer slotting rule
  - Routing layer physical profile
  - Antenna definition

Floorplanning: Technology File definitions

- **Manufacturing grid** is determined by the smallest geometry that a semiconductor foundry can process. All drawn geometries during physical design must snap to this manufacturing grid.
- **Routing grids or tracks** are used by physical synthesis and place-and-route tools during detail routing. The routing tracks can be grid-based, grid-less based, or sub-grid-based.
- **Standard cell placement tile** is used during the placement phase. The placement tile is defined by one vertical routing track and the standard cell height.
- **Routing layer definition** is used to define the layers that are used to route the design. These definitions include wire width, routing pitch, and preferred routing direction such as vertical, horizontal, or diagonal.

Floorplanning: Technology File definitions

- **Placement and routing** blockage layer definitions are internal to physical design tools and are used to define “keep-out” regions for standard cell placement and routing.
- **Via definition** defines the layer, size, and type for connection between overlapping geometries of conductor for different conductive layers. This cut layer, or via, can be a single via, stacked via, or array of via.
- **Conducting layer density** rule defines the percentage of area of the chip that is required for processes that are using Chemical Mechanical Polishing (CMP) for each physical layer in the design.
- **The chemical mechanical polishing process** requires limited variation in feature density on conducting layers. This dictates that the density of layout geometries in a given region must be within a certain range. With new silicon processes (i.e. metallization), violation of this rule can have negative impacts on the yield.

Floorplanning: Technology File definitions

- **Configuration of metal layers** slotting rule defines the minimum layer width that may need to have slotting features (i.e. a cut inside a wide routing layer). This rule varies between foundries and is used to limit mechanical stress for a given conducting layer.
- **Physical profile** for each layer is used to define and include conductor thickness, height, and interlayer dielectric thickness. Definition of the electrical interconnect profile includes resistance and dielectric constants.

Floorplanning: Circuit Description

```
module bottom_level ( D, E, Y1, Y2, Y3, Y4);
    input     D[3:0], E;
    output    Y1,Y2,Y3;
    inout     Y4;

    assign          D[2] = Y2;
    OR_type         I0 ( .A(D[0]), .B(D[1]), .Z(Y1) );
    AND_type        I2 ( .A(D[2]), .B(Y4), .Z(Y3) );
    TRIBUF_type     I3 (. A(D[3]), .ENB(E), .Z(Y4) );
endmodule

module top_level (IN0,IN1,IN2,IN3,EN,OUT0,OUT1,BIDIR);
    input     IN0,IN1,IN2,IN3,EN;
    output    OUT0, OUT1;
    inout     BIDIR;
    wire      NET1,NET2,NET3,NET4,NET5,NET6;

    bottom_level  bottom_level_instance ( .D{net4,net2,IN1,
          net1}, .Y1(net5), .Y2(), .Y3(net6), .Y4(BIDIR) );
    BUF_type      I0 ( .A(IN0), .Z(net1) );
    BUF_type      I1 ( .A(IN2), .Z(net2) );
    BUF_type      I2 ( .A(EN), .Z(net3) );
    BUF_type      I3 ( .A(IN3), .Z(net4) );
    BUF_type      I4 ( .A(net5), .Z(OUT0) );
    BUF_type      I5 ( .A(net6), .Z(OUT1) );
endmodule
```

Floorplanning: Design Constraints

- Design constraints are ASIC design specifications that are applied during logic and physical synthesis. Each tool attempts to meet two general design constraints:
- Timing constraints
- Design rule constraints

- Timing constraints are user-specified and are related to speed, area, and the power consumption of the ASIC design. Timing constraints utilized by physical design tools are performance related. The most basic timing constraints are as follows: -
    - System clock definition and clock delays
    - Multiple cycle paths
    - Input and output delays
    - Minimum and maximum path delays
    - Input transition and output load capacitance
    - False paths

Floorplanning: Design Constraints

- Design rule constraints are imposed upon ASIC designs by requirements specified in a given standard cell library or within physical design tools.
- Design rule constraints have precedence over timing constraints because they have to be met in order to realize a functional ASIC design. There are four types of major design rule constraints:
  - Maximum number of fan-outs
  - Maximum transitions
  - Maximum capacitance

Floorplanning: Design Rule Checks

- **Maximum number of fan-outs** specify the number of destinations that one cell can connect to for each standard cell in the library. This constraint can also be applied at the ASIC design level during the physical synthesis to control the number of connections one cell can make.
- **Maximum transition constraint** is the maximum allowable input transitions for each individual cell in the standard cell library. Apart from each element in the standard cell library, this constraint can be applied to a specific net or to an entire ASIC design.
- **Maximum capacitance constraint** behaves similarly to maximum transition constraint, but the cost is based on the total capacitance that a particular standard cell can drive any interconnection in the ASIC design. It should be noted that this constraint is fully independent of maximum transition, and therefore, it can be used in conjunction with maximum transition.

Floorplanning: Design Rule Checks

- **Maximum wire length constraint** is useful for controlling the length of wire to reduce the possibility of two parallel long wires of the same type. Parallel long wires of the same type may have a negative impact on the noise injection and may cause crosstalk.
- These design rule constraints are mainly achieved by properly inserting buffers at various stages of physical design. Thus, it is imperative to control the buffering in an ASIC design during place-and-route to minimize area impact.

Floorplanning: Design Planning

- Efficient design implementation of any ASIC requires an appropriate style or planning approach that enhances the implementation cycle time and allows the design goals such as area and performance to be met.
- There are two style small to medium ASIC's, flattening the design is most suited; for very large alternatives for design implementation of an ASIC–flat or hierarchical.

Floorplanning: Design Planning: Flat Implementation

- **The flat implementation** style provides better area usage and requires effort during physical design and timing closure compared to the hierarchical style.
- The area advantage is mainly due to there being no need to reserve extra space around each sub-design partition for power, ground, and resources for the routing.
- Timing analysis efficiencies arise from the fact that the entire design can be analyzed at once rather than analyzing each subcircuit separately and then analyzing the assembled design later.
- The disadvantage of this method is that it requires a large memory space for data and run time increases rapidly with design size.

Floorplanning: Design Planning: Hierarchical Implementation

- The hierarchical implementation style is mostly used for very large and/or concurrent ASIC designs where there is a need for a substantial amount of computing capability for data processing.
- In addition, it is used when subcircuits are designed individually. However, hierarchical design implementation may degrade the performance of the final ASIC.
- Therefore, when using a hierarchical implementation style, an ASIC design can be design implementation style one needs to assign the critical components to the same partition or generate proper timing constraints in order to keep the critical timing components close to each other and thus minimize the length of the critical path within the ASIC.
- The hierarchical design implementation style, an ASIC design can be partitioned logically and physically.

Floorplanning: Design Planning: Hierarchical Implementation

- **Logical partitioning** takes place in the early stages of ASIC design (i.e. RTL coding).
- The design is partitioned according to its logical functions, as well as physical constraints, such as interconnectivity to other partitions or subcircuits within the design.
- In logical partitioning, each partition is place and- routed separately and is placed as a macro, or block, at the ASIC top level.

Floorplanning: Design Planning: Hierarchical Implementation

- **Physical partitioning** is performed during the physical design activity. Once the entire ASIC design is imported into physical design tools, partitions can be created which combine several subcircuits, or a large circuit can be partitioned into several subcircuits.
- Most often, these partitions are formed by recursively partitioning a rectangular area containing the design using vertical or horizontal cut lines.

Floorplanning: Design Planning: Hierarchical Implementation

- **Physical partitioning** is used for minimizing delay (subject to the constraints applied to the cluster or managing circuit complexity) and satisfying timing and other design requirements in a small number of subcircuits.
- Initially, these partitions have undefined dimensions and fixed area (i.e. the total area of cells or instance added to the partition) with their associated ports, or terminals, assigned to their boundaries such that the connectivity among them is minimized.
- In order to place these partitions, or blocks, at the chip level, their dimensions as well as their port placement must be defined.

Floorplanning: Design Planning: Hierarchical Implementation

- One method that is suggested to estimate the perimeter of a macro instance is to use the number of terminals or ports allowed for each block and their associated spacing between each terminal.
- The relationship between the perimeter of each partition and the number of associated terminals is given by

$$P = NS\ ,$$

where is the perimeter of the physical partition block, P N is the number of terminals or ports, and corresponds to spacing between terminals.

Floorplanning: Design Planning: Hierarchical Implementation

- The perimeter estimate given by Equation determines an appropriate width and height for each partition in the hierarchy, based on the wire demand in both vertical and horizontal directions.
- However, in order to fit each macro instance at the chip top level in an effective manner, the automatic floorplan algorithm needs to have a range of legal shapes that is derived from aspect ratio bounds for each partition in the design.
- The aspect ratio bounds that are generated by the hierarchical floorplan algorithm must have the flexibility to ensure that each macro instance shape can be reshaped for optimum placement.

Floorplanning: Design Planning: Hierarchical Implementation

- Regardless of the physical design implementation style, after physical database creation using the imported netlist and corresponding library and technology files, the first step is to determine ASIC device and core width and height.
- In addition, standard cell rows and I/O pad sites are created. The rows and I/O pad sites are for standard cell and I/O pad placement.
- Figure 2-3 shows an initial ASIC design floorplan.
- The height of a row is equal to the height of the standard cells in the library. If there are any multiple-height standard cells in the library, they will occupy multiple rows

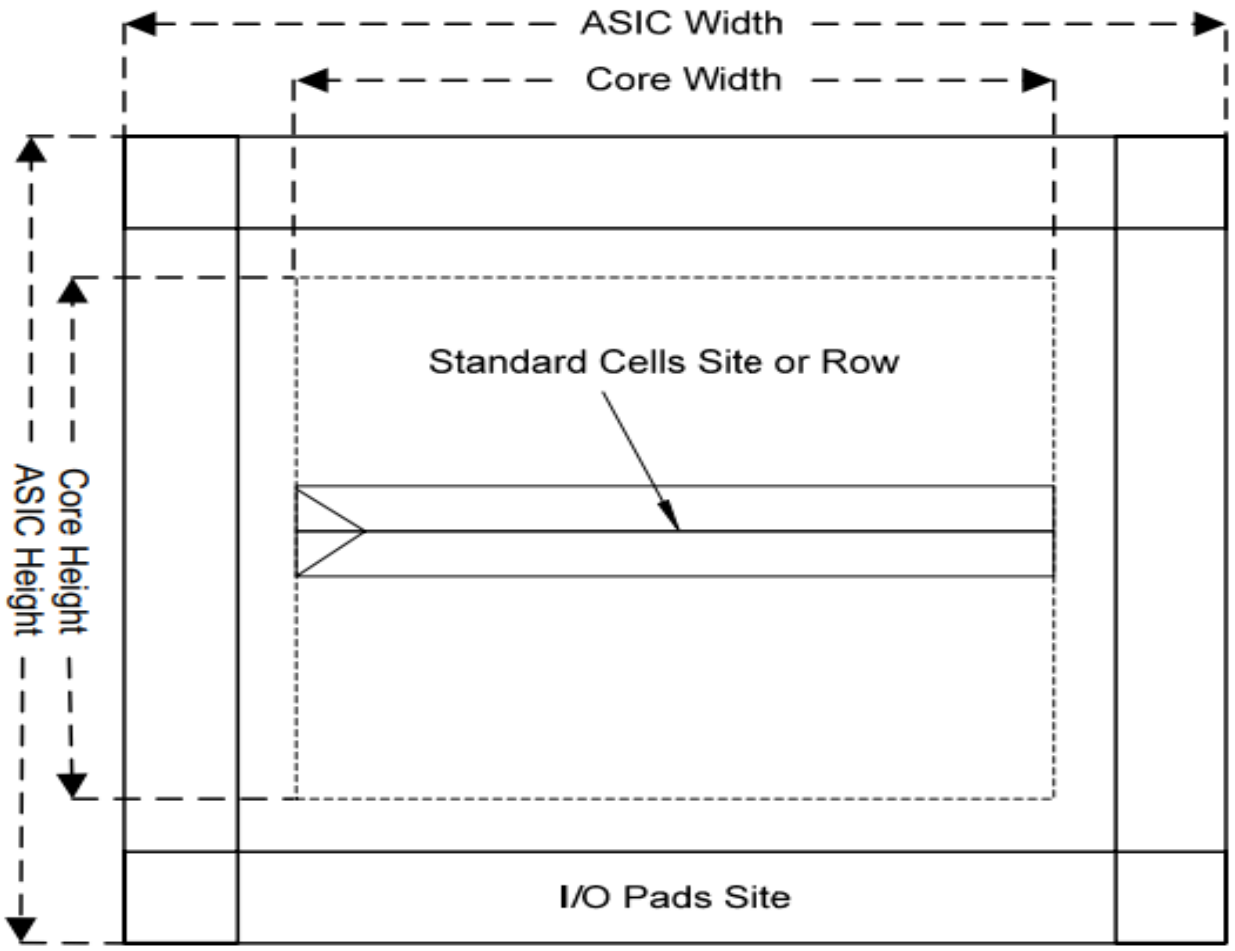


Figure 2-3 ASIC Design Initial Floorplan

Floorplanning: Design Planning: Hierarchical Implementation

- Most of the time, standard rows are created by abutment. The standard rows are oriented in alternating 180-degree rotation or are flipped along the X-axis so that the standard cells can share power and ground busses.
- If the ASIC core has routing congestion owing to the limited number of routing layers, one solution is to create routing channels between rows.

Floorplanning: Pad Placement

- Correct I/O pad placement and selection is important for the correct function of any ASIC design. ASIC design there are three types of I/O pads. These pads are power, ground, and signal.
- It is critical to functional operation of an ASIC design to insure that the pads have adequate power and ground connections and are placed properly in order to eliminate electromigration and current-switching noise related problems.
- Electromigration (EM) is the movement or molecular transfer of metal from one area to another area that is caused by an excessive electrical current in the direction of electron flow (electron “wind”).

Floorplanning: Pad Placement

- Electromigration currents exceeding recommended guidelines can result in premature ASIC device failure.
- Exceeding electromigration current density limits can create voids or hillocks, resulting in increased metal resistance, or shorts between wires, and can impair ASIC performance.
- Using Equation 2.5.1, one can determine the minimum number of ground pads required to satisfy the electromigration current limits.
- The required number of power pads is equal to the required number of ground pads and is given by:

$$N_{gnd} = \frac{I_{total}}{I_{\max}}, \qquad (2.5.1)$$

- where GND is the number of ground pads; I_{total} is the total current in an ASIC design (which is the sum of its static and dynamic currents in amperes); and I_{MAX} is the maximum EM current in amperes per ground pad.

Floorplanning: Pad Placement

- Switching noise is generated when ASIC outputs make transitions between states.
- An inadequate number of power and ground pads will lead to system data errors due to these switching noise transients. There are two types of mechanisms that can cause noise:
- **dv / dt** caused by a capacitive coupling effect
- **di / dt** caused by an inductive switching effect

Floorplanning: Pad Placement

- The capacitive coupling effect is the disturbance on the adjacent package pin caused when switching transients inject pulses via parasitic coupling capacitance.
- The **C (dv / dt)** maximum noise occurs when the ASIC output current nears the maximum current for a given capacitive output load of C . This noise problem can be resolved by proper pad placement, package pin selection, ASIC output pad type and drive current, and input pad type.

Floorplanning: Pad Placement

- To reduce or eliminate capacitive coupling effects during I/O pad placement and selection, one may consider the following guidelines:
  - Isolate sensitive asynchronous inputs such as clock or bidirectional pins from other switching signal pads with power or ground pads.
  - Group bidirectional pads together such that all are in the input or output mode,
  - Group slow input pads together positioning them on higher inductance package pins.
  - Use input pads with hysteresis as much as possible

Floorplanning: Pad Placement

- The maximum occurs when ASIC output starts to make the transition to another voltage level and its absolute current increases from L(di / dt) zero through a wire with inductance of L.
- Factors such as process, ambient temperature, voltage, location of output pads, and number of simultaneous switching output pads determine the magnitude of inductive switching noise.
- To control inductive switching noise, enough power and ground pads must be assigned and placed correctly. This way the noise magnitude will be limited. This noise reduction will prevent inputs of ASIC design from interpreting the noise as valid logic level.

Floorplanning: Pad Placement

Successful reduction of inductive switching noise can be accomplished by the following:

- Reduce the number of outputs that switch simultaneously by dividing them into groups with each group having a number of delay buffers inserted into their data paths.
- Use the lowest rated sink current or low-noise output pads as long as speed is not an issue.
- Place the simultaneously switching output or bidirectional pads together and distribute power and ground pads among them according to their relative noise rating.
- Assign static and low frequency input pads to higher inductance package pins.
- Reduce the effective power and ground pin inductance by assigning as many power and ground pads as possible

Floorplanning: Power Planning

- The next step is to plan and create power and ground structures for both I/O pads and core logic. The I/O pads' power and ground busses are built into the pad itself and will be connected by abutment.
  - For core logic, there is a core ring enclosing the core with one or more sets of power and ground rings.
  - A horizontal metal layer is used to define the top and bottom sides, or any other horizontal segment, while the vertical metal layer is utilized for left, right, and any other vertical segment.
  - These vertical and horizontal segments are connected through an appropriate via cut. The next consideration is to construct the standard cell power and ground that is internal to the core logic.
  - These internal core power and ground busses consist of one or two sets of wires or strips that repeat at regular intervals across the core logic, or specified region, within the design. Each of these power and ground strips run vertically, horizontally, or in both directions.

