Advanced VLSI Design Lab, IIT Kharagpur

Digital Design Group

Logic Synthesis with Verilog HDL

Gourav Sarkar
Advanced VLSI Design Lab, IIT KGP

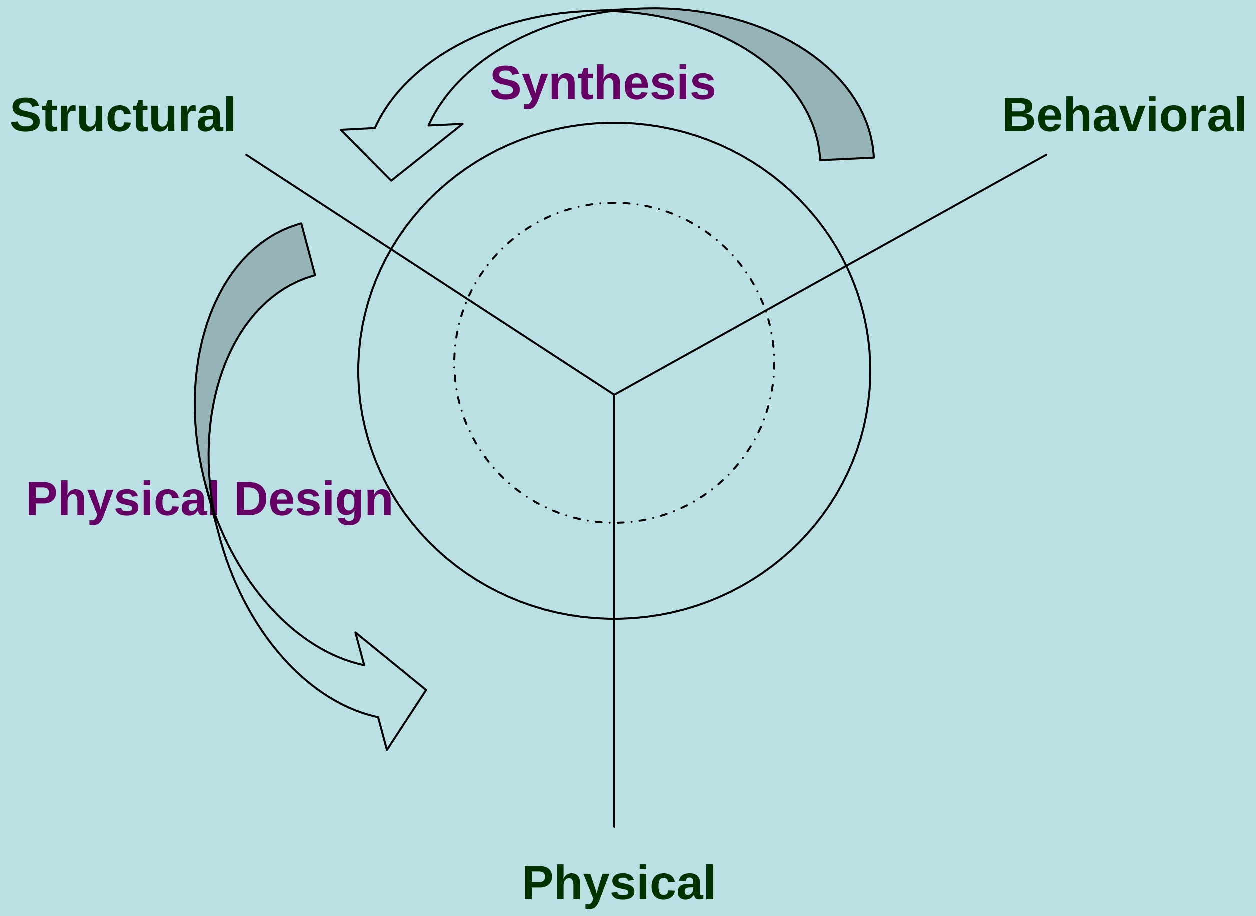
Contents:

What is Logic Synthesis?

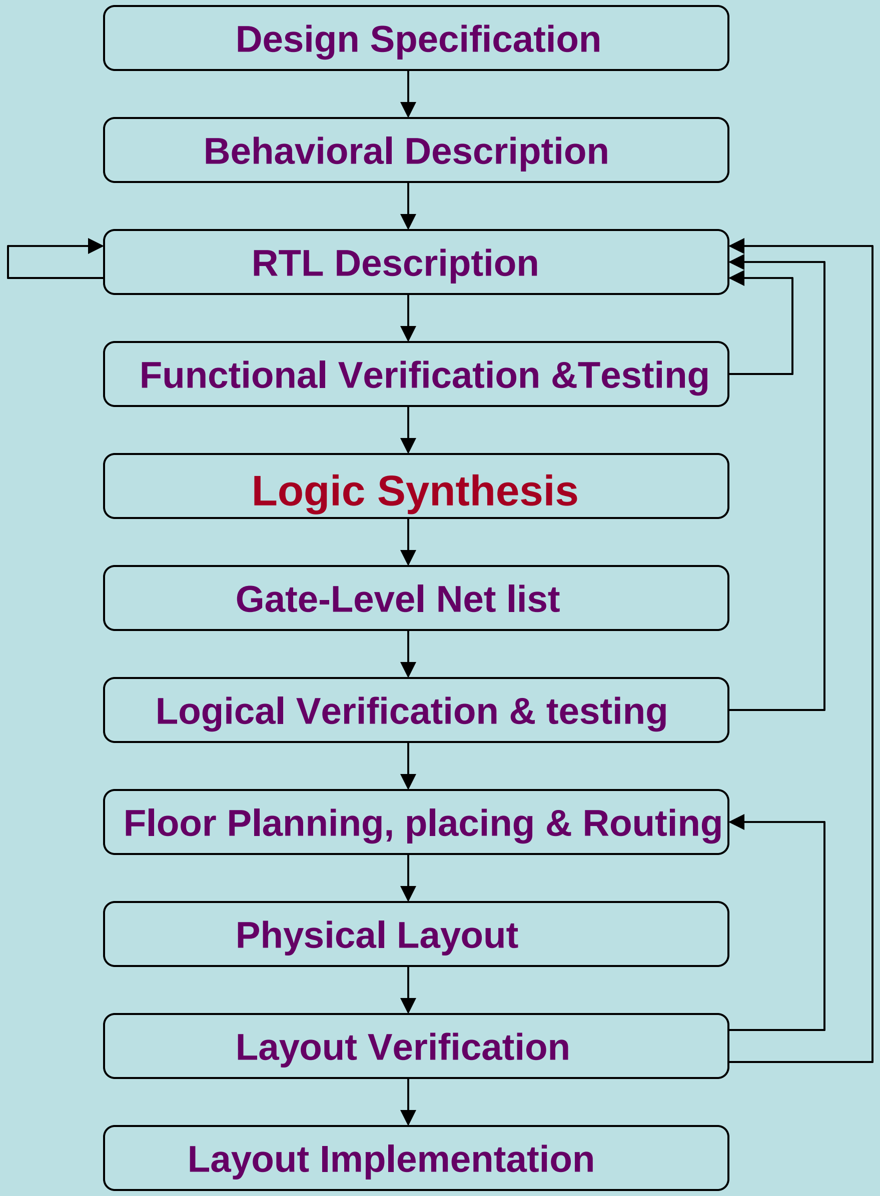
Logic synthesis is the process of converting a high-level description of the design into an optimized gate-level presentation, given a standard cell library and certain design constraints.

