Sharif University of Technology
Department of Computer Engineering

Digital System Design
Logic Synthesis with Verilog HDL

Siavash Bayat-Sarmadi

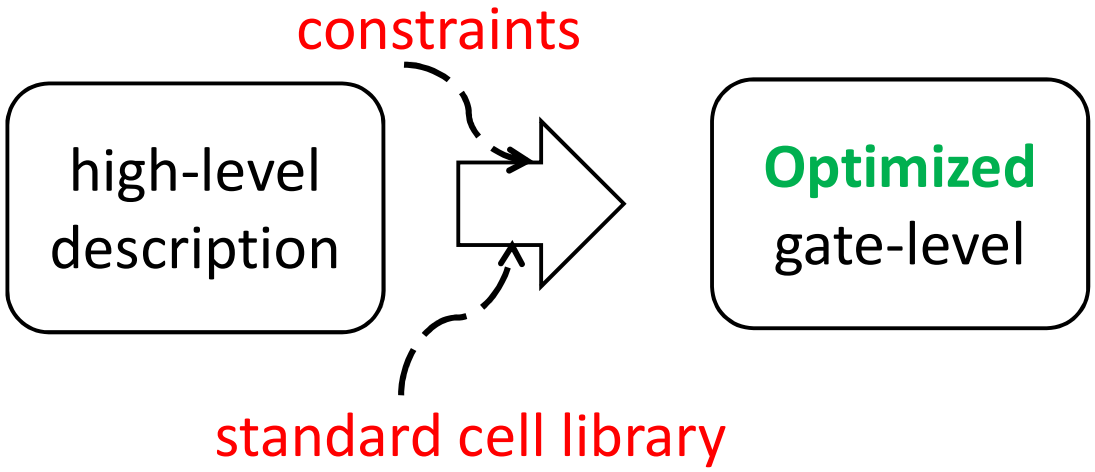
Outline

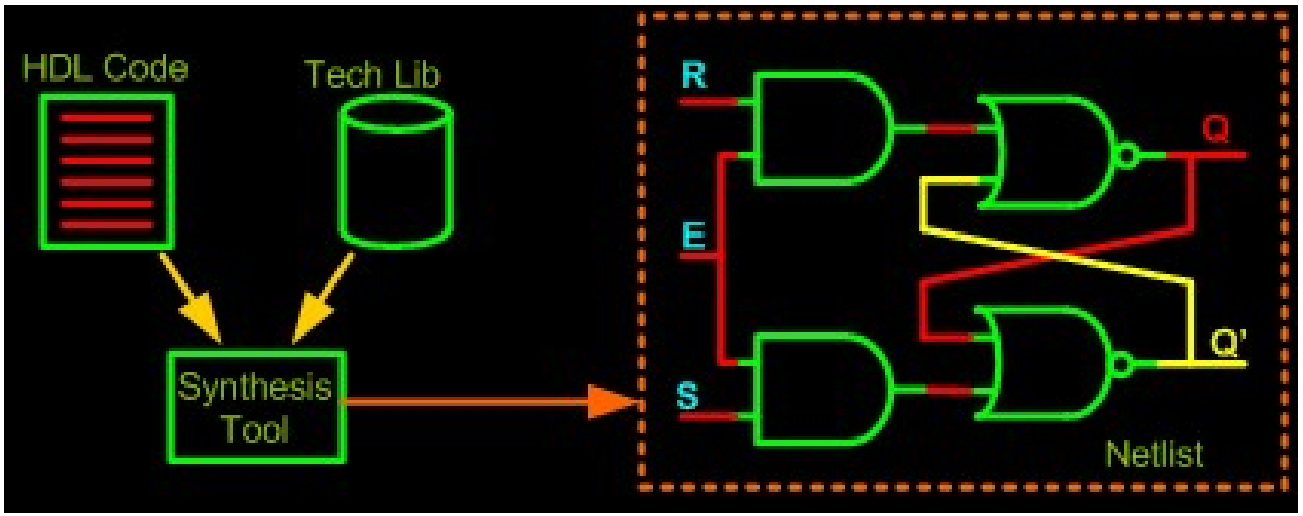
- What is logic synthesis?
- Synthesizable constructs
- Non-synthesizable constructs
- Cares must be taken
  - Combinational
  - Sequential
  - Dataflow
  - Structural
  - Behavioral
- Synthesis tools

What is Logic Synthesis? (1)

- High-level description to gate-level representation
  - given a standard cell library
    - nand, nor, mux, adder and etc.
    - Usu. known by transistor size (e.g. 0.18u)
  - and certain design constraints
    - Timing
    - Area
    - Power