Floorplanning: Power Planning

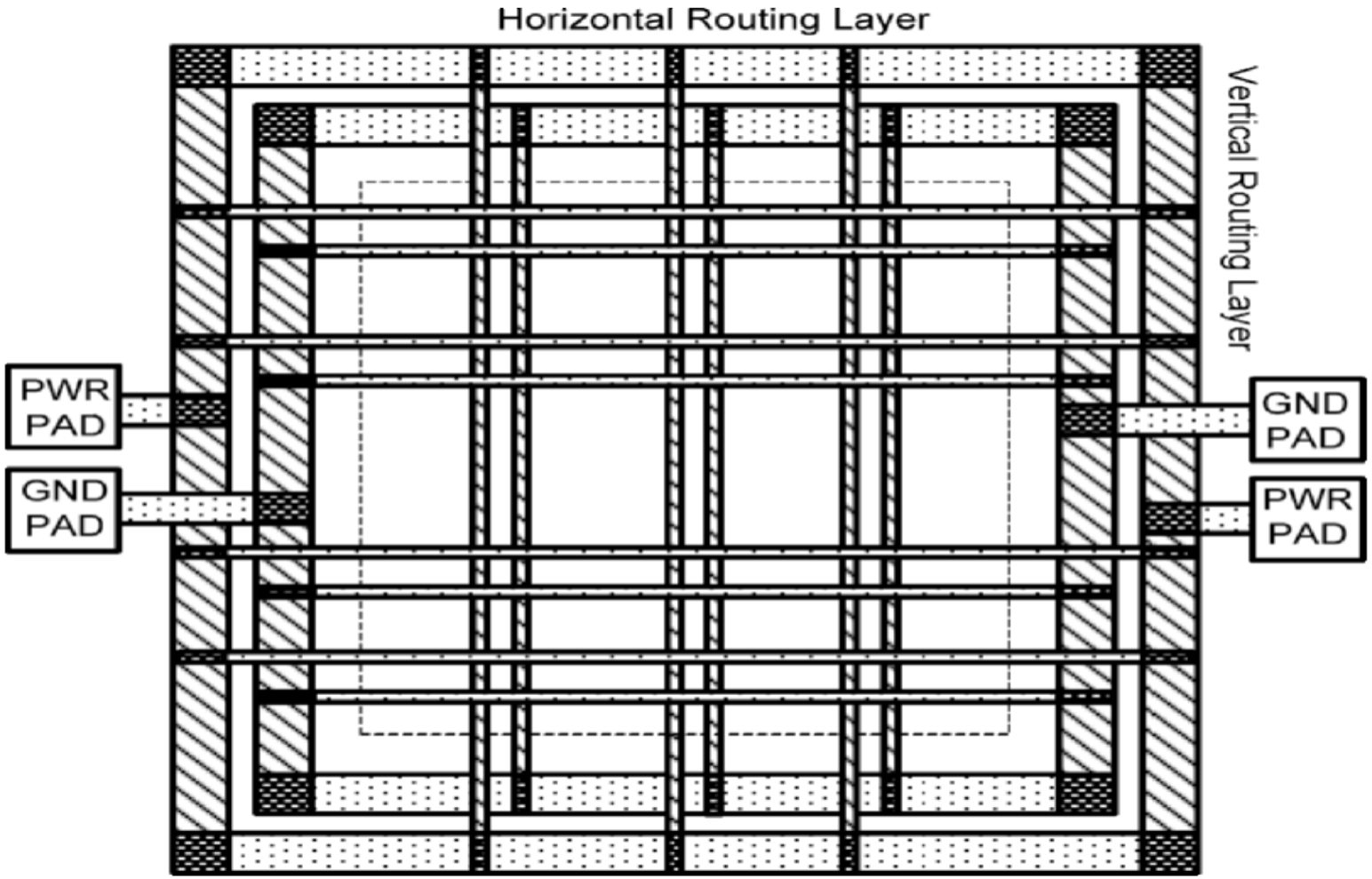


Figure 2-5 Ring and Core Power and Ground

Floorplanning: Power Planning

- If these strips run both vertically and horizontally at regular intervals, then the style is known as power mesh. The total number of strips and interval distance is solely dependent on the ASIC core power consumption.
- As the ASIC core power consumption (dynamic and static) increases, the distance of power and ground strip intervals increases.
- This increase in the power and ground strip intervals is used mainly to reduce overall ASIC voltage drop, thereby improving ASIC design performance.
- In addition to the core power and ground ring, macro power and ground rings need to be created using proper vertical and horizontal metal layers. A macro ring encloses one or more macros, completely or partially, with one or more sets of power and ground rings.

Floorplanning: Power Planning

- Another important consideration is that when both analog and digital blocks are present in an ASIC design, there is a need for special care to insure that there is no noise injection from digital blocks or core into the sensitive circuits of analog blocks through power and ground supply connections.
- Much of this interference can be minimized by carefully planning the power and ground connections for both digital core and analog blocks. There are several methods to improve the noise immunity and reduce interference.

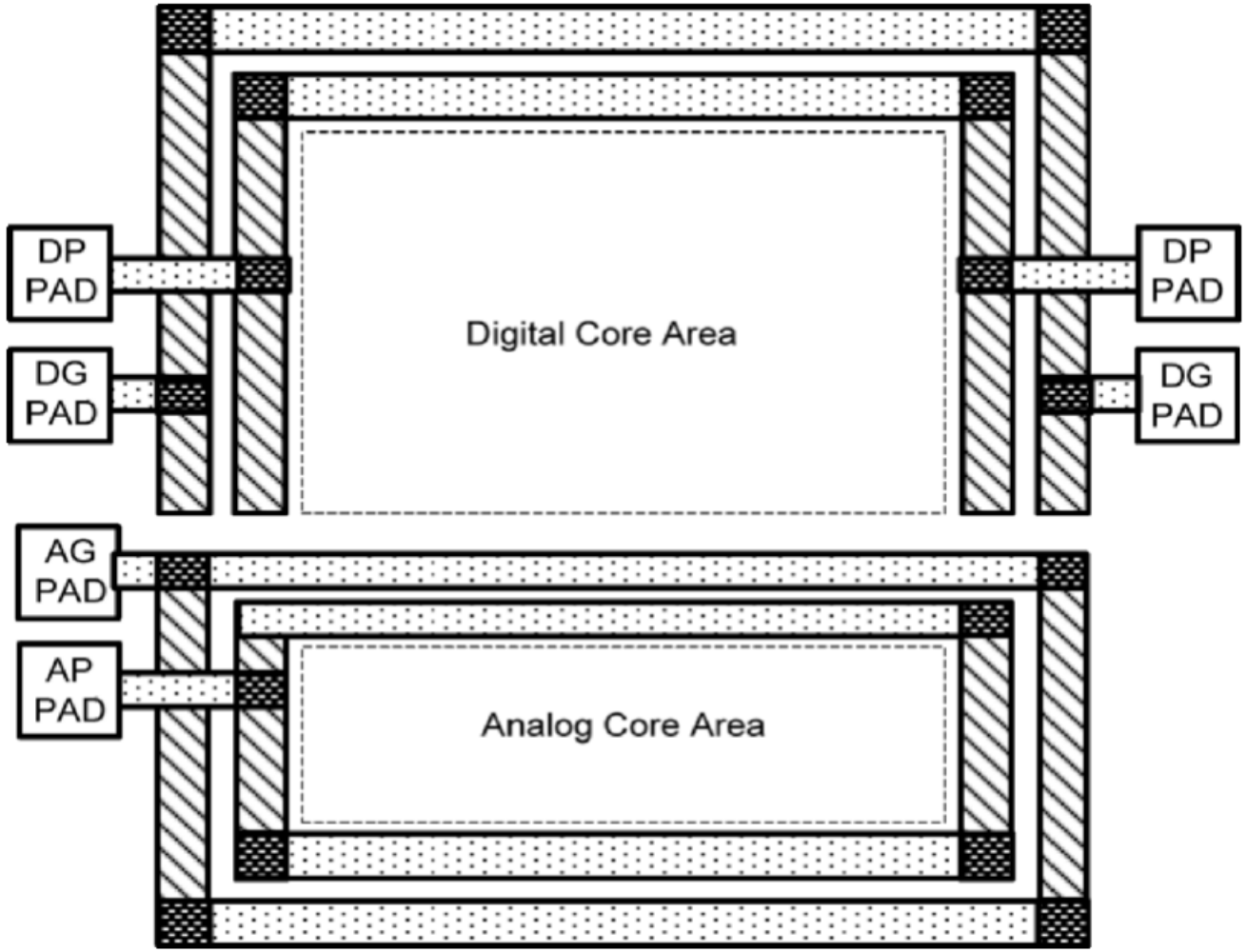


Figure 2-6 Decoupled Analog and Digital Core Power Supply

Floorplanning: Macro Placement

- Typically, the placement of macros takes place after I/O placement, and after power and ground bus structure has been defined.
- Macros may be memories, analog blocks, or in the case of hierarchical style, an individually placed and routed subcircuit.

- Macro placement can be manual or automatic. Manual macro placement is more efficient when there are few macros to be placed and their relationship with the rest of the ASIC design is known.
- Automatic macro placement is more appropriate if there is not enough information on which to base the initial macro placement and/or the number of macros is large.

Floorplanning: Macro Placement

- To avoid area segmentation, macros should be positioned such that the standard cell area is continuous.
- An area with close to 1:1 aspect ratio is recommended as it allows standard cell placers to utilize the area more efficiently and thereby reduce total wire length.
- The segmented floorplan leads to an excess of wire length interconnections from the standard cells located at the bottom of the die to those at the top of the die.
- Thus, it is necessary that the macros be kept along the ASIC core area in order to avoid a floorplan segmentation problem.

Floorplanning: Macro Placement

- Figure 2-7 shows a problematic segmented floorplan that may lead to long interconnections between the bottom and top of the die.
  - Another aspect of increased wire length is related to macro placement with respect to their orientation and pin placements.
  - Depending on the macro orientation and their actual pin location, the length of the nets connecting to the macros can be different, and can have significant impact on the routing optimization process.

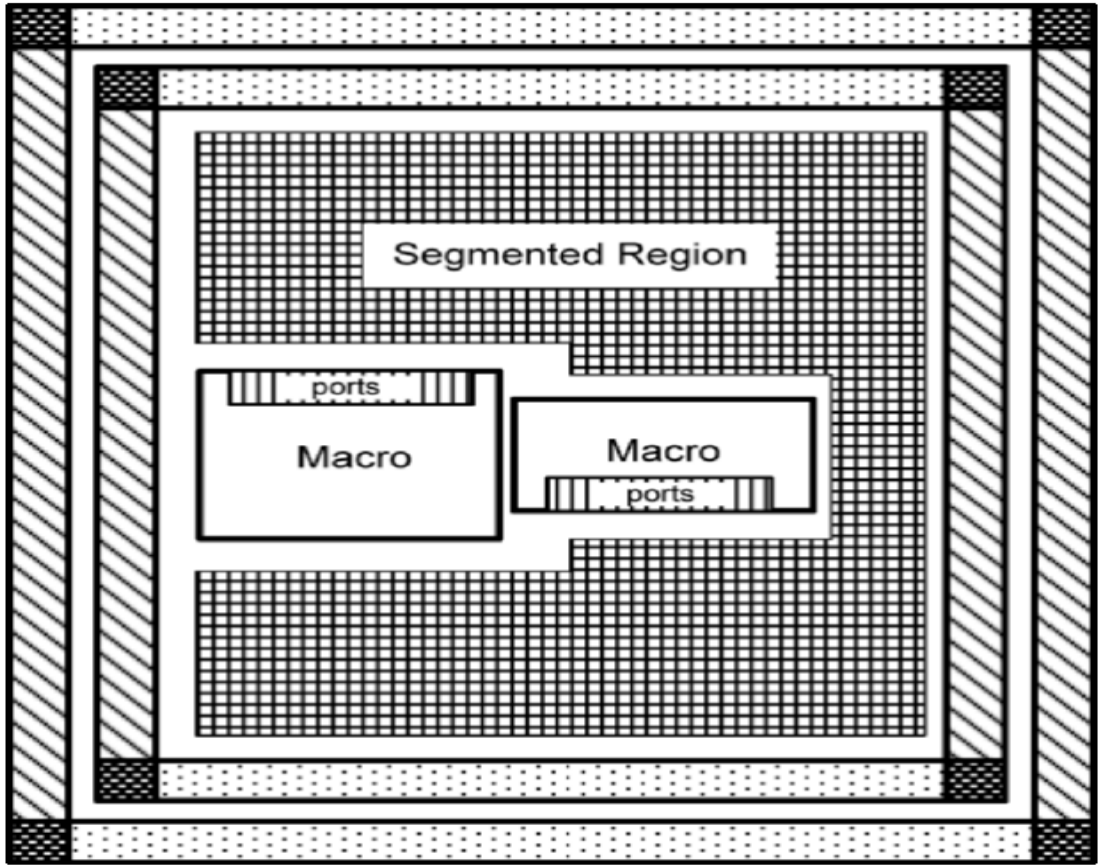


Figure 2-7 Segmented Floorplan

Floorplanning: Macro Placement

- Figure 2-8 shows a floorplan with macro ports facing the standard cell region, thereby minimizing localized increase in wire length.
  - Another aspect of physical measure is that of macro placements relative to each other, as well as to the standard cell placement, and the macro port accessibility has a direct impact on the chip's final routing.
  - The macro placement, and thereby the quality measure, can be determined by the analysis of routing congestion produced by a global router.

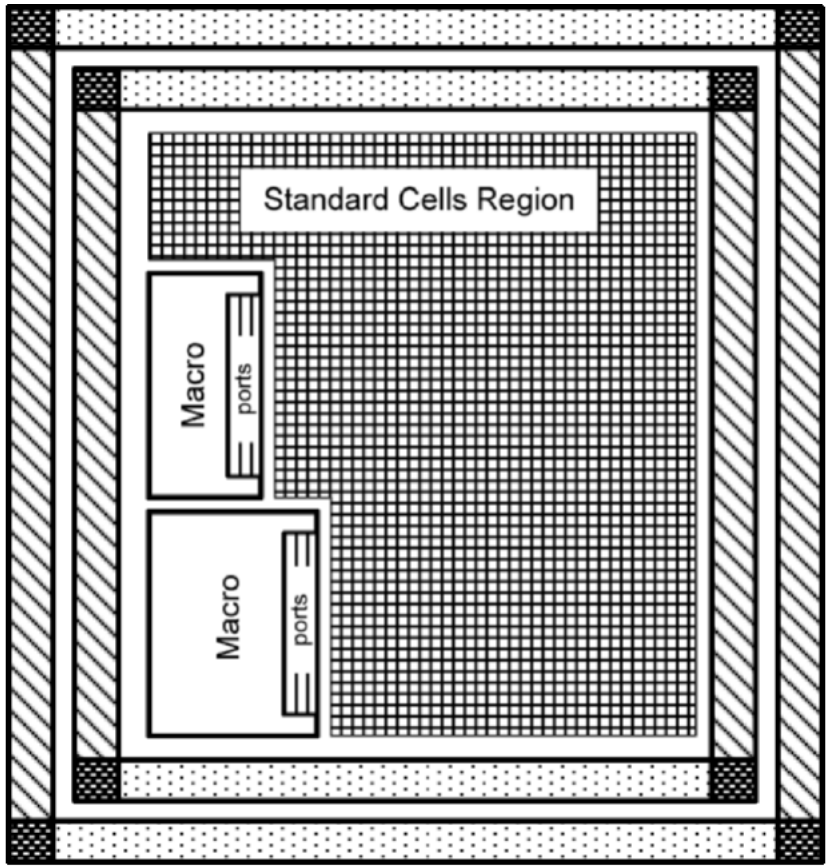


Figure 2-8 Floorplan with Macros Facing Standard Cells Region

Floorplanning: Macro Placement

- The most common scenarios that cause physical design routing congestion are: there may not be enough space between the macros to provide routing placed at the ASIC periphery and over the macro); routing is prohibited; or standard cell trap pockets may be around the edges of macros or within the corners of the floorplan.
- Standard cell trap pockets are long, thin channels between macros. If many cells are placed in these channels, routing congestion can result.
- Therefore, these channels need to be kept free for most standard cells and should be available for repeater or buffer insertion (if this type of insertion is supported by the physical design tool).
- Figure 2-9 shows a floorplan with a standard cell trap pocket.

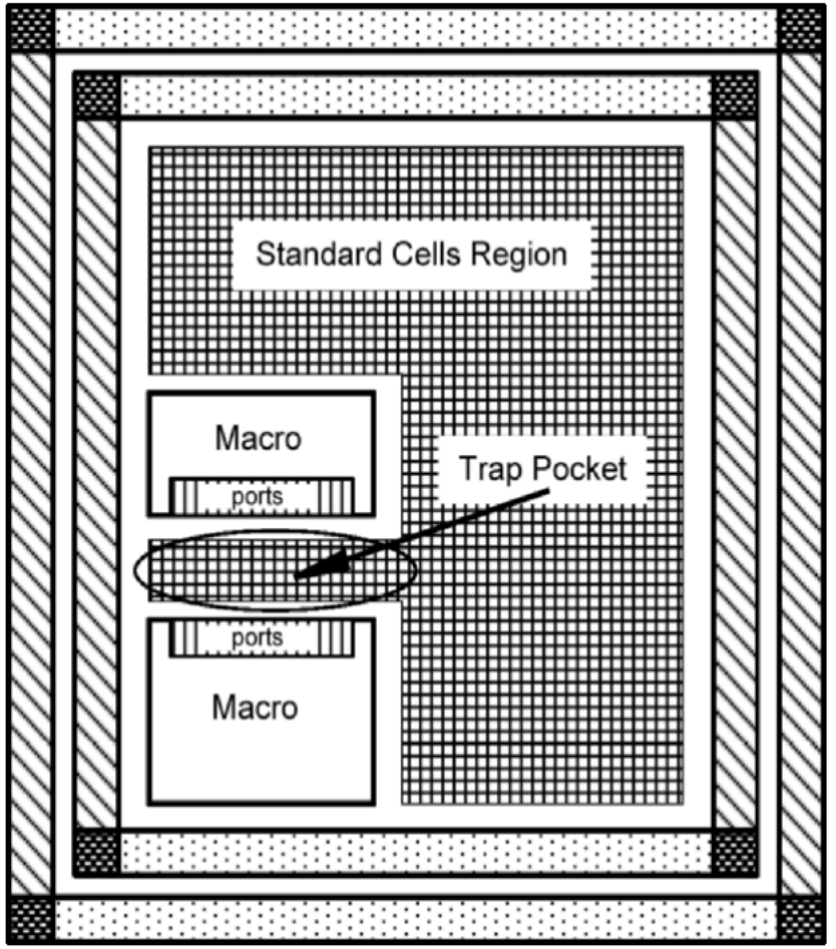


Figure 2-9 Floorplan with Standard Cells Trap Pocket

Floorplanning: Macro Placement

- Blockage layers are created over pre-placed macros such that their power and ground rings are covered. Blockage layers are also used to relieve routing congestion around the macro's corners. When a macro is blocked on many routing layers, wires have a tendency to detour around corners and connect to nearby standard cells thereby creating routing congestion at the corner of the macros.
- To reserve more resources for the router, one can draw a blockage layer at these corners. These blockage regions can be simple or gradual as illustrated in Figure 2-10.