Standard cell library: and, or, nor etc.

Design constraints: timing, area, power

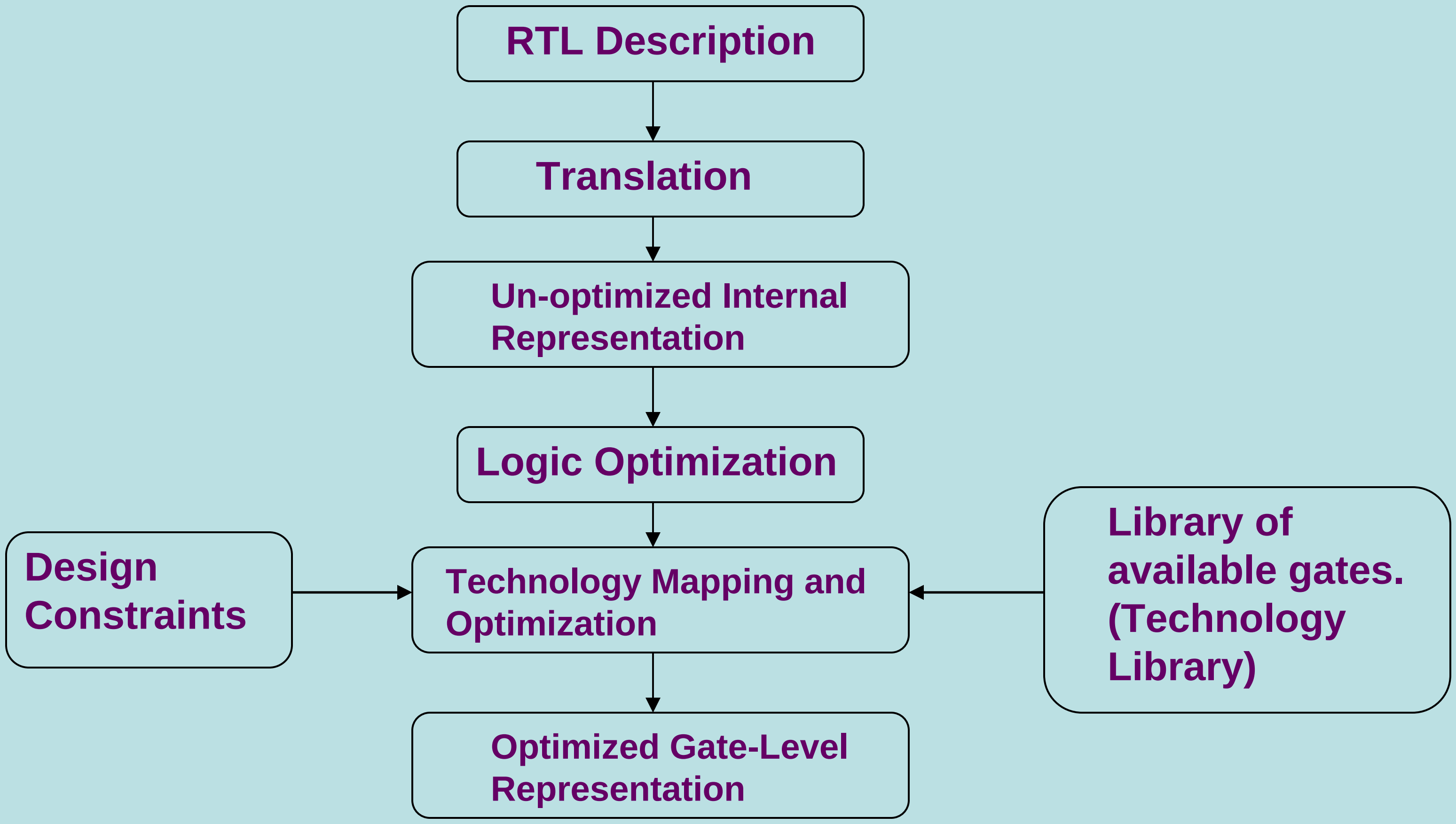

Synthesis
Structural
Behavioral
Physical Design
Physical

The Verilog Design Flow



LOGIC SYNTHESIS FLOW:

RTL TO GATES


RTL Description
Translation
Un-optimized Internal Representation
Logic Optimization
Design Constraints
Technology Mapping and Optimization
Library of available gates. (Technology Library)
Optimized Gate-Level Representation

Discussion On Each Block::

RTL description: Design at a high level using RTL constructs.