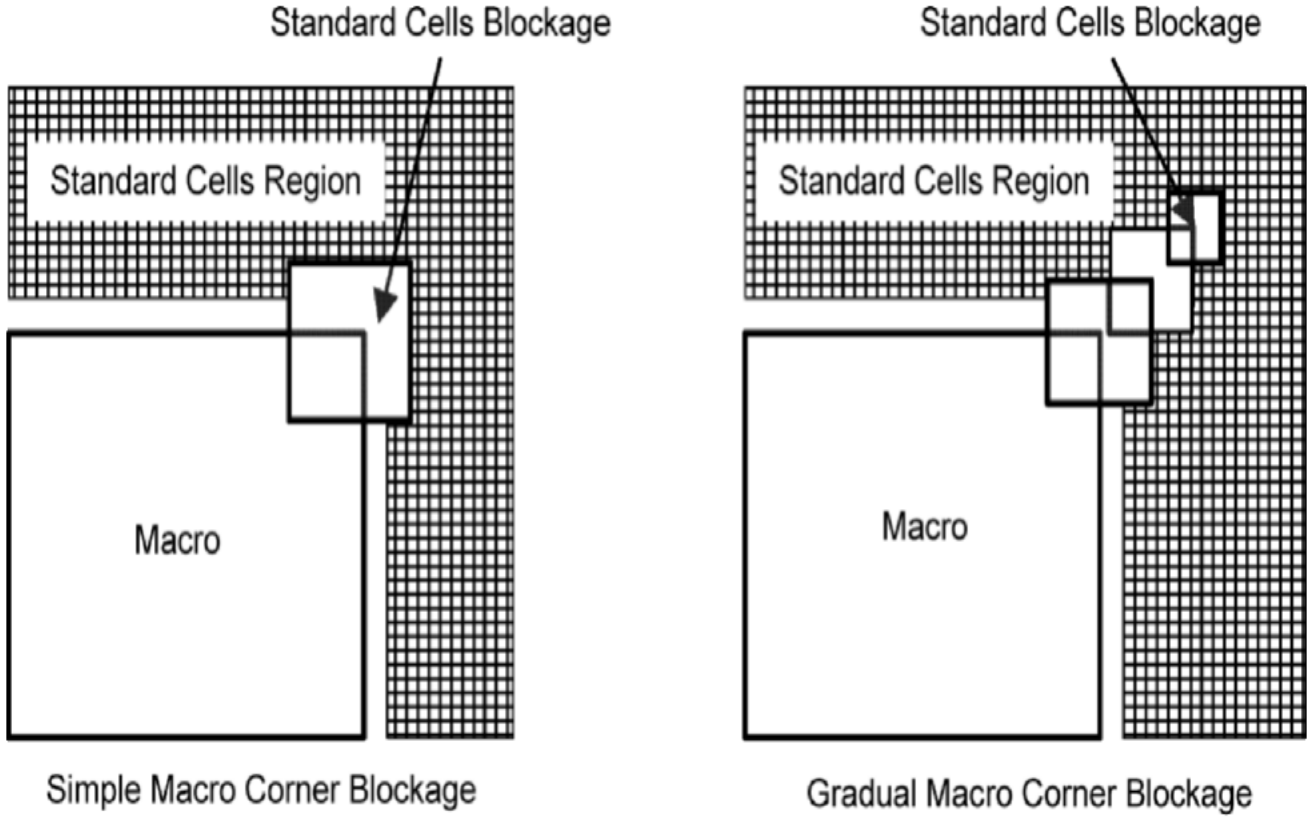


Figure 2-10 Macro Corner Standard Cells Blockage

Floorplanning: Macro Placement

- After refinement of floorplan and macro placement, standard cells are placed and connectivity analysis is performed.
- Connectivity analysis is the process of studying the connections between macro pairs, macro, I/O pads, and related standard cell instances.
- Connectivity analysis is also used to identify macros that have substantial direct connectivity and to refine their locations accordingly

Floorplanning: Clock Planning

- Clock planning is considered another part of floorplanning. The idea of the implementation of clock distribution networks is to provide clock to all clocked elements in the design in a symmetrically-structured manner.
- Although most ASIC designs use clock tree synthesis, clock tree synthesis may not be sufficient for very high-performance and synchronized designs.
- In this case, one needs to implement the distributed clock networks manually in order to minimize the skew between communicating elements due to their line resistance and capacitance.

Floorplanning: Clock Planning

- The basic idea of manual implementation of clock distribution networks is to build a low resistance/capacitance grid similar to power and ground mesh that covers the entire logic core area.
- In order to minimize such clock skew, a clock tree that balances the rise and fall time at each clock buffer node should be utilized during clock planning
- It is essential to realize that clock grid networks consume a great deal of power due to being active all the time and it may not be possible to make such networks uniform owing to floorplanning constraints (e.g. to spread the power dissipation evenly across the chip).

Floorplanning: Clock Planning

- Figure 2-12 illustrates typical hierarchical clock planning.

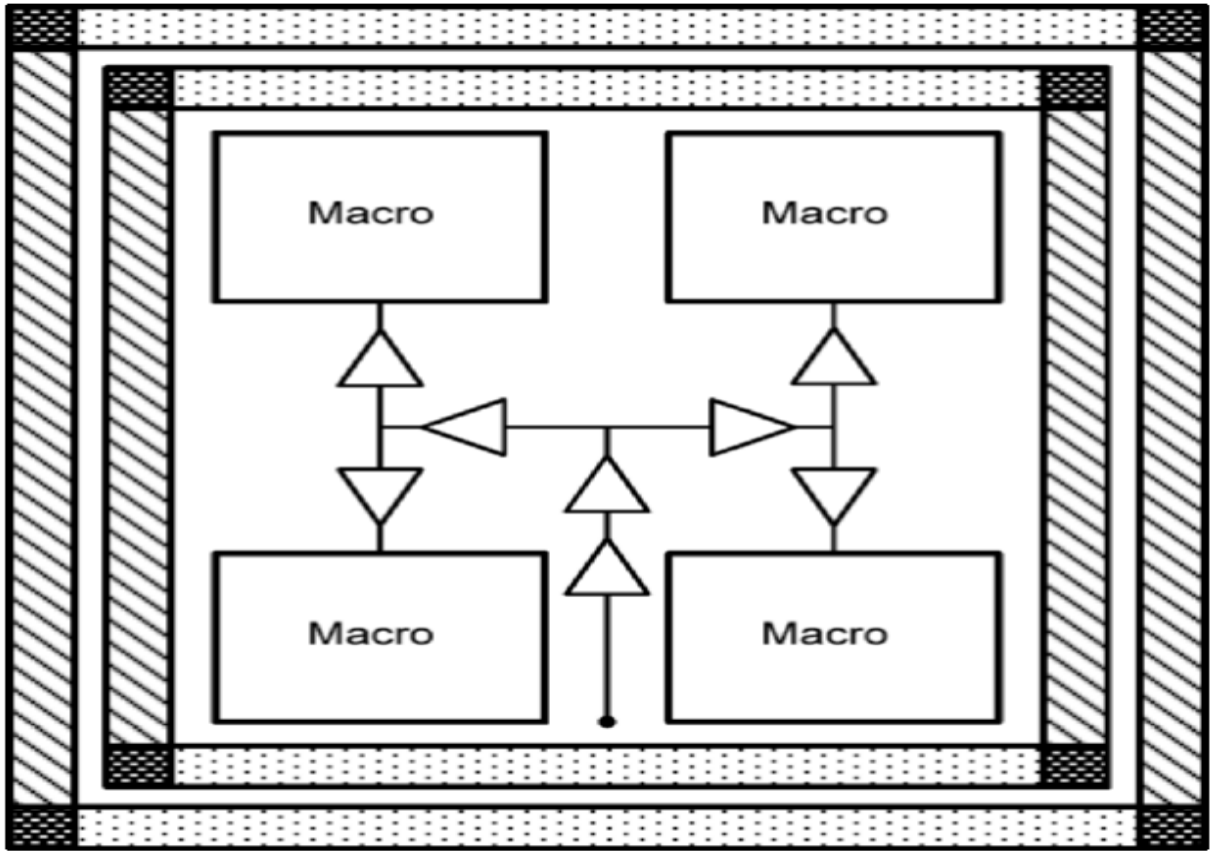


Figure 2-12 Hierarchical Clock Planning

References

- “Physical Design Essentials, An ASIC Design Implementation Perspective”. Khosrow Golshan.

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu

Digital System Design

Dr. Sudeendra kumar K
Department of Electronics and Communication Engineering

DIGITAL SYSTEM DESIGN

Placement

Sudeendra kumar K
Department of Electronics and Communication Engineering

Introduction to Placement

- Standard cell placement and insertion of buffers along clock paths or Clock Tree Synthesis (CTS) are the most important and challenging phases in ASIC physical design.
- The goal of standard cell placement is to map ASIC components, or cells, onto positions of the ASIC core area, or standard cell placement region, which is defined by rows.
- The standard cells must be placed in the assigned region (i.e. rows) such that the ASIC can be routed efficiently and the overall timing requirements can be satisfied.
- Standard cell placement of an ASIC physical design has always been a key factor for achieving physical designs with optimized area usage, routing congestion, and timing behavior. Almost all of today's physical design tools use various algorithms to place standard cells automatically.

Global Placement

- When the floorplan is first created, standard cells are in a floating state. This means that they are placed arbitrarily in the ASIC core and have not been assigned to a fixed location within the standard cell rows.
- At this time one can partition the standard cell area and assign a group of cells to these partitions, or simply group a set of standard cells.
- Almost every place-and-route tool **supports cluster and region options.** These two options are used to guide placement algorithms during standard cell placement.

Global Placement

- Cluster refers to a group of standard cells that, during placement, are placed near each other. The location of the cluster is undefined until all standard cells have been placed.
- This option is mainly used to control the closeness of timing-critical components during placement and resembles a module definition in the structural netlist.

Global Placement

- Figure 3-1 Standard Cell Cluster Region is very similar to the cluster method with the exception that the location of the region is defined prior to standard cell placement.
- The way this option is implemented is that a cluster or group of standard cells is created and then assigned to a particular area on the core ASIC. algorithms (e.g. interconnect driven), this option has been rarely used –

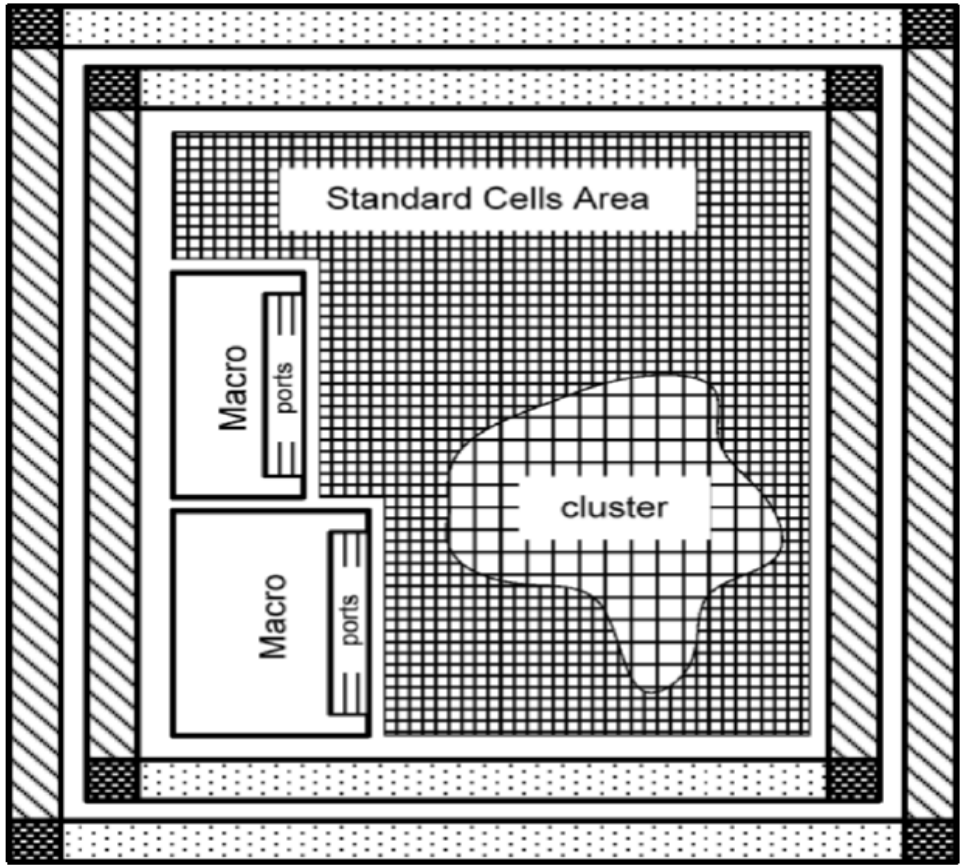


Figure 3-1 Standard Cell Cluster

Global Placement

- Region is very similar to the cluster method with the exception that the location of the region is defined prior to standard cell placement.
- The way this option is implemented is that a cluster or group of standard cells is created and then assigned to a particular area on the core ASIC.
- Regions can be soft or hard. Soft region, is physical constraint where logical module is assigned to a location in the core and boundary of the region, it is subject to change during standard cell placement.

Global Placement

- Hard region, however, is more rigorous than the soft region and defines a physical partition for modular design. It has “hard” boundaries that prevent standard cell crossing during placement.
- Using the hard region option, one must define the location as well as the shape of the region. This option is used primarily for timing related issues such as grouping clock, voltage, or threshold voltage domains.

Global Placement

- In addition, a region can be exclusive or non-exclusive. An exclusive region only allows standard cells assigned to the region to be placed within the region.
- On the other hand, a non-exclusive region will allow standard cells that do not belong to the region to be placed within it.
- A hard region (or an exclusive region with a predefined physical boundary) might be used to enforce a floorplan consisting of separate blocks.
- This approach is useful for dividing the ASIC core area into regions that have different functions or physical aspects.
- For example, a hard region can be used to partition the ASIC core area so one region has a different voltage from the rest of the design or other regions as shown in Figure 3-2.

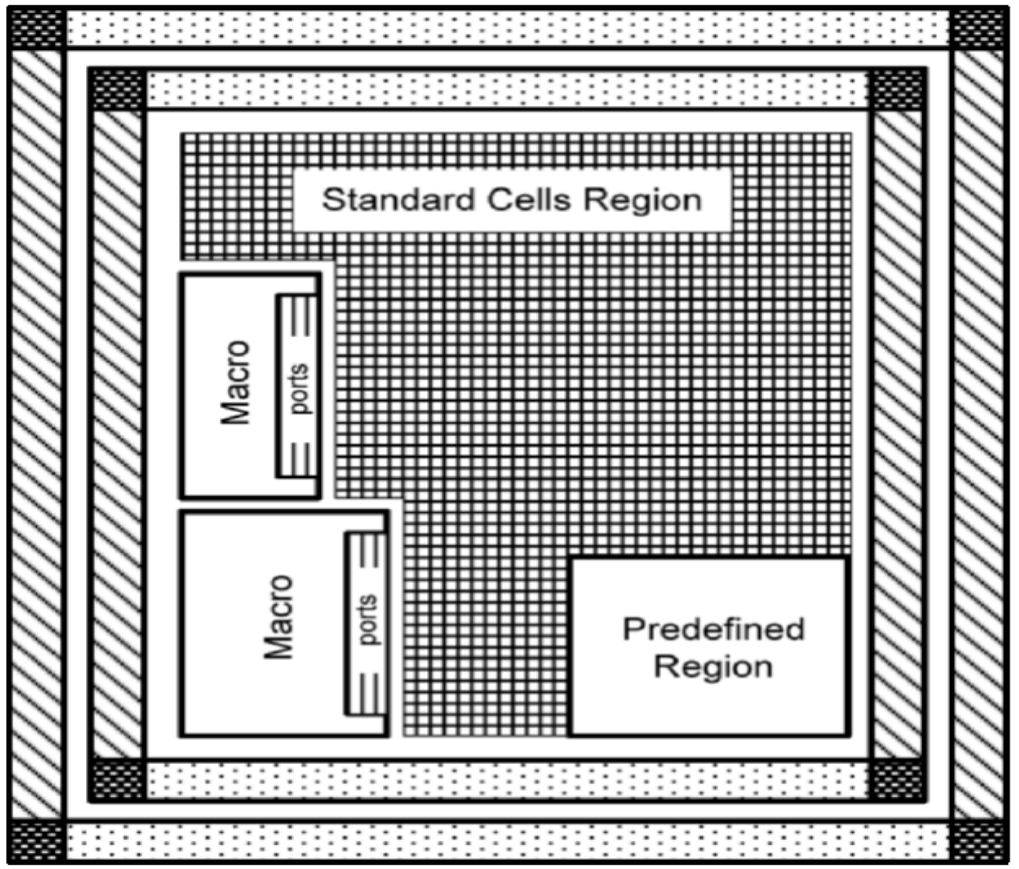


Figure 3-2 Predefined Region

Global Placement

- After clusters and regions are defined, global placement algorithms begin distributing standard cell instances uniformly across the available ASIC core and use a method of estimation to minimize wire lengths.
- During this time, ASIC design is recursively partitioned along alternatively horizontal and vertical cut lines, and standard cell instances are assigned to rectangular bins, or slots, taking partitioning into account.

Global Placement

- Then, instances in each bin will be moved across each cut line in order to minimize the number of connections between each partition.
- The procedure of partitioning and moving standard cell instances across cut lines terminates when certain stop criteria are satisfied (e.g. total number of standard cell instances in each bin).
- After design partitioning is completed, a legalization step is executed to remove any standard cell overlap and fit current placement into row structures.

Global Placement

- The problem with nets that have a large number of fan-outs, such as a reset signal, is that one source drives many standard cells across the ASIC core.
- Although these nets are not critical from a timing perspective, they have a strong impact on the core routing area due to their global nature.
- Thus, reducing the total number of these fan-outs will improve the overall routing of the ASIC core area. Reducing the number of fan-outs (or connections) to between 40 and 50, to which one standard cell connects, is reasonable.

Global Placement

- Long wires are not the same as high fan-out nets as they are not global in nature. They have very small fan-outs, but the driver instance is located far from the receiver.
- Often this situation is a result of the receiver having very strong connectivity to instances other than the driver. These types of long wires are highly resistive and can create large input transition times at the input port of the receiver cell.
- This large input transition time increases the receiver cell's propagation delay. Therefore, it is beneficial to segment these long wires using buffers.

Global Placement

- Another timing optimization that is used during global placement is logic restructuring.
- Logic restructuring is mainly supported by physical synthesis tools that combine several primary logic functions into a few standard cells, decompose functional gates into their equivalent primary logic gates, and/or duplicate combinational logic (cloning).
- The logic restructuring algorithm used during global placement mainly focuses on logic reconstruction of critical paths that are not meeting required timing constraints.
- The objective of the algorithm is to rearrange logic in the critical paths such that the timing constraints are met.

Detail Placement

- Once all standard cell instances are placed globally, a detail placement algorithm is executed to refine their placement based on congestion, timing, and/or power requirements.
- Congestion refinement or congestion-driven placement is more beneficial to ASIC designs with very high density, and the objective of the detail placer is to distance standard cell instances from each other such that more routing tracks are created among them.
- The quality of congestion placement directly relates to how well the global placer partitions the design and could have a negative impact on the device size and performance.

Detail Placement

- Timing-driven placement algorithms have been classified as either net or path based.
- Net-based schemes try to control the delay on a signal path by imposing an upper-bound delay or by assigning a weight to each net.
- Path-based approaches apply constraints to delay paths of small subcircuits (the disadvantage of path-based algorithms is the fact that it is impossible to enumerate all paths within a design).
- The major challenge in timing-driven placement is to optimize large sets of path delays without enumerating them in the ASIC design.
- This optimization is accomplished by interleaving weighted connectivity-driven placements with timing analysis that annotates individual instances, nets, and path delays with design constraint information.

Detail Placement

- To meet these types of design constraint, various placement techniques have been proposed or used. The most well-known detail placement method is simulated annealing. Not only is simulated annealing efficient, it can also handle complex design constraints.
- Simulated annealing is a simulation-based placement technique and is used as an iterative improvement algorithm during the detail placement process.
- The objective of this procedure is to find an optimal or near-optimal placement for each pre-placed standard cell instance.

Detail Placement

- This difference in the wire model and actual wire load (e.g. after routing) can lead to an endless iterative process during ASIC timing closure.
- To overcome this iterative process, the use of global routing during load-based optimization is suggested, as long as the initial global and final detail routes correlate closely.

Detail Placement

- For deep submicron and predictability issues associated with wire capacitance models, gain-based optimization is becoming widely used due to the acceptance of the load-independent cell delay concept.
- The idea of gain-based optimization relies on the concept of logical effort. The method of logical effort simply reformulates a classical CMOS delay model, and expresses it with some new terminology.

Detail Placement

- The CMOS delay model contains two parameters: intrinsic (which is constant) and extrinsic (which is proportional to the output capacitance load). In the logical effort model, the intrinsic value is known as the parasitic delay, and the extrinsic value is referred to as the effort delay. Using parasitic delay and effort delay, then the total delay through a CMOS gate is given by

$$d = p + f$$

where p is parasitic delay and f is effort delay. Both are measured in units of τ. The τ value is based on the characterization of the semiconductor process used for a given ASIC design.

Detail Placement

- The effort delay can be viewed as having an electrical effect on the output capacitance load as well as having a logical effect that is the ability to drive the capacitive load. Thus, the effort delay is expressed as

$$f = gh\,,$$

Where **g** is logical effort and **h** is electrical effort.

- The logical effort determines the standard cell's capability of producing the output current based on its topology and is independent of the size of transistor used in its circuit.
- In other words, logical effort describes the delay in terms of complexity rather than transistor size. Thus, the more complex the standard cell, or gate, is, the slower it becomes.

Detail Placement

- Logical effort is defined by the ratio of the total number of transistor units in a standard cell circuit to the total number of transistor units in an inverter circuit (the smallest unit in the library). The ratio is expressed as

$$g = \frac{T_{gate}}{T_{inverter}},$$

where T_{gate} is the total number of standard cell transistors and $T_{inverterv}$ is the total number of transistors in an inverter circuit. In a typical inverter with an equal rise and fall delay time with two units of PMOS and one unit of NMOS transistor, the value of $T_{inverterv}$ and T_{gate} is equal to three and is considered to have a logical effort equal to one.

Detail Placement

- The electrical effort (or gain) captures the electrical behavior and performance of the standard cell associated with its output capacitance load. Essentially, this parameter represents the standard cell's load-driving capability and is given by

$$h = \frac{C_l}{C_i},$$

Where $\mathbf{C_l}$ is the output capacitance and $\mathbf{C_i}$ is the input capacitance.

Detail Placement

$$d = gh + p\,. \qquad (3.2.6)$$

In Equation 3.2.6, g, h, and p parameters contribute to standard cell delays separately. In the standard cell delay calculation, parameters g and p are independent of the transistor size, while parameter h directly relates to the transistor size. Typically, standard cell delays are measured using τ parameter. The τ parameter is simply the delay through the smallest inverter for a given standard cell library. Typically, an inverter chain is used to measure the value of τ.

Detail Placement

- During gain-based optimization, the gain for each standard cell along the critical path is computed. Then the algorithm tries to maintain the gains within a given timing path (equal gain for each component in the timing path means optimal timing).
- In this process if an instance requires more gain due to an increase in its output capacitance loading, then its input capacitance will be increased in order to maintain the original gain.
- This technique of changing the input capacitance rather than changing the underlying standard cell drive strength exhibits wire capacitance load independency.

Detail Placement

- Apart from congestion and/or timing optimization, one can consider power minimization during detail placement or after the clock tree synthesis. There are two main sources of power consumption for an ASIC design—dynamic and static.

Clock Tree Synthesis

- The concept of clock tree synthesis (CTS) is the automatic insertion of buffers/inverters along the clock paths of the ASIC design to balance the clock delay to all clock inputs. Naturally, clock signals are considered global (or ideal) nets.
- These nets exhibit high resistance and capacitance due to their being very long wires. The principle of CTS is to reduce the delay associated with these long wires. These long wires can be modeled as distributed networks such that

$$C\frac{dV}{dt} = \frac{(V_{i-1} - V_i)}{R} - \frac{(V_i - V_{i+1})}{R}$$

- where V is voltage at a point i in the wire, and R and are resistance and capacitance of each wire segment as shown in Figure 3-8.

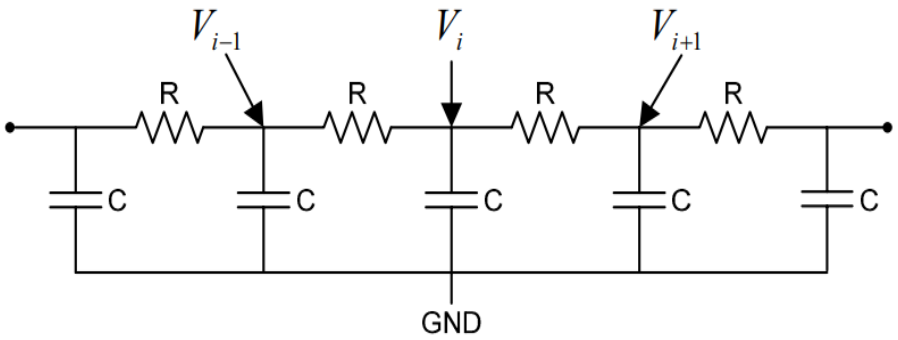


Figure 3-8 Distributed RC Network

Clock Tree Synthesis

- In practice, the actual size of these tapering buffers may not be identical.
- In fact, to obtain optimal propagation delay, these buffers may be selected to monotonically increase in their drive strength **d** by a factor **α** for each level of clock tree path, as shown in Figure 3-9.

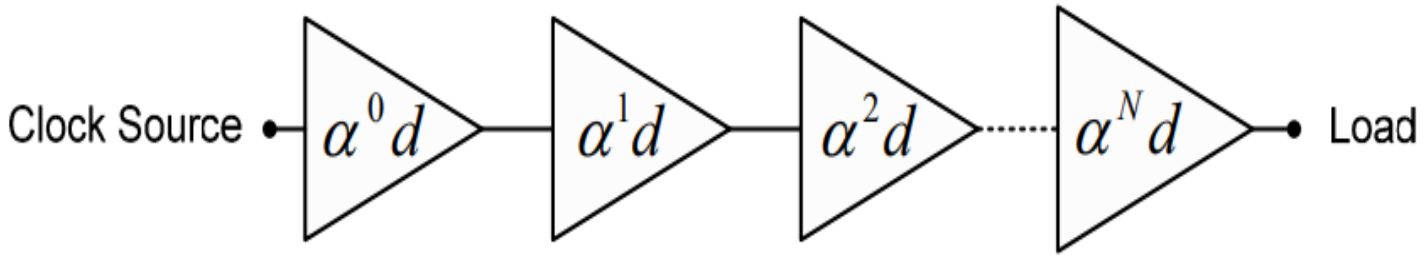


Figure 3-9 Cascaded Buffers

Clock Tree Synthesis

- It is recommended for buffers that are used for tapering the clock paths to have equal rise and fall delay time.
- The main reason for having equal rise and fall time is to maintain the original duty cycle and to insure that there is no clock signal overlap due to any difference in propagation delays.
- The proper usage of balance buffers or inverters during clock tree synthesis is extremely important, especially when dealing with very high-speed clocking (i.e. clock pulse with small period) requirements.

Clock Tree Synthesis

- In dealing with very high-speed clocking, if the clock buffers or inverters are not selected correctly they may cause the clock pulse width to degrade as the clock propagates through them before reaching the final destination.

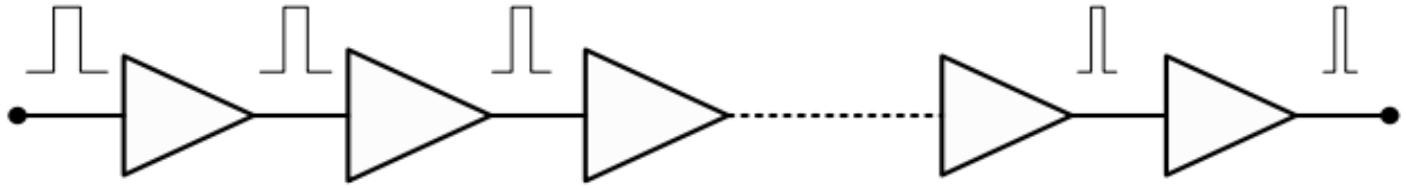


Figure 3-10 Clock Pulse Width Degradation Effect

Clock Tree Synthesis

- Often it is difficult to achieve perfect signal rise and fall time balance in the context of an ASIC design during clock tree synthesis. Thus, one remedy to overcome clock pulse narrowing is to use inverters rather than buffers.
- Most clock tree synthesis algorithms use either the **Sum ( ) or Pi (π) configuration** during the clock buffer insertion along each clock path.

Clock Tree Synthesis

- In the Sum configuration, the total number of inserted buffers is the sum of all buffers per level.
- This type of structure exhibits unbalanced trees due to the different number of buffers and wire lengths.
- In this method, the systematic skew is minimized through delay matching along each clock path and, due to its non-symmetrical nature, is highly process-corner-dependent.

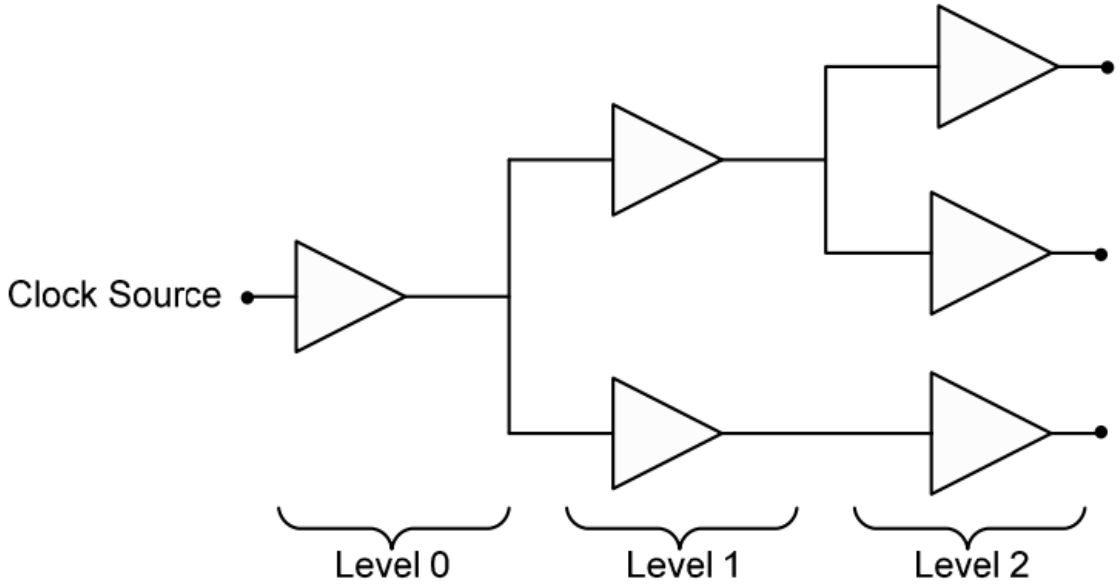


Figure 3-11 Sum ($\sum$) Configuration

In this configuration, the total number of clock buffers is given by

$$N_{total} = n_{level0} + n_{level1} + n_{level2} + \ldots\ldots + n_{level\,n}. \qquad (3.3.8)$$

Clock Tree Synthesis

- In the Pi configuration, the total number of buffers inserted along clock paths is a multiple of the previous level.
- This type of structure uses the same number of buffers and geometrical wires and relies on matching the delay components at each level of the clock tree.
- The Pi structure clock tree is considered to be balanced, or symmetrical, and the total number of buffers is given by

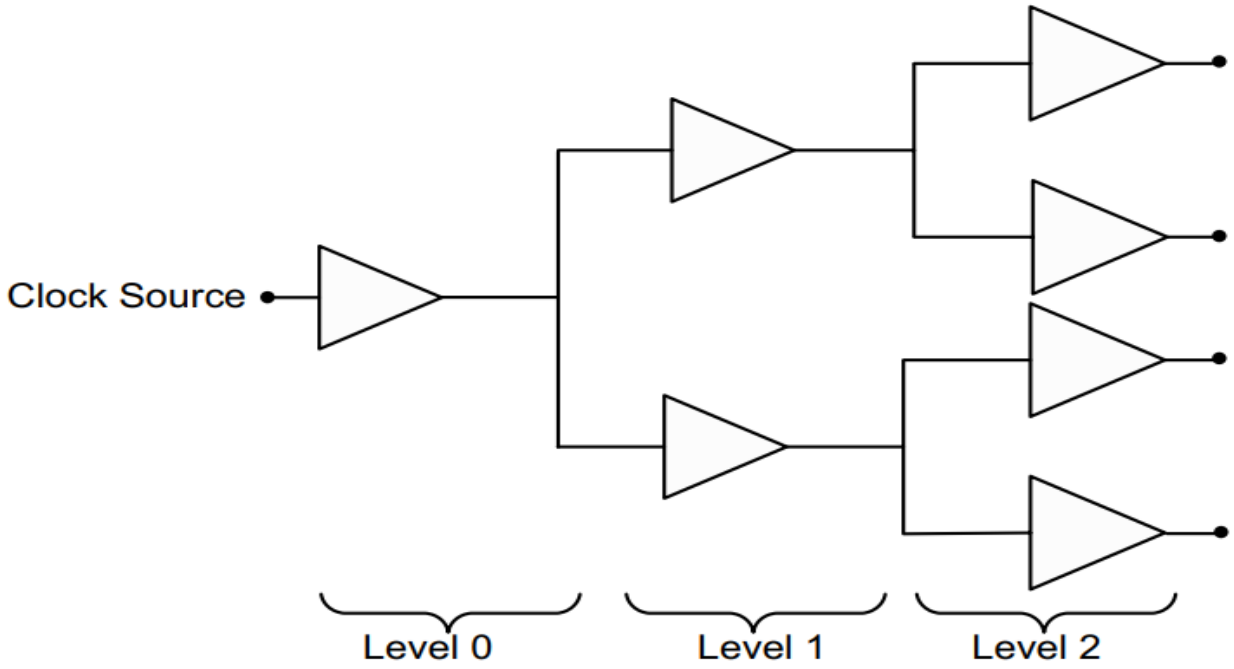


Figure 3-12 Pi ($\prod$) Configuration

$$N_{total} = (n_{level0} \times n_{level1}) + (n_{level1} \times n_{level2}) + .. + (n_{level(n-1)} \times n_{level\ln}) . \quad (3.3.9)$$

Clock Tree Synthesis

- In contrast to the Sum configuration, in a Pi configuration, the systematic skew is minimized due to symmetry, and the variation in clock skew is determined by process uniformity rather than by process corner.
- In today's ASIC designs where dynamic power consumption plays an important role, Sum configurations are most often used regardless of the number of clock domains.
- This is because the total number of buffer insertions during clock tree synthesis is less than the total number of buffers used in the Pi configuration.
- Regardless of which configuration is used to insert buffers during clock tree synthesis, the objective that governs the clock tree quality is achieving minimal skew while maintaining an acceptable clock signal propagation delay.

Clock Tree Synthesis

- Clock skew can be local in the case of multiple clock domains or global in the case of a single clock domain. The root cause of clock skew can be systematic due to non-uniformity of clock routing and/or randomly caused by differences in delay components.
- In the past, variation in component delay was observed from lot-to-lot (a set of wafer undergoes the process at the same time).
- As ASIC manufacturing processes advanced, variations became apparent in both wafer-to-wafer and The variation in delay components is mainly due to the different device sizes, process gradients, etching effects, supply voltage variation, temperature gradients, or mismatched minimum transistor channel lengths.
- Any one of these effects can influence the clock propagation delay along each clock path die-to-die.

Placement: Power Analysis

- Analysis relies on knowing the operating frequency of each net and corresponding load capacitance in an ASIC design.
- In practice, this is very computationally intense (e.g. functional simulation of the entire ASIC to exercise every net) and is becoming almost impossible to obtain meaningful results.
- To overcome this dynamic power analysis problem, one method is to perform a static power analysis based on the power distribution systems (i.e. power and ground routing) that are designed to provide needed voltages and currents to the transistors that perform the logical functions.