Translation: Synthesis Tool convert the RTL description to un-optimized internal representation.

Un-optimized Intermediate Representation: Represented internally by the logic synthesis tool in terms of internal data structure.

Logic Optimization: Logic is optimized to remove redundant logic.

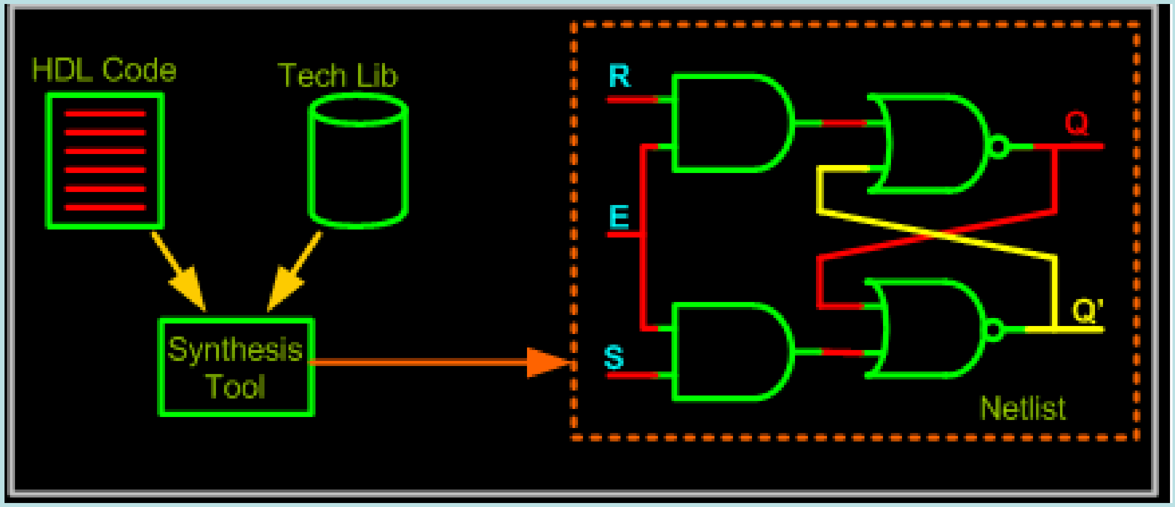
Technology Mapping and Optimization: Here the synthesis tool takes the internal representation and implements the representation in gates, using the cells provided in the technology library.

Technology Library

It contains library cells that can be basic gates or macro cells.

The cell description contains information about the following:

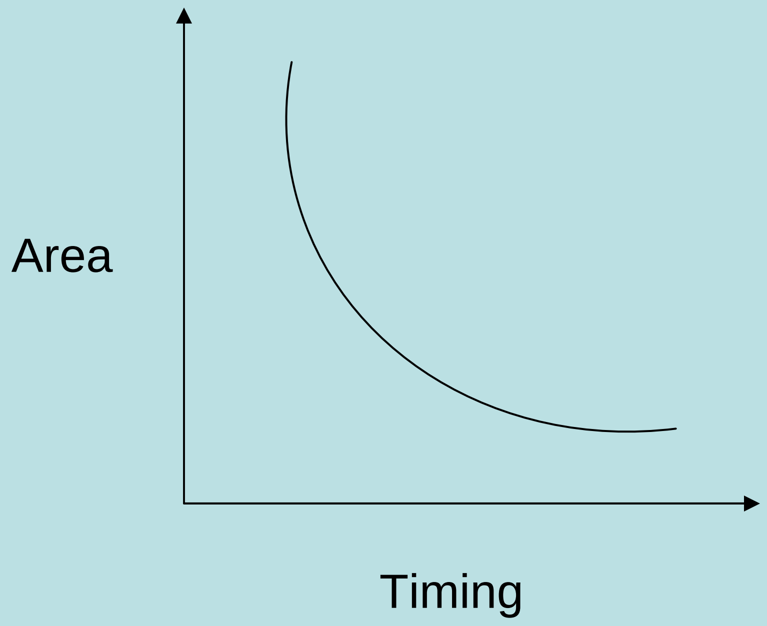
- Functionality of the cell.
- Area of the cell layout.
- Timing information about the cell.
- Power information about the cell.



Design Constraints

It includes the following:

- **Timing**
- **Area**
- **Power**



Points to note about synthesis

- For very big circuits, vendor technology libraries may yield non-optimal result.
- Translation, logic optimization and technology mapping are done internally in the logic synthesis tool and are not visible to the designer.
- Timing analyzer built into synthesis tools will have to account for interconnect delays in the total delay calculation.

Verilog HDL Synthesis

Verilog Constructs:

Ports, Parameters, Signals & variables, functions & tasks, loops, …

Verilog Operators:

Arithmetic, Logical, Bit-wise, Shift, …

Statements that don’t get Synthesis:

•Primitive definitions.

•Time declaration (#delay).

•Event declaration.