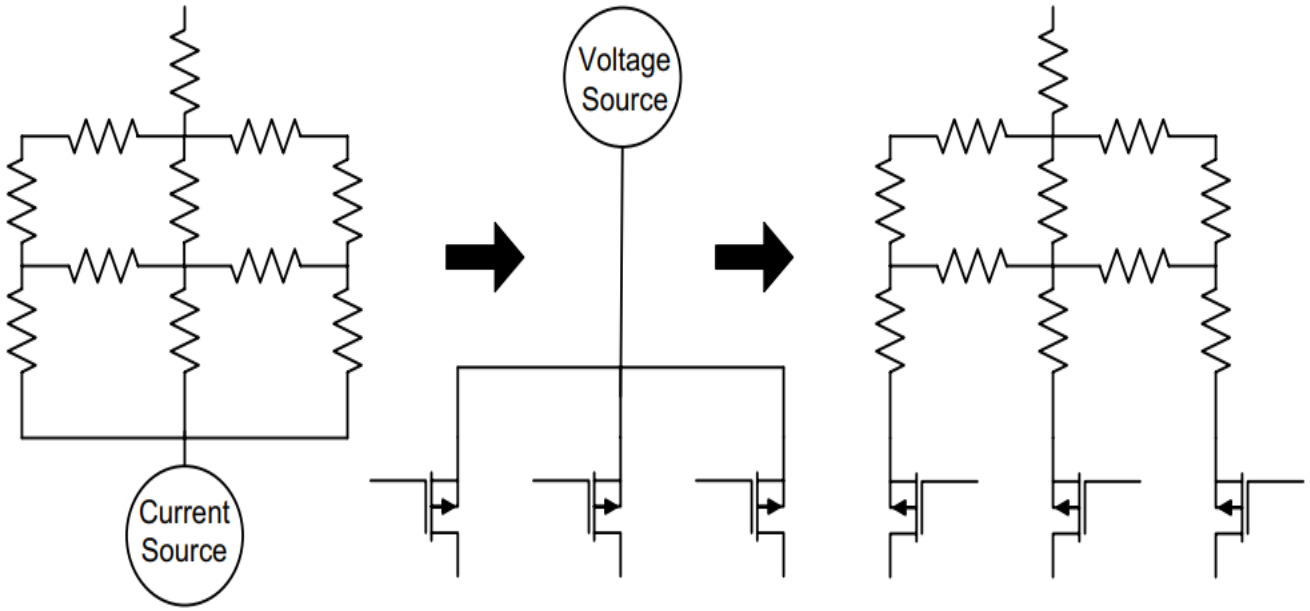


Figure 3-16 Static Power Calculation Method

Placement: Power Analysis

- Most static power analysis algorithms are based on Ohm's and Kirchoff's laws. In these methods, the resistances of power and ground routing are extracted and a resistor network, or matrix, of these nets is built.
- Once the resistor matrix is formed, then the average current for each transistor connected to the power net is calculated with the power net replaced by a constant voltage source.

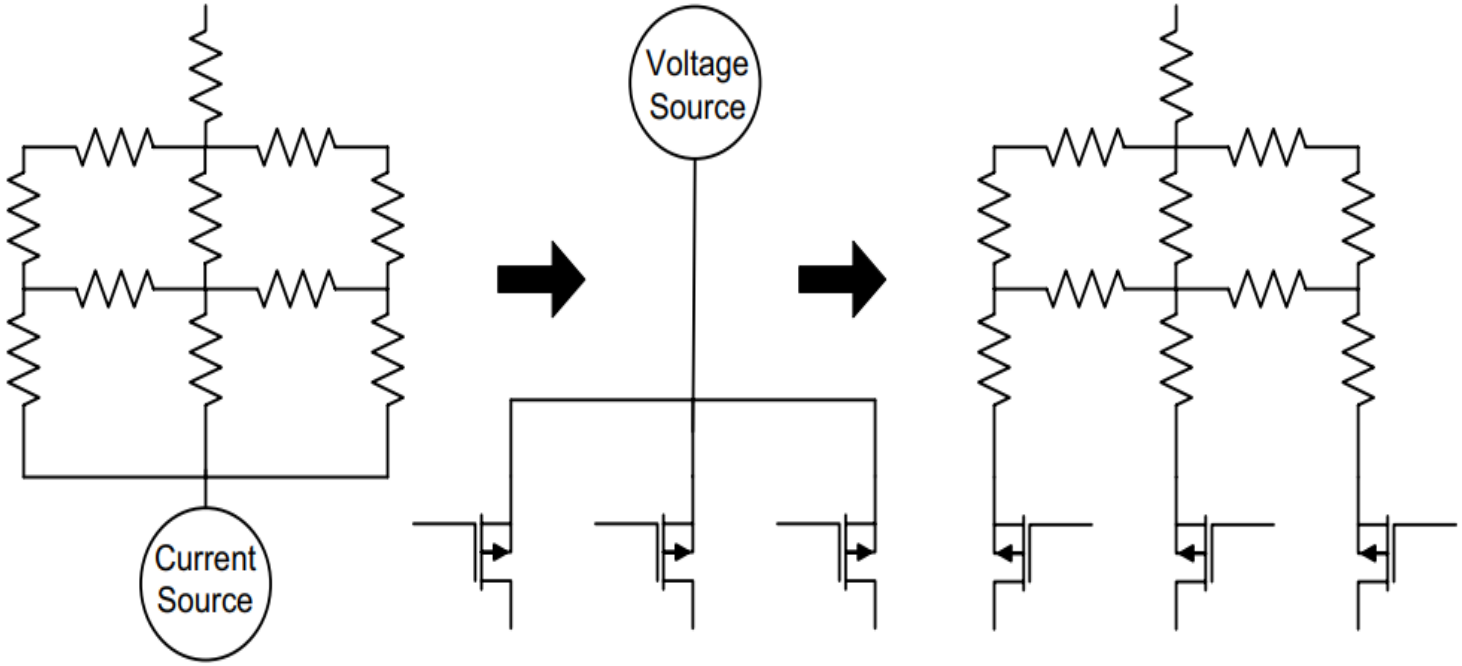


Figure 3-16 Static Power Calculation Method

Placement: Power Analysis

- Next, the average currents are distributed throughout the power net based on the location of each transistor, calculating node voltages and branch current densities using every power source (e.g. power pads).

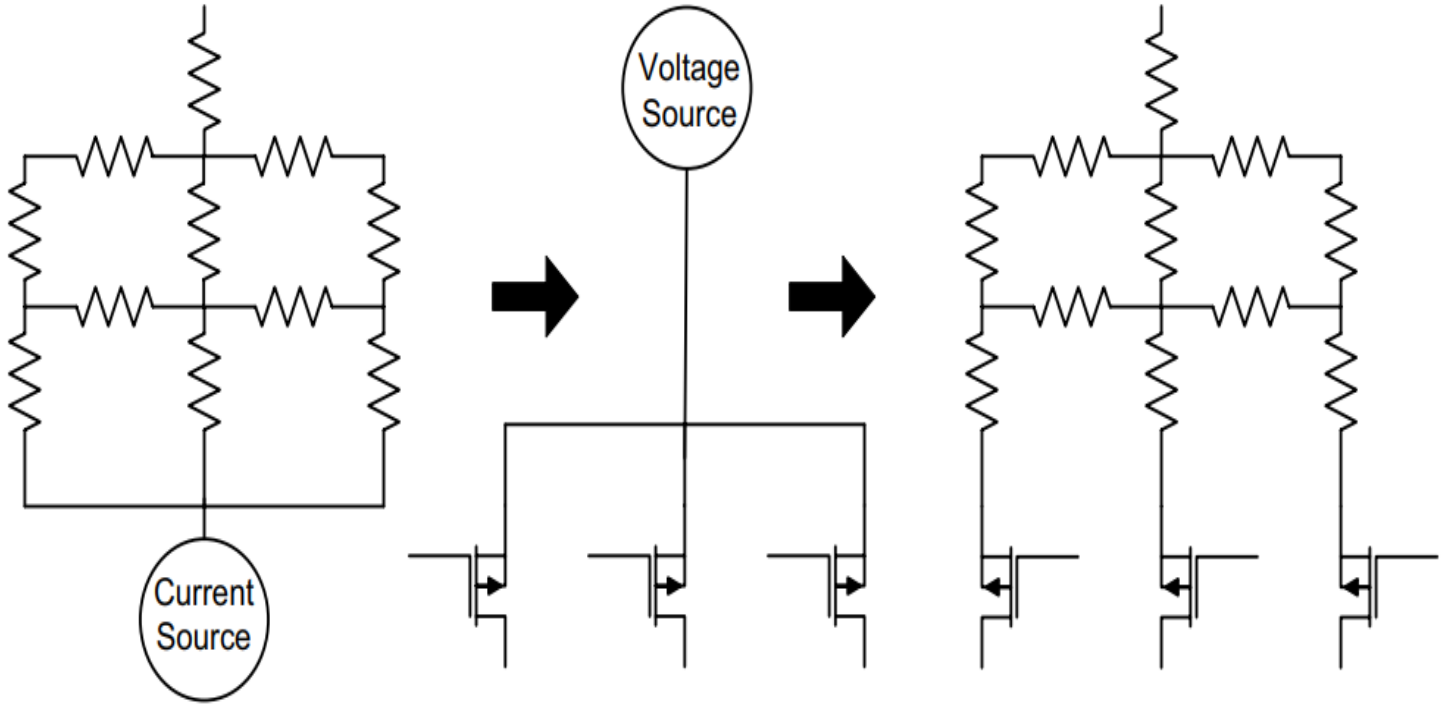
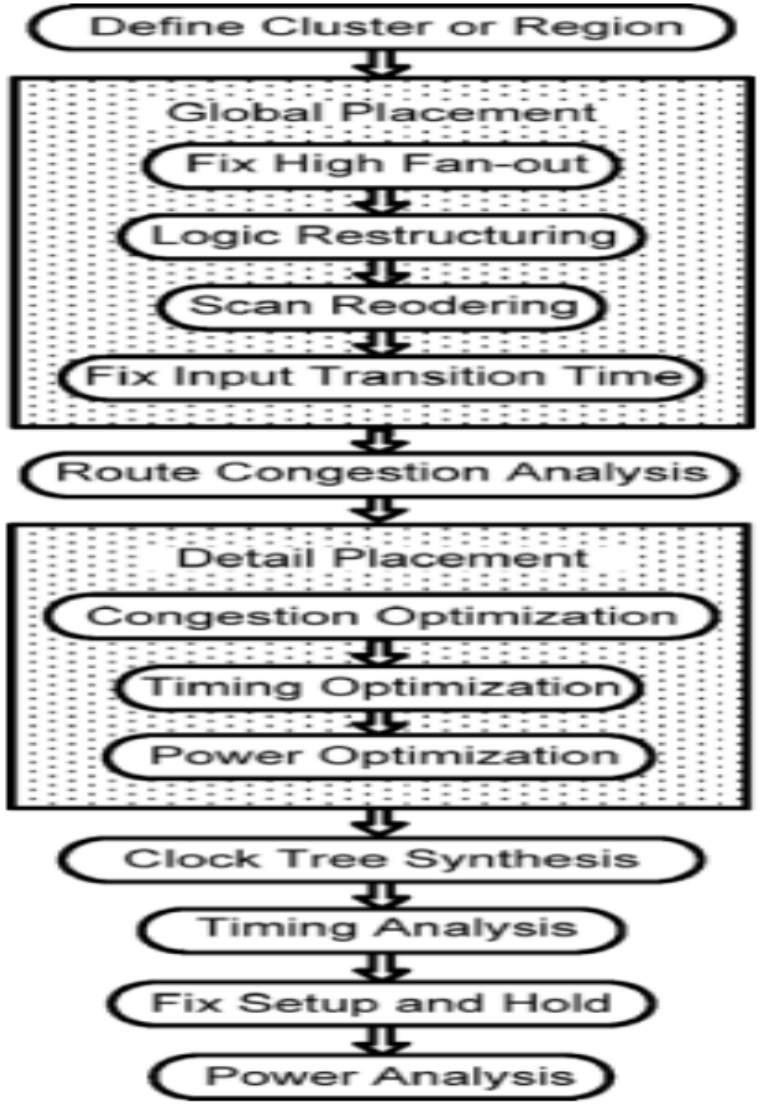


Figure 3-16 Static Power Calculation Method

Standard Cell Placement and Clock Tree Synthesis Steps



References

- “Physical Design Essentials, An ASIC Design Implementation Perspective”. Khosrow Golshan.

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu

Digital System Design

Dr. Sudeendra kumar K

Department of Electronics and Communication Engineering

DIGITAL SYSTEM DESIGN

Routing

Sudeendra kumar K
Department of Electronics and Communication Engineering

Introduction to Routing

- After completion of standard cell placement and power analysis, the next phase is to route the ASIC design and perform extraction of routing and parasitic parameters for the purpose of static timing analysis and simulation.

- As ASIC designs are getting more complex and larger (e.g. sea of cells), routing is becoming more difficult and challenging.
- It is possible for routing to fail to complete, or to take an unacceptable amount of execution run time.

- Besides the routing algorithms, the factors which influence the routability of a given ASIC are the layout of standard cells style, a well-prepared floorplan, and the quality of standard cell placement

Introduction to Routing

- Routing algorithms are mainly classified as channel or over-the-cell based routers. Channel-based routing was used in the early days of ASIC physical design.
- This was because semiconductor factories were not able to process large numbers of metal layers (e.g. two or three).
- Thus with a limited number of routing layers, all connections were restricted to the area between cells or around macro blocks such as memories.
- Fundamentally, channel, or river-based, routers use reserved space between standard cell rows (routing channels) and feed-through (dedicated routing areas inside the standard cell layout) to perform routing between instances as shown in Figure 4-1.

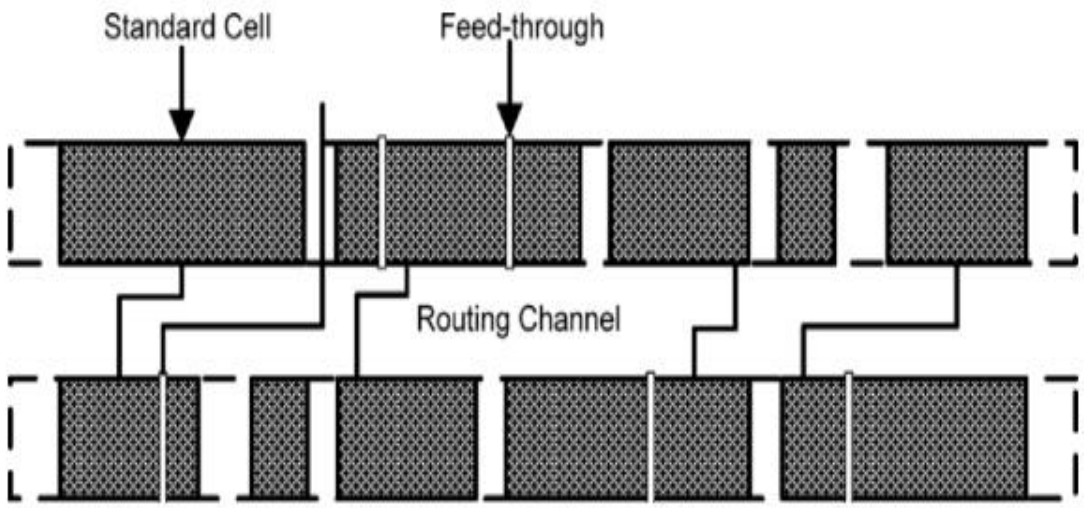


Figure 4-1 Channel Routing Method

Introduction to Routing

- With recent improvements in semiconductor processes and increasing numbers of routing layers, the routing channel and standard cells with feedthrough have been eliminated, and over-the-cell based routing is widely utilized by many physical synthesis and place-and-route engines during the physical design of ASIC devices.
- Due to the inherent complexity of ASIC designs and the very large numbers of interconnections associated with them, the overall routing is performed in three stages: **special routing, global routing, and detail routing.**

Special Routing

- **Special Routing: -** Special routing is used for standard cells, macro power, and ground connections. Most special routers use line-probe algorithms. The line-probe method uses line segments to connect standard cells, macro power, and ground ports to ASIC power and ground supplies.
- Line-probe routers use a generated connectivity list of sources and targets (the generated connectivity list can be port-to-port, port-to-line, or line-to-line) for connection.
- The line segments are used to perform routing according to the connectivity list starting from the target and tracing the line segment until the source is reached.

Special Routing

- These generated line segments do not pass through any obstructions. If they were to pass through an obstruction, they might create an unfixable design rule violation.
- Therefore, one needs to make sure that there are no obstructions where the power and ground ports are located.

- In connecting macro power and ground ports to the main power and ground nets, line-probe routers use the size of the ports to set the width of the power and ground segments.
- This automatic width setting may not be adequate for current density considerations, and one may need to create more power and ground ports to satisfy the current density requirements.

Special Routing

- Excessive current density in an ASIC design can lead to an electromigration problem that reduces the Mean Time To Failure (MTTF) of the device.
- Most ASIC chips must have an MTTF of at least 10 years. Failure due to current density and electromigration of a given wire is expressed by Black's equation:

$$MTTF = \frac{A}{J^2}\exp(\frac{E_a}{kT}), \qquad (4.1.1)$$

where A is the metal constant, J is the current density (i.e. the number of electrons crossing a unit area per unit time), k is the Boltzmann constant, E_a is the activation energy, and T is the temperature.

- Based on Equation 4.1.1, the MTTF due to electromigration depends on two routing, one must focus on current density as a major parameter to address electromigration problems.
- These generated line segments do not pass through any obstructions. If they rule violation. Therefore, one needs to make sure that there are no obstructions where the power and ground ports are located. parameters–temperature and current density. During power and ground routing, one must focus on current density as a major parameter to address electromigration problems.

Global Routing

- Global routing is the decomposition of ASIC design interconnections into net segments and the assignment of these net segments to regions without specifying their actual layouts.
- Thus, the first step of the global routing algorithm is to define routing regions or cells (i.e. a rectangular area with terminals on all sides) and calculate their corresponding routing density.
- These routing regions are commonly known as Global Routing Cells, or GRC, as shown in Figure 4-2.

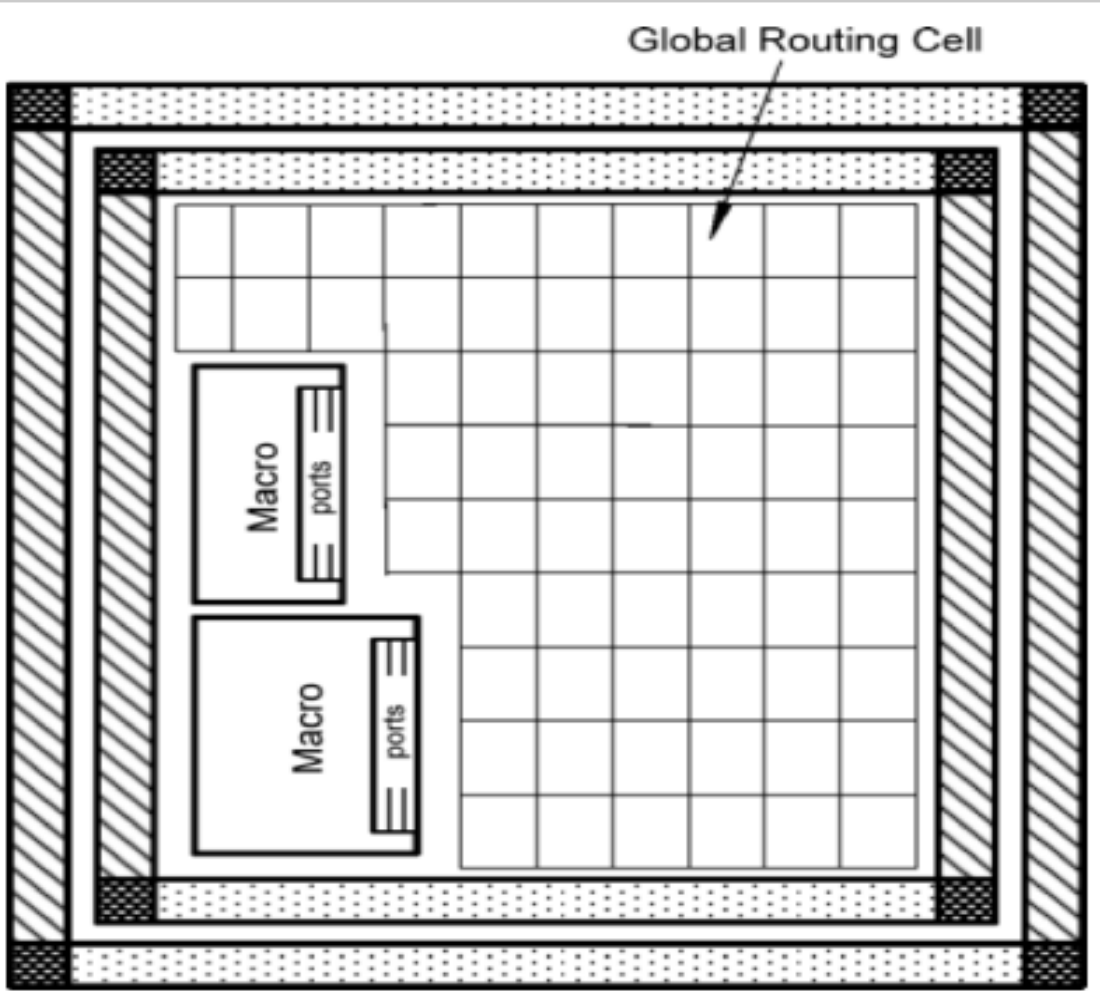


Figure 4-2 Global Routing Cells

Global Routing

- The density, or capacity, of these cells is defined as the maximum number of nets crossing the routing regions and is a function of the number of routing layers, cell height in vertical or horizontal direction, minimum width, and spacing of wire as defined in the technology file and is given by

$$C_d = \frac{nh}{w+s}, \qquad (4.2.1)$$

where C_d is the global routing cell density, n is the number of routing layers that are available for horizontal or vertical direction, h is the height of the global routing cell length, and w and s correspond to minimum wire width and spacing for vertical or horizontal directions.

Global Routing

- Global routing uses a graph to model the interconnection networks. The vertices of the graph denote the standard cell ports. The edges of the graph correspond to connections between two ports within routing cells and among routing cells themselves.
- The graph is constructed by means of region assignment (i.e. the order that a region will be routed first) and assignment of each net to a pin on the boundary of the routing region.

Global Routing

- After global routing is performed, the pin locations will be determined such that the connectivity among all standard cells in the ASIC core area is minimal.
- Almost all global routers report the design routability statistic using overflow or underflow for Global Routing Cells (GRC), which is the ratio of routing cells' capacity and the number of nets that are required to route a given routing cell for all vertical and horizontal routing layers.
- The GRC statistic is a very good indication of wiring congestion and shows the number of nets needed to route a region versus the available number of routing layers.
- Global routing algorithms generate a non-restricted route (i.e. not a detail route) for each net in the design and use some method of estimation to compute wire lengths and extract their corresponding parasitics.

Global Routing

- Depending on the global routing algorithm, there are several ways to estimate the associated wire length for each net in the design. The most common wire estimation methods are:
  - Complete-graph
  - Source-to-sink
  - Steiner minimal tree
  - Minimum spanning tree
  - Half-perimeter
  - Minimum chain

Complete-graph

Complete-graph has connection from each port to every other port. In this method, all the interconnect lengths of the complete-graph connection are added together and then divided by $n/2$, where n is the number of ports. In a graph with n nodes, $(n-1)$ connections will emanate from each node to connect to other $(n-1)$ nodes in a complete-graph connection. Thus, a complete-graph has a total of $n(n-1)$ interconnections that include duplicate connections. Therefore, the total connectivity that requires forming a complete-graph connection is $n(n-1)/2$. Realizing that only $(n-1)$ connections need to connect n nodes, then to estimate reasonable wire length, the total net length of a complete-graph is divided by $n/2$.

- Figure 4-3 shows an example of complete-graph wire length estimation based on the number of horizontal and vertical grids from a given source to multiple sink points indicated by dark and clear dots.

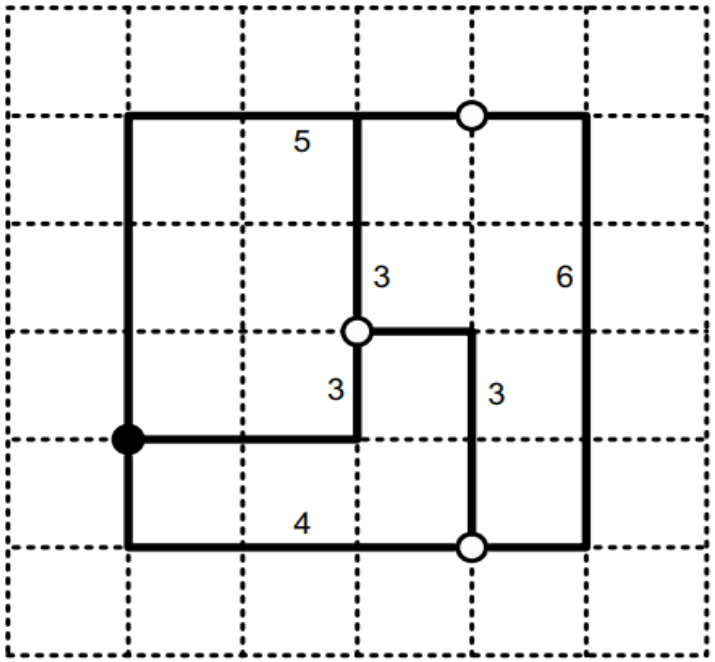


Figure 4-3 Complete Graph Wire Length Estimation

Complete-graph

- Source-to-sink global routing algorithm connects one port to all other ports for every net in the design. In other words, one wire from a source port to other ports is connected.
- It is important to note that for the ports that are located far from each other, this algorithm does not produce an accurate wire estimate.

Global Routing

- It is important to recognize that global routing is a key step in any place-and route and physical synthesis tool.
- Global routing is used during placement and clock tree synthesis to estimate the wire lengths and RC delay time constants.
- Correlation between actual interconnect lengths after detail routing and approximation provided by any global route algorithm should be key for determining the appropriate place-and-route and/or physical synthesis tool to be used.

Digital System Design
Detail Routing

- The objective of detail routing is to follow the global routing and perform the actual physical interconnections of ASIC design. Therefore, the detail router places the actual wire segments within the region defined by the global router to complete the required connections between the ports.
- Detail routers use both horizontal and vertical routing grids for actual routing. The horizontal and vertical routing grids are defined in the technology file for all layers that are being used. **The detail router can be grid based, grid-less-based, or sub-grid-based.**

Detail Routing

- Grid-based routing imposes a routing grid (evenly spaced routing tracks running both vertically and horizontally across the design area) that all routing segments must follow.
- In addition, the router is allowed to change direction at the intersection of vertical and horizontal tracks as indicated in Figure 4-10.
- The advantage of grid-based routing is efficiency. When using a grid-based router, one needs to make sure that the ports of all instances are on the grid.
- Otherwise, they can create physical design rule errors and will be difficult to resolve with the router.

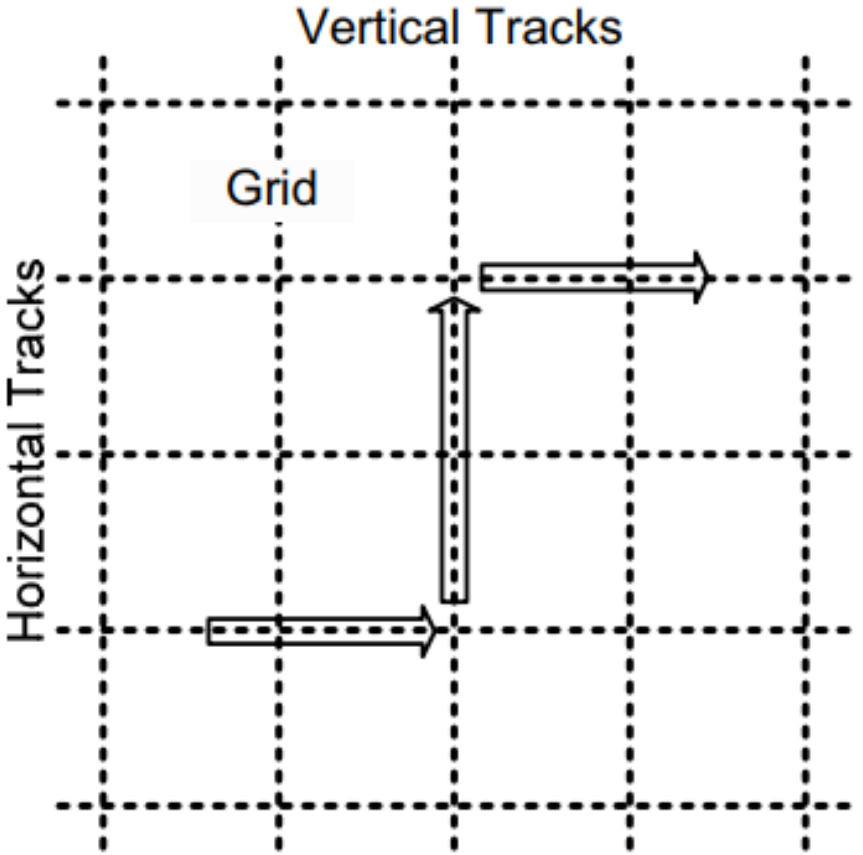


Figure 4-10 Grid-based Routes

Detail Routing

- Grid-less-based (or shape-based) routers do not follow the routing grid explicitly, but are dependent on the entire routing area and are not limited by grid's restrictions.
- They can use different wire widths and spacing without routing grid requirements. The most fundamental problem with this type of router is that they are very slow and can be very complicated.
- The sub-grid-based router brings together the efficiency of grid-based routers with the flexibility (of varying the wire width and spacing) of the grid-less based routers.
- The sub-grid-based router follows the normal grid similar to the grid-based router. However, a sub-grid-based router considers these grids only as guidelines for routing and is not required to use them, as illustrated in Figure 4-11.

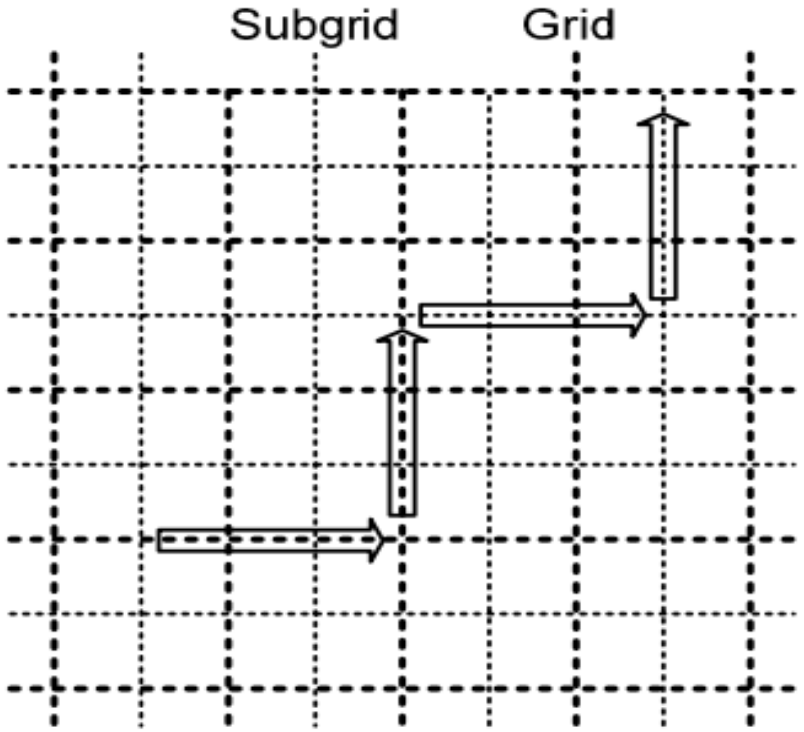


Figure 4-11 Subgrid-based Routes

Detail Routing: Maze routing

- In searching for the shortest path from source to target connection, the Maze algorithm performs a breadth-first search and labels each track crossing point with its distance from the source (similar to wave propagation).
- This expansion phase will eventually reach the target node if a connection is possible.
- Once the expansion phase is completed, the trace-back phase begins with the connection from target to source by following the path with decreasing labels until the original source is reached.
- This algorithm is guaranteed to find the shortest path between source and target for a given connection.
- Figure 4-12 illustrates the Maze routing (expansion and track back) algorithm.

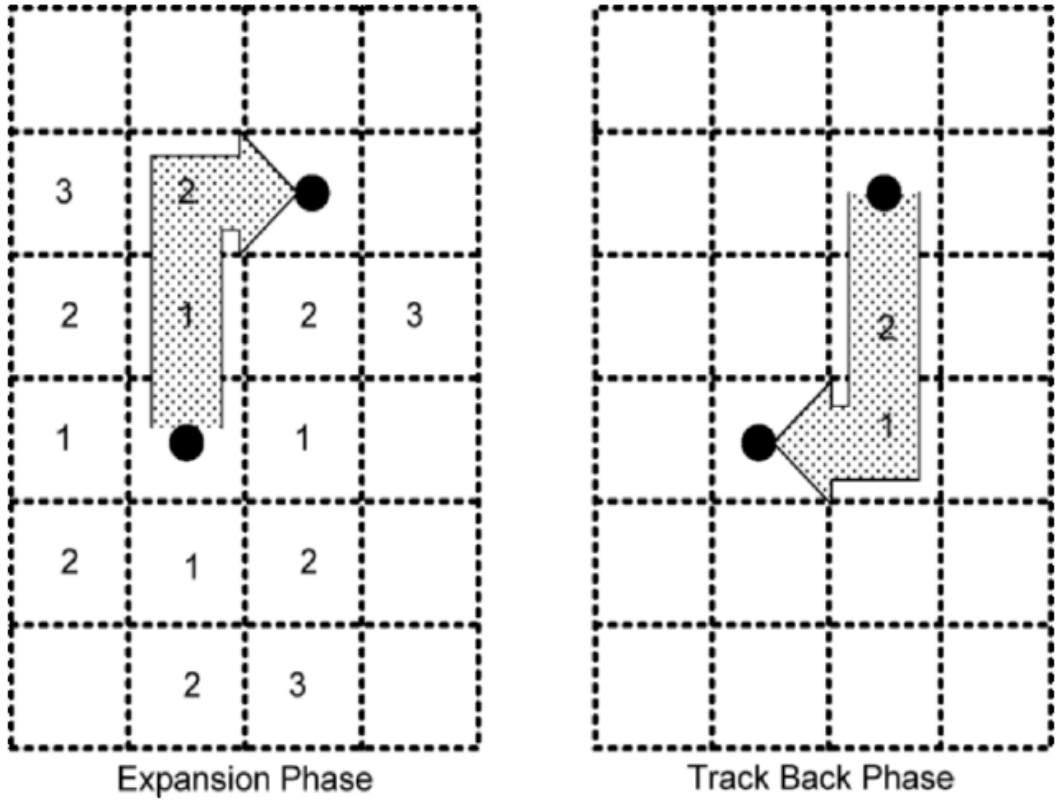


Figure 4-12 Maze Router Algorithm Illustration

Detail Routing

- This procedure of detail routing is very similar to global routing. The only difference is that during detail routing, physical wire segments will be used for connection rather than connectivity projections.
- Thus, it is important to have strong correlation between the detail and global routers with regard to the wire length approximation and actual wire connection.
- The reason for the close correlation between global and detail routers is that one can determine whether the ASIC design timing meets actual timing requirements by estimating the wire resistance and capacitance early on in the physical design cycle.

Detail Routing

- During detail or final routing of any ASIC physical design, total routing area is divided into regions or sub-areas known as switchboxes; these switchboxes are numbered in sequence for subsequent routing by the detail router.
- These switchboxes have terminals on all four sides (i.e. 2D) or may have connection points inside (i.e. 3D) as shown in Figure 4-13.
- If the detail router uses a 3D-switchbox, then this type of routing is commonly referred to as area routing style.