•Case operators (…===…, …!==…)

•Repeat, wait

•… and few more

Interpretation of few Verilog Constructs

The assign statement:

Adder:
```
assign {cout, sum} = a + b + cin;
assign out = (a & b) | c;
```

Multiplexer:
```
assign out = (s) ? i1 : i0;
```

The if-else statement

```
if (s)
        out = i1;
else
        out = i0;
```

The case statement

```
case (s)
        1'b0 : out = i0;
        1'b1 : out = i1;
endcase
```

For loops:

```
for (i=0; i<5;i=i+1)
  {c, sum[i]} = a[i] + b[i] + c;
```

Always statement:

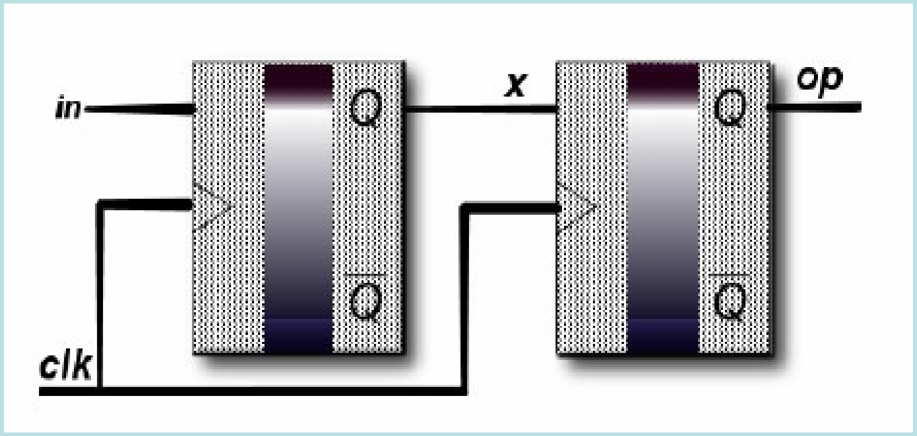
For D- Filp Flop

```
always @(posedge clk)
        q <= d;
always @(clk or d)
        if (clk)
          q <= d;
```

Verilog and Synthesis

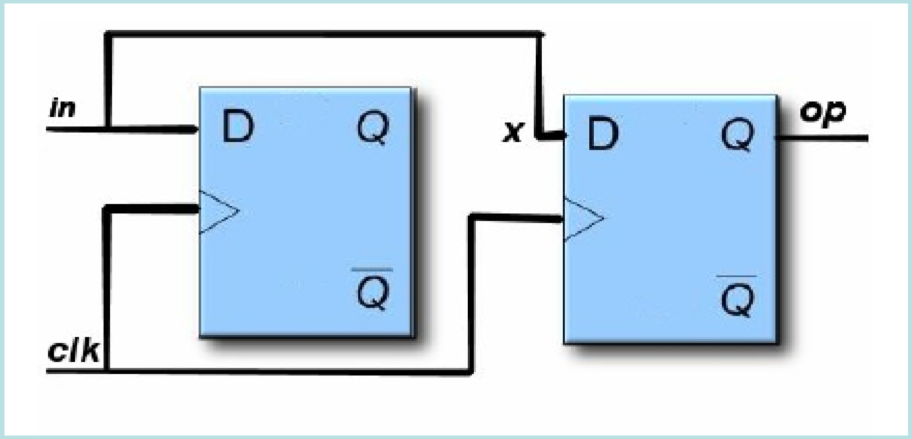
Blocking Statements :

```
always @(posedge clk)  begin
x = in;
op = x;
end
```