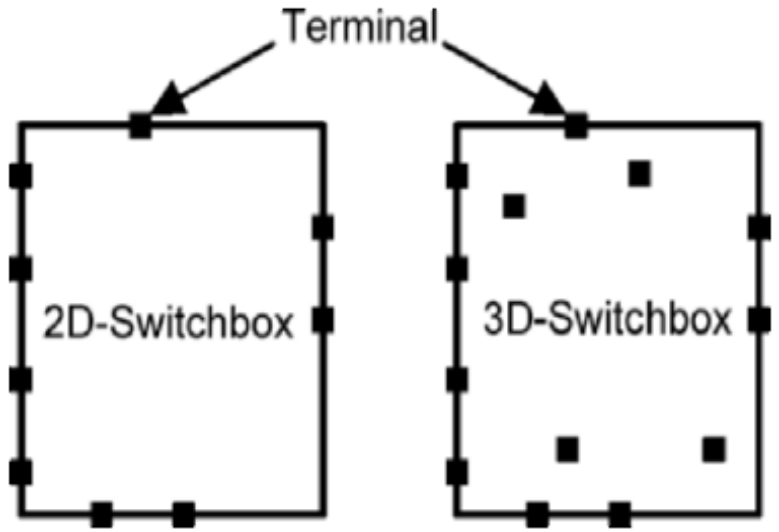


Figure 4-13 Detail Routing Switchbox

Detail Routing

- Once the switchbox construction is completed, then the detail router starts the track assignments. During track assignment, all routing paths for connection among the entire instance that are placed will be routing layer.
- Depending on the standard cell design, the track assignment algorithm will use HVH or VHV models to assign tracks.
- Currently, most of the standard cell designs are optimized for HVH (i.e. metal one is routed horizontally; metal two, vertically and so on).
- If there are more than six metal layers available for routing, then the VHV model (i.e. metal one is routed vertically; metal two, horizontally and so on) provides much better results with respect to area.
- This routing area reduction is due to the fact that the VHV model has a tendency to produce shorter wire lengths and fewer via insertions.

Detail Routing

- After completion of track assignment, the detail router starts performing the actual routing of the first switchbox and, in sequence order, tries to complete the routing of all switchboxes.
- The objective of the detail router during this stage of route is to complete the entire routing area with no open connections between terminals even if connections would cause design rule violations or shorts.
- The next phase of detail routing is known as search-and-repair. During this phase, the detail router begins with resolving all types of physical design rule violations such as metal spacing, fill notches, and wire shorts.

Detail Routing

- After completion of route and search-and-repair, via minimization and optimization is recommended–especially for deep submicron physical designs.
- Because the impact of large numbers of vias on any ASIC fabrication yield loss, most place-and-route tools offer via minimization and optimization options.
- Via minimization refers to the process of removing as many vias as possible by reducing the number of jogs associated with a wire connection.
- This via minimization not only reduces the wire resistance (i.e. fewer vias), but also provides better yield with respect to via processing (e.g. one of the major reasons yield is lost in deep submicron).

Detail Routing

- Via minimization refers to the process of removing as many vias as possible by reducing the number of jogs associated with a wire connection.
- This via minimization not only reduces the wire resistance (i.e. fewer vias), but also provides better yield with respect to via processing (e.g. one of the major reasons yield is lost in deep submicron).
- Via optimization or redundancy increases the number of isolated single vias as much as possible as long as there is no impact on the routing area.
- The benefit of this option is two-fold—one is that it reduces the resistance of the overall via within a path (i.e. two vias in parallel) and two, it provides via yield enhancement by means of redundancy from a statistical point of view. In addition, it provides for better electromigration immunity.

Detail Routing

- The antenna effect is a common name for the effect of charge accumulation in metal segments that are connected to an isolated transistor gate during the metallization process.
- This effect is also known as plasma-induced or process-induced damage. Plasma-induced damage has become a major concern for ASIC reliability.
- Most place-and-route tools that are currently used are capable of checking the topology of overall ASIC wiring for antenna ratio rule violations.

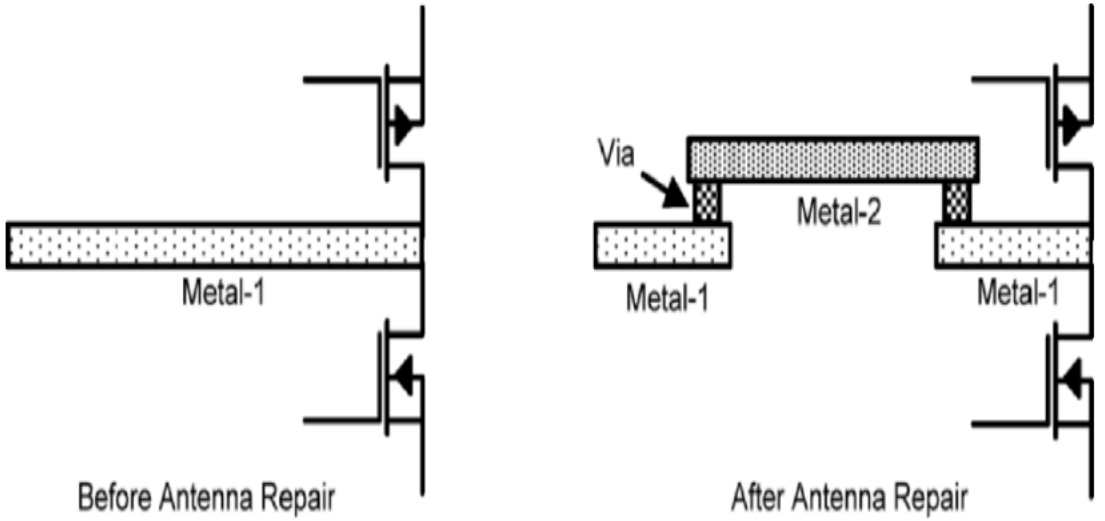


Figure 4-14 Antenna Repair Procedure

Detail Routing

- These antenna ratio rules are coded in the place-and-route tools technology file for the transistor gate area connected to the standard cell ports, as well as to any transistors connected to the macro port that is being used.
- The most common technique used to resolve antenna problems is to reduce the peripheral metal length that is attached to the transistor gates.
- This is accomplished by segmenting the wire of one type into several segments of different metal types, and connecting these various types of metals through via connections as shown in Figure 4-14.

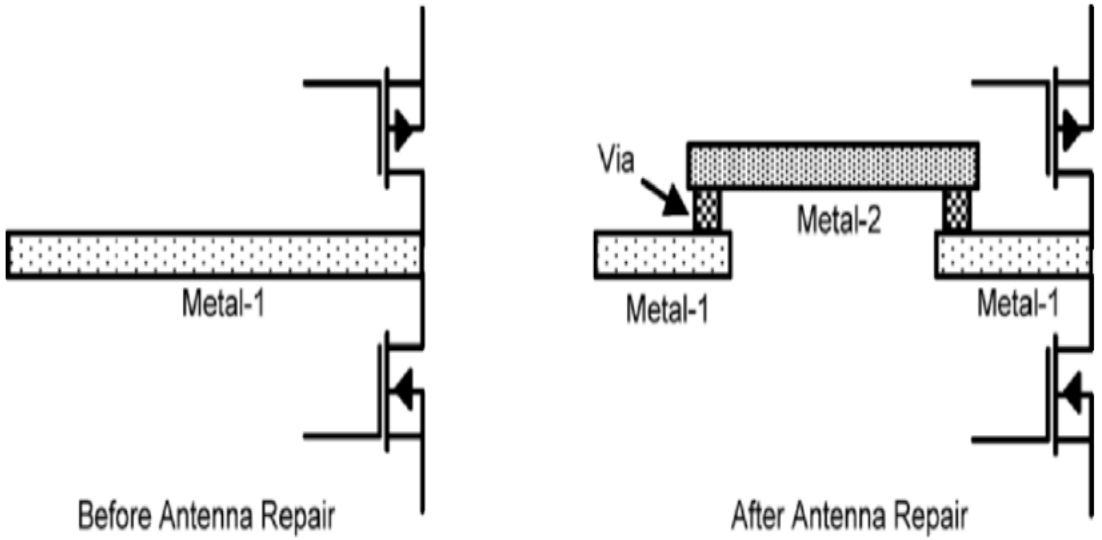


Figure 4-14 Antenna Repair Procedure

Extraction

- Parasitic extraction is the calculation of all routed net capacitances and resistances for the purpose of delay calculation, static timing analysis, circuit simulation, and signal integrity analysis.
- Parasitic extraction is performed by analyzing each net in the design and taking into account the effects (such as dielectric stack) of the net's own topology and proximity to other nets. Currently, most physical design tools are using three-dimensional (3D) models to extract capacitance associated with each net.

Extraction

- To extract wire capacitance and resistance, the semiconductor foundries provide the capacitance coefficient and sheet resistance values for various drawn layers in their electrical documentation.
- These electrical parameters (capacitance coefficient and sheet resistance) are based on measurements performed using test-keys from actual silicon data for best, nominal, and worst process conditions or corners.
- The change in the value of capacitance coefficient and sheet resistance at each process corner is due to variations in dimension of the ASIC silicon features as well as to the effects of voltage and temperature.

Extraction

The wire resistance extraction is based on the resistance of a uniform conducting material and can be expressed as

$$R = \rho(l/s), \qquad (4.4.1)$$

where ρ is the resistivity, l is the length, and s is the cross-section of the material. Cross-section s is the product of height h and width w of the material. Thus, Equation 4.4.1 can be rewritten as

$$R = (\rho/h)(l/w), \qquad (4.4.2)$$

where (ρ/h) refers to the sheet resistance of the material in Ohm per square. Obtaining the resistance of a wire segment using Equation 4.4.2 is done by multiplying the material sheet resistance by the ratio of the segment length and width. To extract the correct value of the segment resistance, the effect of process and temperature must be included.

Extraction

- There are two process-induced topology variations associated with segment resistance—variation in thickness, or height, and variation in the width.
- Variation in thickness (Δh) results in a change of sheet resistance and variation in the width (Δw) results in a change of calculated resistance as illustrated in Figure 4-15.
- Variation in is a result of metal layer patterning during the etching process. Since etching will cause a change in width of wire segment, and therefore its spacing to a neighboring wire segment, one should be able to apply this etching coefficient to the extraction tools for each metal layer.

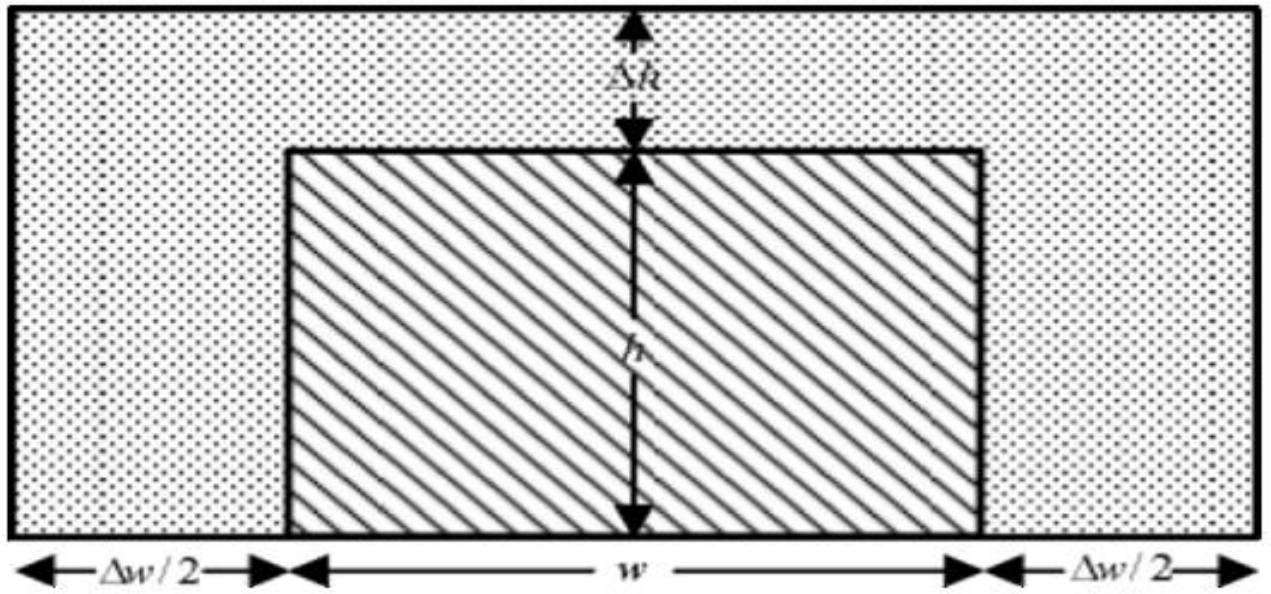


Figure 4-15 Wire Segment Cross-section

Extraction

- Change in the value of is mainly due to metal layer and interlayer dielectric planarization for eliminating irregular and discontinuous conditions between successive metal layers.
- Several techniques are used for planarization such as spin-on glass, deposit and etch-back, and Chemical Mechanical Polishing (CMP).
- The performance of CMP is greatly determined by polishing parameters such as down-pressure, rotation and speed of pads, and types of abrasives used.
- One of the main problems encountered during the CMP process (which is caused by the parameter setting such as rotating speed, and pressure of polishing pad) is known as the dishing effect.
- This effect impacts the interconnect sheet resistance and becomes especially important in deep submicron processes such as 130nm and below. The dishing effect is considered to be a localized problem and is determined by the CMP window size.

Extraction

- Figure 4-16 shows the result of CMP dishing effects as well as the profile of corresponding changes in the sheet resistance.
- Therefore, it is imperative that extraction tools be able to account for changes of uniform values in Δh due to dishing effect during resistance extraction.
- Another consideration during wire segment extraction is the effect of temperature on the resistance value.
- Since the resistance of a metal segment is dependent on the collision process within the segment, the resistance value is expected to increase with temperature due to the increase in electron collisions with metal atoms.
- This temperature dependency of resistivity is characterized by a fractional change in resistance which is proportional to the temperature change and is expressed as

$$\frac{\Delta R}{\Delta T} = \beta R_0, \qquad (4.4.3)$$

where $\Delta T = (T - T_0)$ is the change in temperature T from its initial temperature T_0, $\Delta R = (R - R_0)$ corresponds to the change of resistance R from its initial resistance R_0, and proportionality constant β refers to Temperature Coefficient of Resistance (TCR).

Extraction

- In the deep submicron process, and for very high-performance ASICs, assumption of uniform temperature across the chip for resistance calculation may not be adequate and non-uniformity of the temperature effects on resistance need to be considered.
- Figure 4-17 shows three regions with different temperature profiles: T1, T2, and T3.
- The non-uniformity of temperature gradient across the chip substrate (e.g. T1, T2, and T3) is often due to the difference of operating frequency of functional blocks and their actual placement within the ASIC device.
- This different switching activity of different areas of the die and their corresponding non-uniform temperature gradients will create hot-area regions in the substrate. Interconnects across the hot-area will generate different resistances.

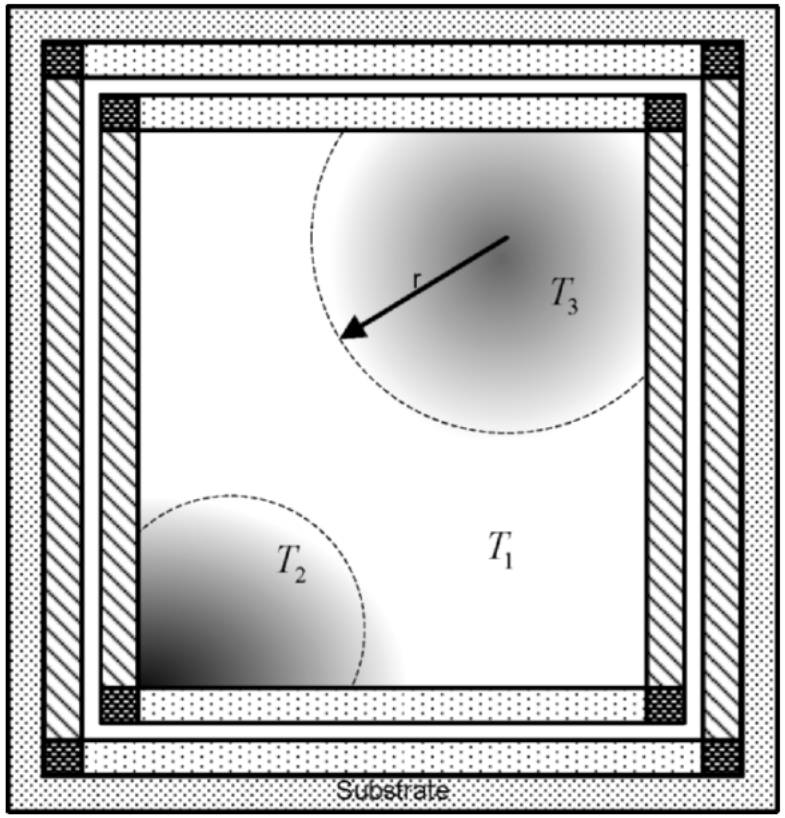


Figure 4-17 Non-uniform Substrate Temperature Profile

Extraction

- The next consideration is gate capacitance and routing segment capacitance extraction. Almost all extraction tools today use the 3D capacitance extraction method to extract routing segment capacitance.
- The gate capacitance is dictated by the topology of the transistor. Each transistor gate capacitance is actually the sum of capacitances between gate to MOSFET body, gate to source, and gate to drain. Gate to MOSFET body (or substrate) capacitance with gate area of A defined as:

$$C_{gs} = \left(\frac{\varepsilon_{ox} A}{T_{ox}}\right) = \left(\frac{\varepsilon_{ox} WL}{T_{ox}}\right), \qquad (4.4.10)$$

where ε_{ox} is the dielectric constant of the gate oxide, T_{ox} is the gate oxide thickness, L is the transistor drawn gate length, and W is the transistor drawn gate width. The term $\varepsilon_{ox} / T_{ox}$ expresses the oxide capacitance.

Extraction

In actuality, both source and drain of gate extend below the oxide by amount L_d. This is known as lateral diffusion. Therefore, the affected transistor gate length becomes shorter than the drawn length by a factor of $2L_d$ as shown in Figure 4-18.