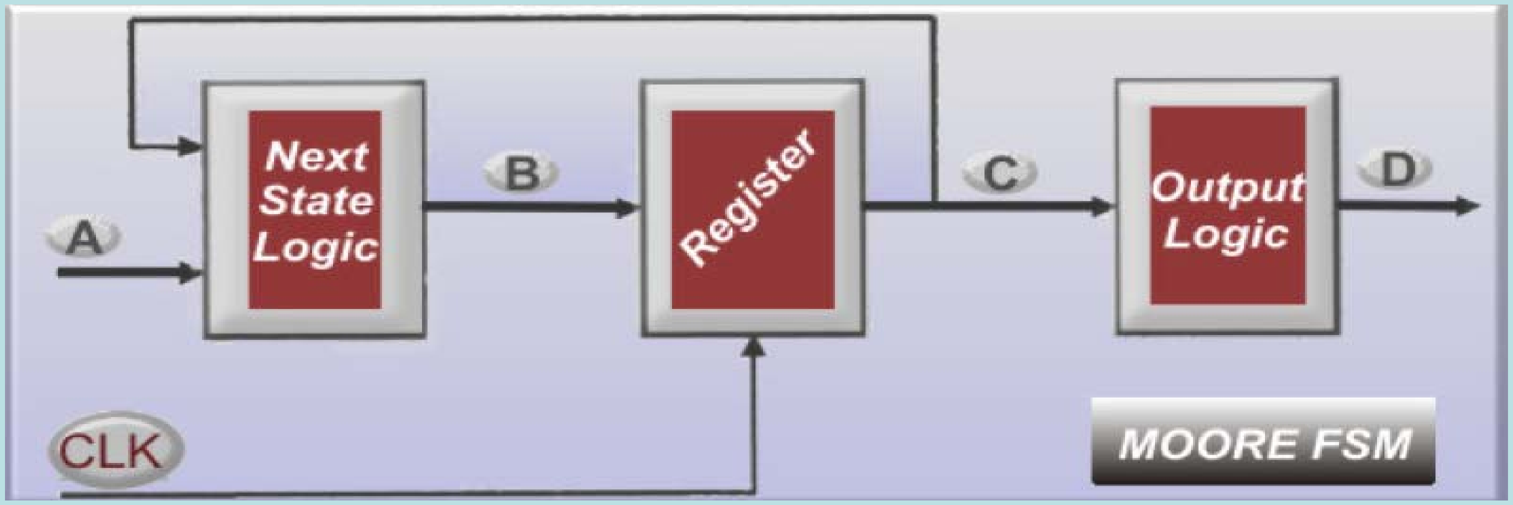


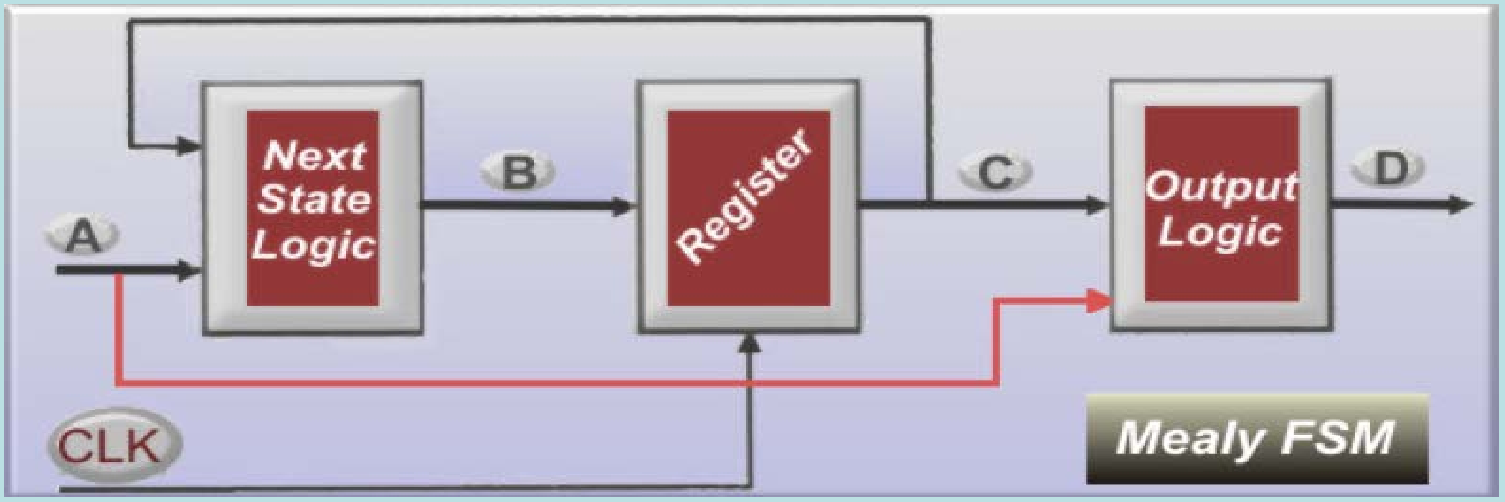
Non-Blocking Statements :

```
always @(posedge clk)  begin
x <= in;
op <= x;
end
```



STATE MACHINE HARDWARE





Verification of the Gate-Level Netlist

Functional Verification:

Identical stimulus is run with the original RTL and synthesized gate-level descriptions of the design. The output is compared to find any mismatches.

Timing Verification:

Gate-level netlist is checked for timing by use of timing simulation or by static timing verifier.

Modeling Tips for Logic Synthesis

I. Verilog Coding Style :

1. Use meaningfull names for signals and variables.
2. Avoid mixing positive and negative edge-triggered flip-flops.
3. Use basic building blocks vs. use continuous assign statements.
4. Instantiate multiplexers vs. use if-else or case statements.
5. Use parenthesis to optimize logic structure.
6. Use arithmetic operators *,/ and % vs. design building blocks.
7. Be careful with multiple assignments to the same variable.
8. Define if-else and case statements explicitly.

II. Design Partitioning:

1. Horizontal partitioning.
2. Vertical partitioning.
3. Parallelizing design structure.

III. Design Constraint Specification:

Accurate specification will produce gate-level netlist that is optimal.

Basic Problem

a. Simplification of logic equation.

-- minimization of boolean expression.

-- two level (karnaugh map, Quine Mc-clauskey, Espresso).

-- multilevel

-- reduce no. of literals.

-- factoring , common sub-expression.

$$\boldsymbol{F(A,B,C,D,E,F,G)} = ADF + AEF + BDF + BEF + CDF + CEF + G$$
$$= (A + B + C)(D + E)F + G$$

b. Mapping logic equation to gates.

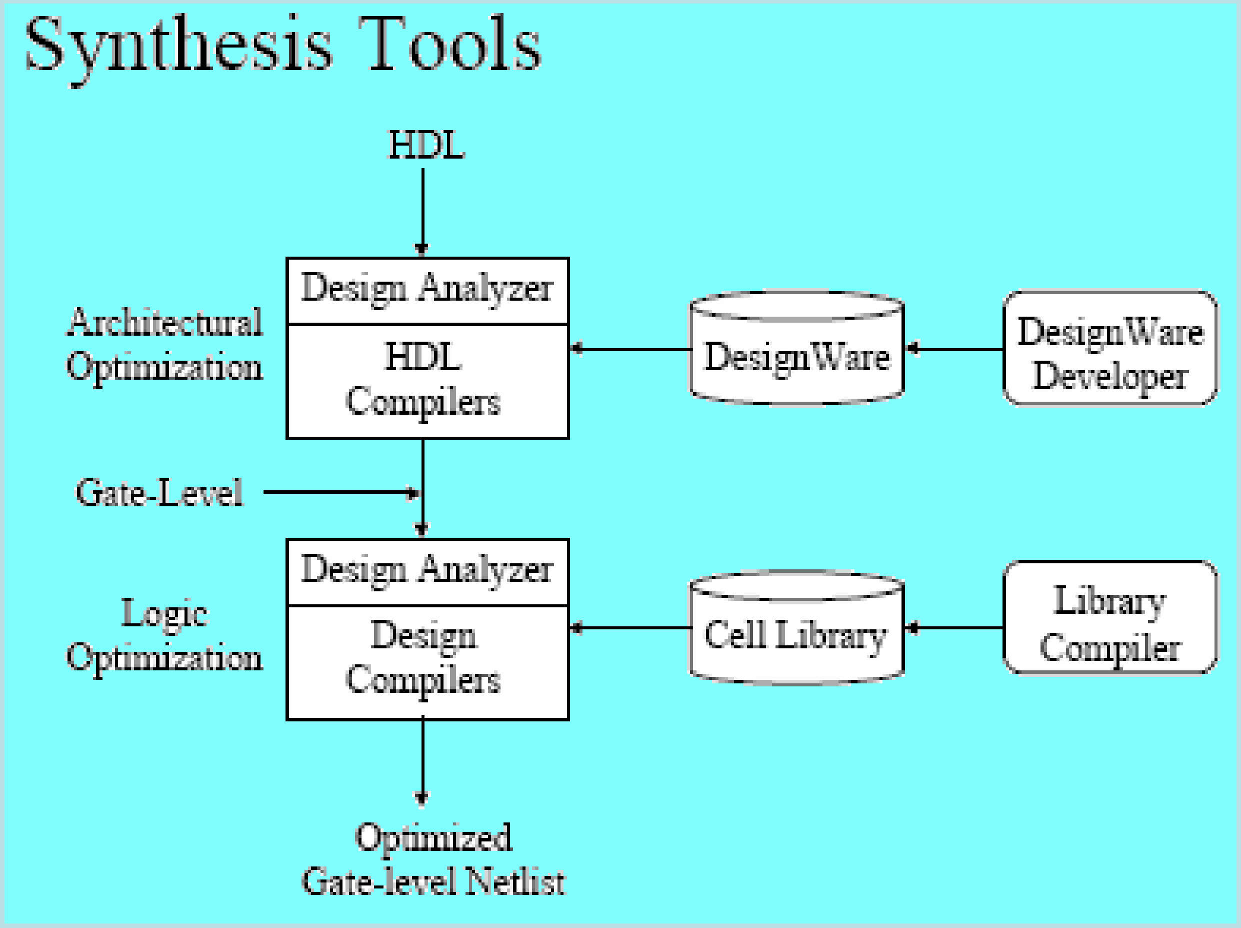
c. Gate level optimization.

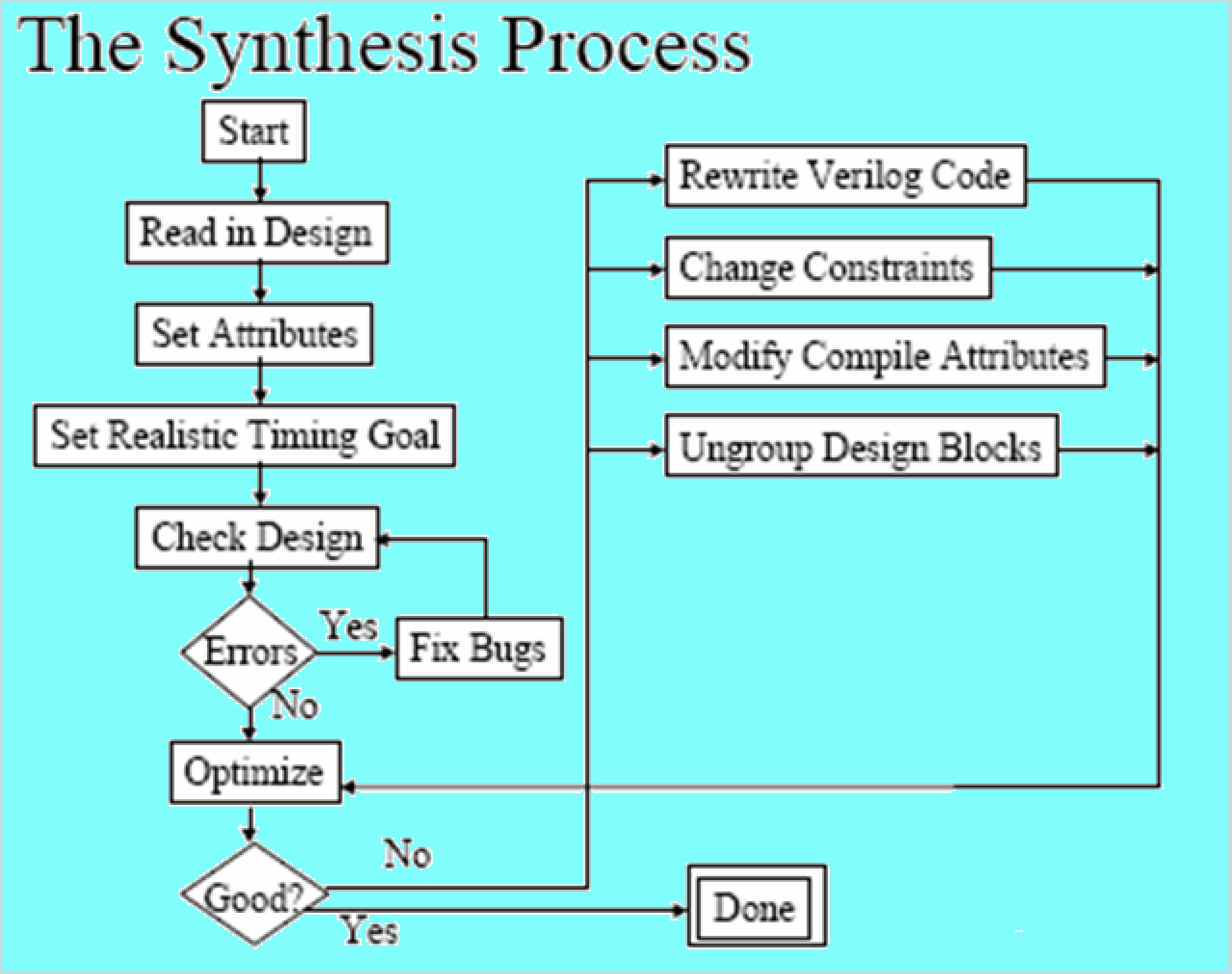
-- replace OR-NOT by NOR

d. Technology mapping.