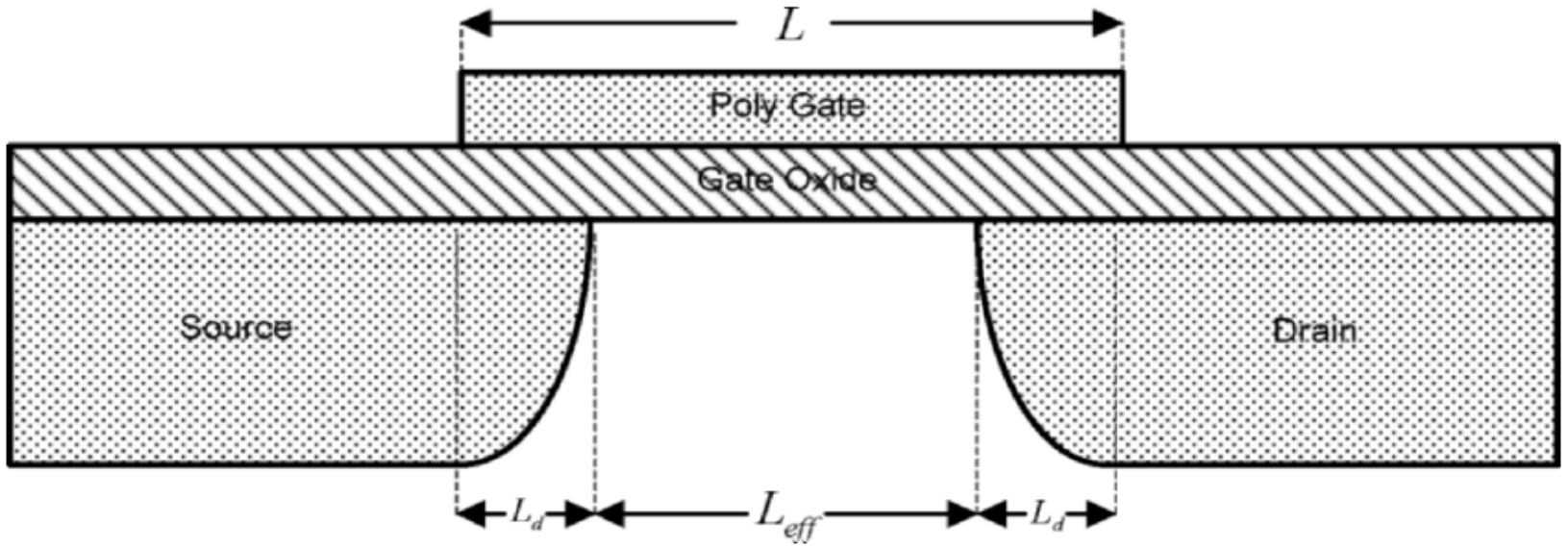


Figure 4-18 Transistor Cross-section

Extraction

Rewriting Equation 4.4.10 to include the gate length effect results in

$$C_{gs} = \frac{\varepsilon_{ox} W (L - 2L_d)}{T_{ox}}. \qquad (4.4.11)$$

The reduction in the transistor gate length by the lateral diffusion effect gives rise to gate-to-source and gate-to-drain capacitance given by

$$C_s, C_d = \frac{\varepsilon_{ox} W L_d}{T_{ox}}, \qquad (4.4.12)$$

where C_s and C_d are gate-to-source and gate-to-drain capacitance.
The routing segment capacitance is extracted after final routing. During the routing segments' capacitance extraction, the following types of capacitance are considered:

- Area capacitance
- Fringe capacitance
- Sidewall capacitance

Extraction

Area capacitance is routing segment capacitance to substrate and to other exposed routing segments above and below. The value of area capacitance is given by

$$C_p = \varepsilon \frac{LW}{d}, \qquad (4.4.13)$$

where C_p is the area or plate capacitance, ε is the permittivity of the dielectric, L and W are the overlapping length and width of the routing segment, d is the distance between the routing segments and corresponds to interlayer dielectric, or ILD, thickness, and the term $(\varepsilon W / d)$ refers to capacitance per unit length.

Extraction

- Figure 4-19 illustrates the area capacitance for metal and polysilicon layers.
- The values of interlayer dielectric thickness and the corresponding capacitance per unit area (ε/d) for various layers are defined by semiconductor foundries for different process corners.

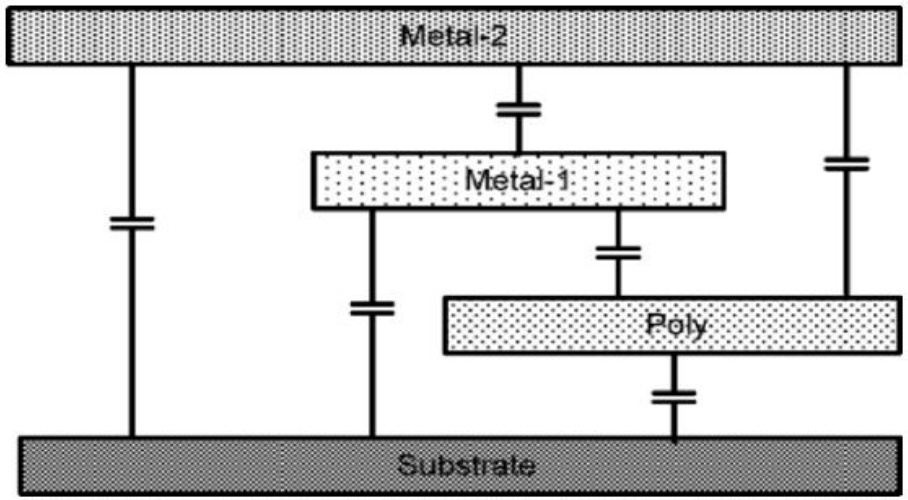


Figure 4-19 Area Capacitance of Two Metal Route

A simple illustration of such an electrostatic field between two conducting plates is shown in Figure 4-20.

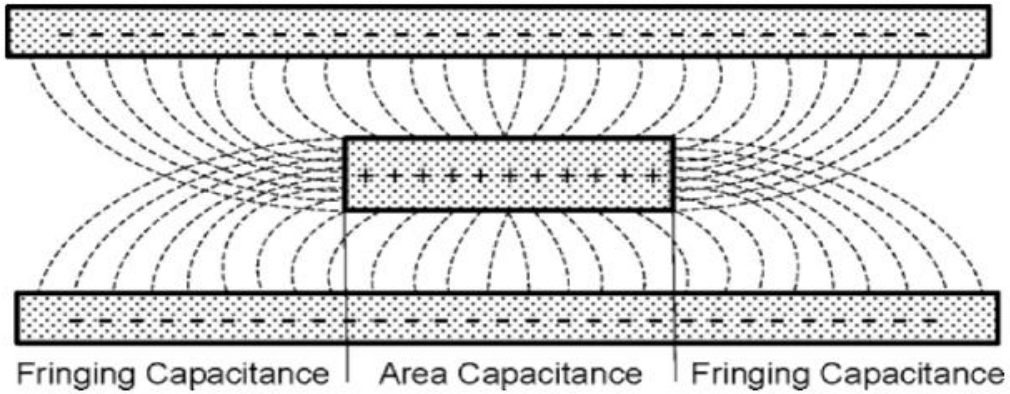


Figure 4-20 Electrical Field Flux Between Three Conducting Layers

Extraction

There are two frequently used methods for fringing capacitance extraction. One is a simple analytic approximation that has a direct physical interpretation [3]. Using this method, the value of fringing capacitance is given by

$$C_f = \frac{2\pi\varepsilon L}{\ln\{1+\frac{2d}{h}+[\frac{2d}{h}(\frac{2d}{h}+2)]^{1/2}\}} \quad \text{for } W \geq \frac{d}{2}. \quad (4.4.14)$$

Second is to use an empirical expression. In this method, several numerical solutions are evaluated and defined. A purely empirical expression allows for all sidewall effects [4] as well as the fringing capacitance of a routing segment, and is given by:

$$C_f = \varepsilon L[0.77 + 1.06(\frac{W}{d})^{0.25} + 1.06(\frac{h}{d})^{0.5}]. \quad (4.4.15)$$

In Equation 4.4.14 and 4.4.15, C_f is the fringing capacitance, ε is the permittivity of the dielectric, L is the length of the routing layer, h is the layer thickness, d is the dielectric thickness, and W is the width of the routing layer.

Extraction

Sidewall capacitance is due to the coupling between the sidewalls of routing segments on the same layer and is similar to the area capacitance.
Figure 4-22 illustrates sidewall capacitance associated with different layers.

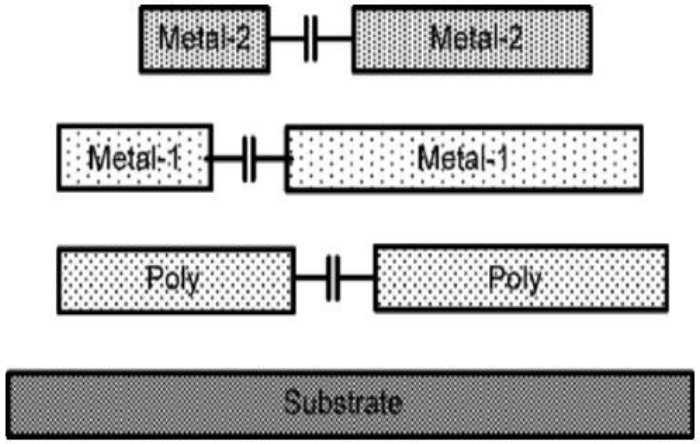


Figure 4-22 Sidewall Capacitance of Two Metal Route

The value of sidewall capacitance is the function of distance separating the routing segments and the length that they extend in parallel. It is given by:

$$C_s = \varepsilon \frac{Lh}{s}, \qquad (4.4.16)$$

Extraction

In the final interconnect capacitance calculation, the sum of all transistor gate capacitances associated with such interconnect must be added to the interconnect capacitance.

Including all contributions for a given interconnect, the total interconnect load capacitance is given by

$$C_{total} = \sum_{i=1}^{n} (C_p + C_f + C_s)_i + \sum_{j=1}^{k} (C_g)_j , \qquad (4.4.17)$$

where C_{total} is the interconnect total capacitance, C_p is the area capacitance, C_f is the fringing capacitance, C_s is the sidewall capacitance of each segment of the interconnect, and C_g is the transistor's input gate and junction capacitance connected to that interconnect.

Extraction

- For the purpose of timing analysis, the extracted value of resistance and capacitance for each interconnect in the design is exported from the place and-route tools and then imported to the timing analysis engine.
  - There are several approximation models used to represent interconnection parasitics. These approximations are:
  - Lumped capacitance model
  - Lumped resistance and capacitance (RC) model
  - Distributed resistance and capacitance (π) model
  - Distributed resistance, capacitance, and inductance (RCL) model

Extraction

- Lumped capacitance is a single order approximation and considers only the total capacitance value of interconnection while ignoring the resistance value.
- This was used in the early days of process technology as wire delay contributions were negligible. Thus, this model was adequate to calculate the propagation delay through the gate as shown in Figure 4-25.
- Due to the simplicity of this model, some layout synthesis tools still use it to estimate the capacitive loading effects during initial placement and logical restructuring.
- This model can provide reasonably accurate results as long as the wire resistance is much smaller than the output driver resistance.
- However, as the driving gate output resistance decreases and wire resistance increases due to feature size scale-down, this first order approximation will no longer be valid.

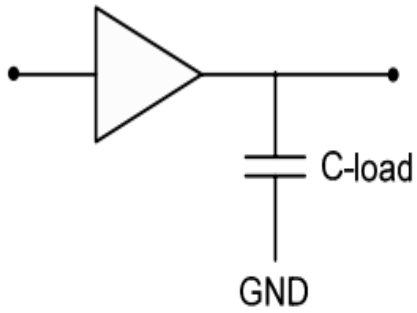


Figure 4-25 Lumped Capacitance Model

Extraction

- Lumped resistance and capacitance (or RC model) is considered to be a second-order approximation and takes into account the effect of loading capacitance as well as the total wire resistance of interconnections as shown in Figure 4-26.
- In the second order approximation, the sum of wire capacitances is used to compute the cell delay based on the characterized model from the library and the product of lumped interconnect capacitance and resistance is used to compute the output slope.
- Hence, the resistance effects of the loading interconnects are taken into account in a decoupled manner. Using lumped RC to calculate interconnection delay is considered pessimistic and inaccurate for long interconnects.

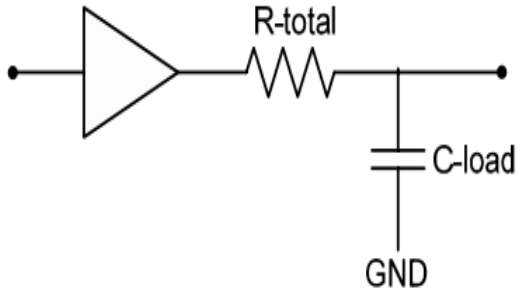


Figure 4-26 Lumped Resistance and Capacitance Model

Extraction

- Distributed resistance and capacitance (or so-called π model) is classified as a third order interconnect delay approximation.
- In this model, interconnections are segmented into a series of resistor and capacitor network resembling the a transmission line.
- Figure 4.27 illustrates a simple distributed resistance and capacitance network.

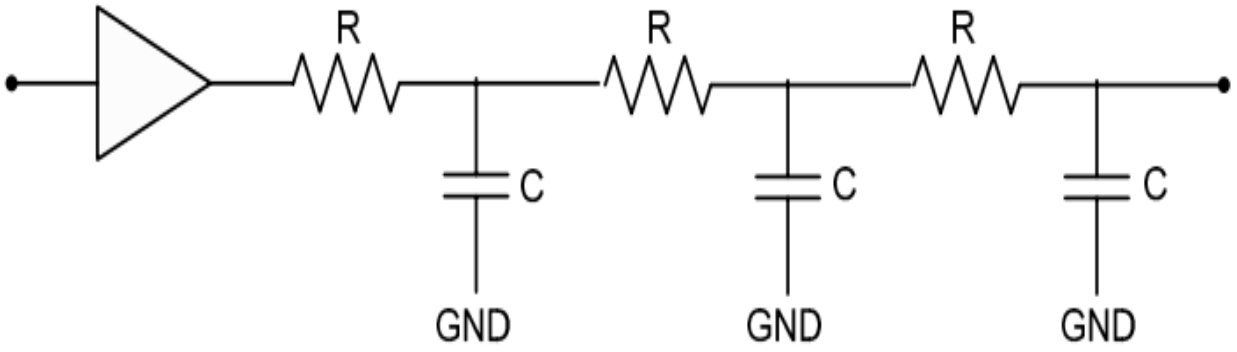


Figure 4-27 Distributed Resistance and Capacitance Model

Extraction

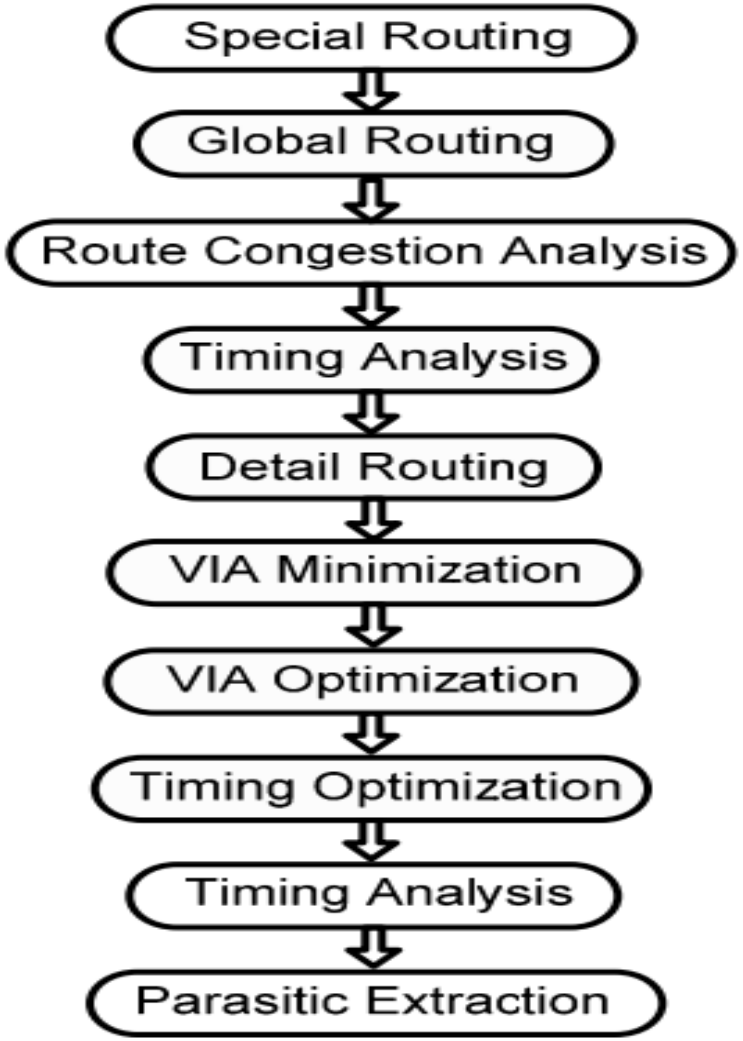
- The method of distributed resistance and capacitance is very effective for computing wire delays of very highly resistive interconnection networks because as wire resistance increases it has a tendency to shield the actual wire capacitance in the presence of driving gate resistance.
- Today's physical design tools are capable of generating lumped capacitance, lumped RC, and distributed RC styles for wire delay calculation.
- Depending upon the style used, the extraction run time could increase. Lumped capacitance extraction has the shortest run time, whereas the distributed RC has the longest run time.

Extraction

- Choice of extraction style is dependent upon the application. For example, if one needs to perform dynamic power analysis, then lumped capacitance would be sufficient.
- For timing analysis or signal integrity where the effects are due to the resistance value and capacitance values, such as coupling effects to neighboring nets, the distributed RC extraction is more appropriate.
- One of the most commonly used formats used to import and export distributed RC parasitic capacitance and resistance values extracted per net based on their actual geometry and layer width and spacing information, is the Standard Parasitic Extended Format (SPEF).

Digital System Design

Routing and Parasitic Extraction Steps



References

- “Physical Design Essentials, An ASIC Design Implementation Perspective”. Khosrow Golshan.

THANK YOU

Sudeendra kumar K

Department of Electronics and Communication Engineering

sudeendrakumark@pes.edu