Impact of Logic Synthesis

- No manual conversion, design is describes at the higher level of abstraction.
- High-level design is done without significant concern about design constraints.
- Conversion from high-level design to gates is fast.
- Turnaround time for redesign of blocks is shorter.
- Logic synthesis tools optimize the design as a whole.
- Logic synthesis tools allow technology-independent design.
- Design reuse is possible for technology-independent descriptions.


Synthesis Tools
HDL
Design Analyzer
HDL Compilers
Architectural Optimization
DesignWare
DesignWare Developer
Gate-Level
Design Analyzer
Design Compilers
Logic Optimization
Cell Library
Library Compiler
Optimized Gate-level Netlist


The Synthesis Process
Start
Read in Design
Set Attributes
Set Realistic Timing Goal
Check Design
Errors
Yes
Fix Bugs
No
Optimize
Good?
No
Yes
Done
Rewrite Verilog Code
Change Constraints
Modify Compile Attributes
Ungroup Design Blocks

Tools used in the lab

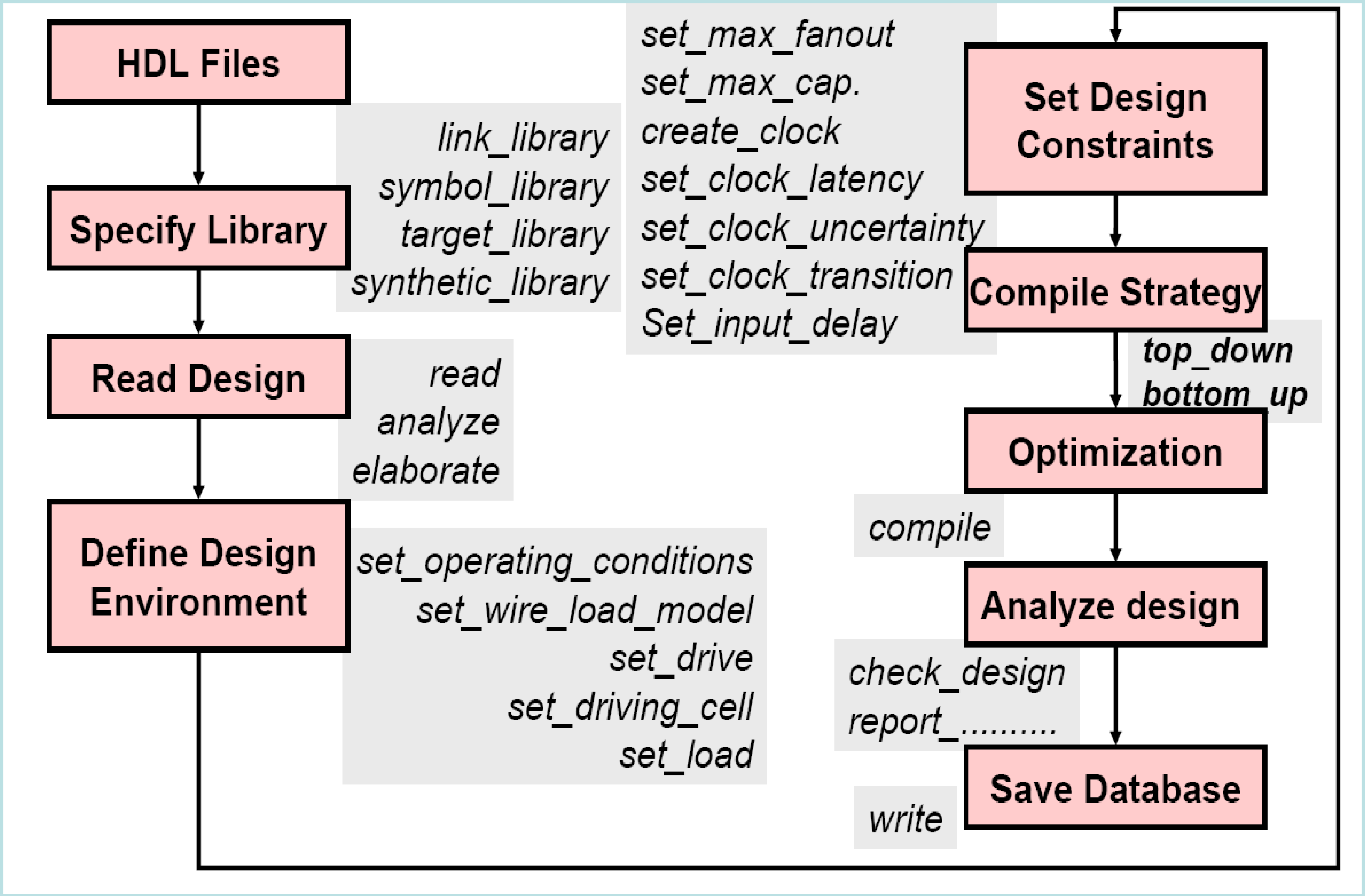
Verilog HDL → **Verilog XL** by Cadence

Logic Simulation → **Simvision** by Cadence

Logic Synthesis → **Design Compiler** by Synopsis

Physical Design → **Silicon Ensemble** by Cadence

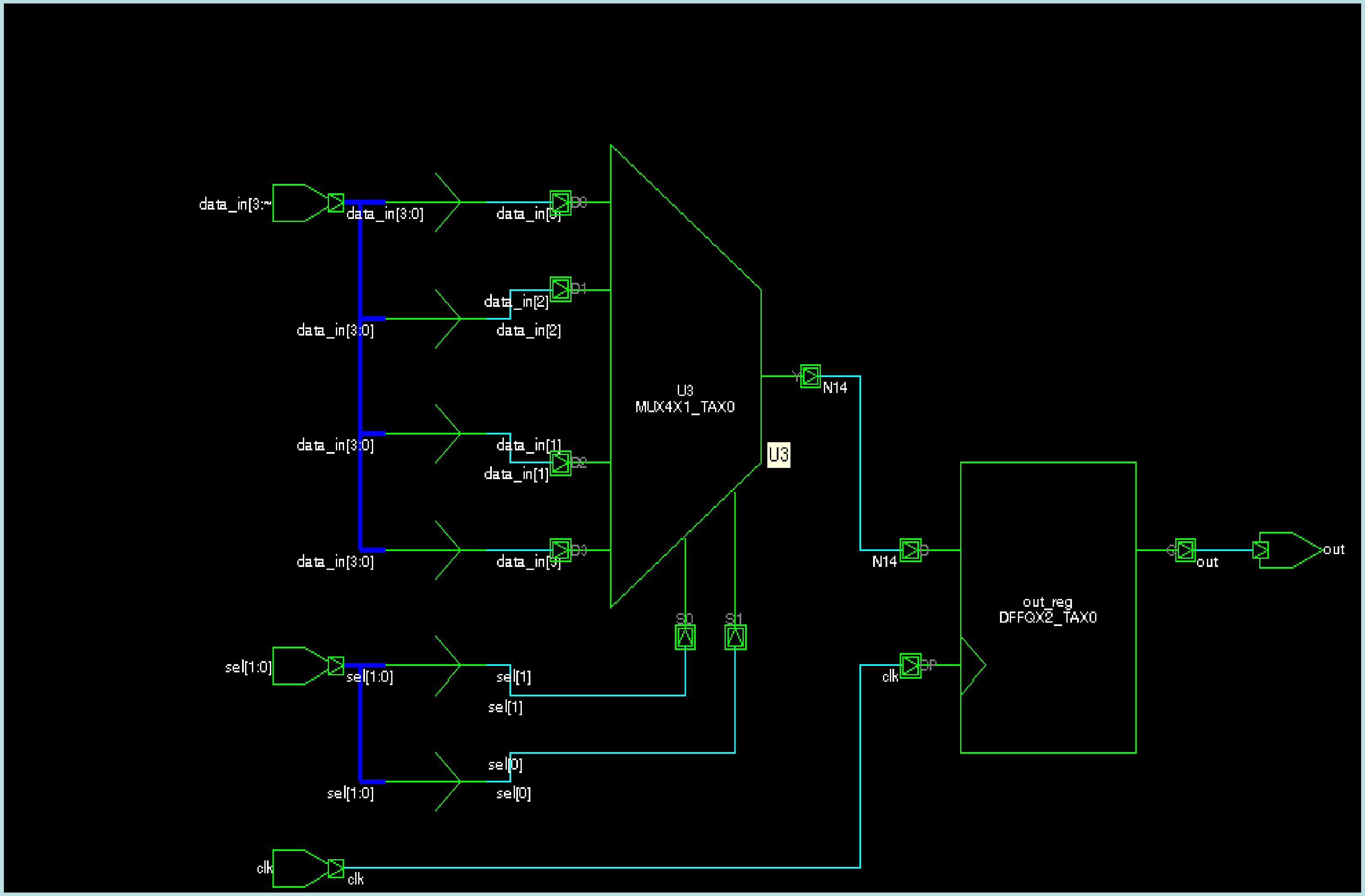
Important Command



Example of 4X1 Multiplexer

```
// Multiplexer in verilog

module muxe(data_in,sel,out,clk);

input [3:0] data_in;
input [1:0] sel;
input clk;
output out;

wire [1:0] sel;
wire [3:0] data_in;
reg out;

always @(posedge clk)
        begin
                case(sel)
                2'b00: out=data_in[0];
                2'b01: out=data_in[1];
                2'b10: out=data_in[2];
                2'b11: out=data_in[3];
                endcase
        end


endmodule
```

Synthesized Schematic View



Area Report

```
****************************************
Report : area
Design : muxe
Version: 2003.06-SP1
Date   : Mon May 12 14:22:32 2008
****************************************
Library(s) Used:

    xlite_core (File: /research1/paras/LIBS/xlite_core.db)
Number of ports:                8
Number of nets:                 9
Number of cells:                2
Number of references:           2
Combinational area:       641.500000
Noncombinational area:     712.809998
Net Interconnect area:      8.099999
Total cell area:         1354.310059
Total area:              1362.410034
```

Power Report

```
****************************************
Report : power
                  -analysis_effort low
Design : muxe
Version: 2003.06-SP1
Date   : Mon May 12 14:23:02 2008
****************************************
Library(s) Used:

    xlite_core (File: /research1/paras/LIBS/xlite_core.db)
Operating Conditions: TYP   Library: xlite_core
Wire Load Model Mode: top

Design       Wire Load Model           Library
------------------------------------------------
muxe                 10000_CELLS       xlite_core
Global Operating Voltage = 1.8
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW
  Cell Internal Power  = 139.0492 uW   (47%)
  Net Switching Power  = 158.6197 uW   (53%)
                         ---------
Total Dynamic Power    = 297.6689 uW  (100%)
Cell Leakage Power     =  13.1517 nW
```

Summary

Logic Synthesis

Input: Boolean Equations and FSMs.

Output: A netlist of gates and flip-flops.

Design Goals:

--Minimize no. of levels (delay).

--Minimize no. of gates (area).

--Minimize signal activity (power).

Typical Constraints:

--Target Library.

THANK YOU