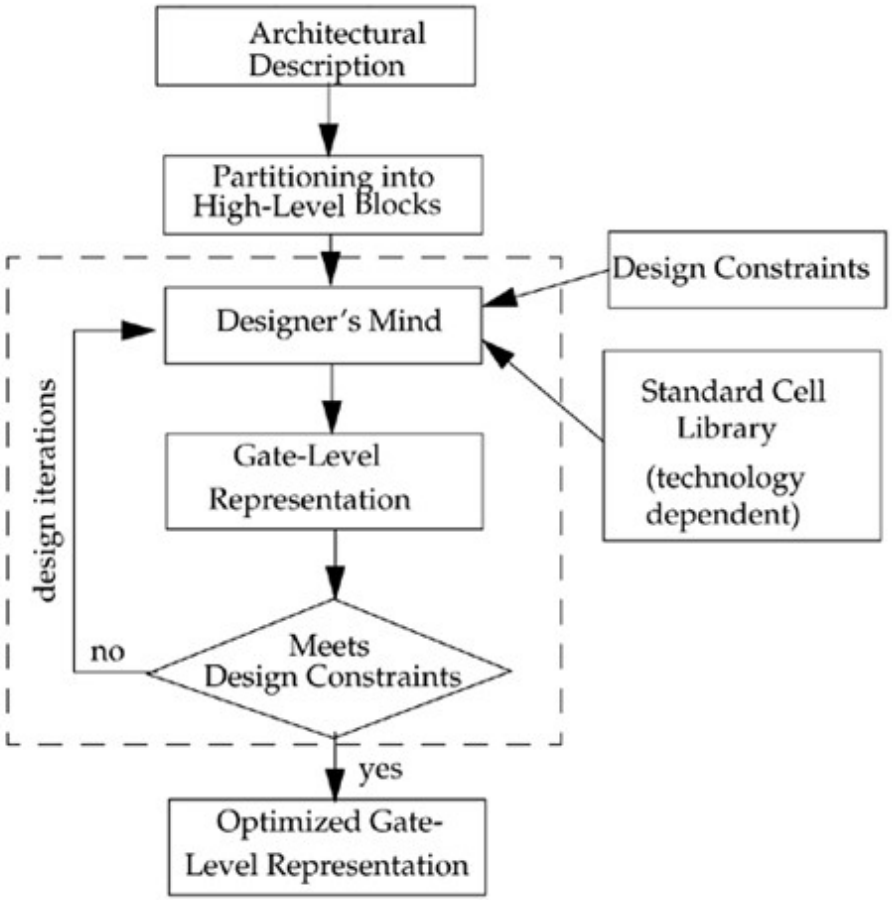
What is Logic Synthesis? (2)



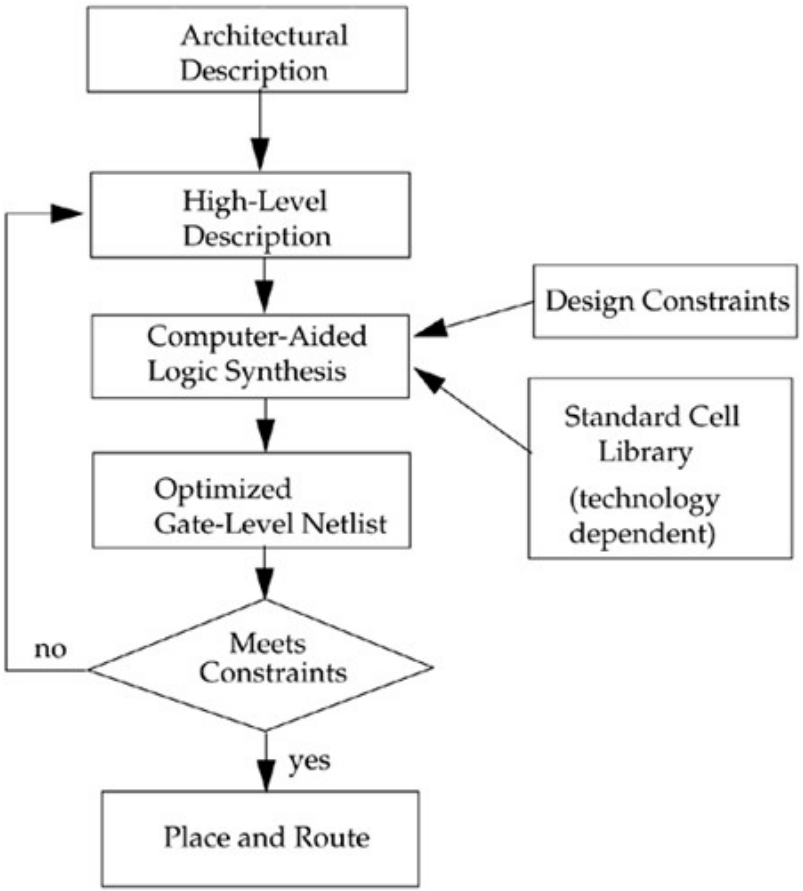


Design Flow Progress

Traditional Logic Design Flow



Logic Synthesis Flow



Causes for Synthesis Error

- Using non-synthesizable constructs
- Bad coding of synthesizable constructs
  - Flip-flop with both posedge and negedge in sensitivity list
  - Driving a reg variable from more than one always block
  - Comparing with z and x
  - Tri-state
    - Limit design to two states: 0 and 1
    - Use tri-state only at chip IO pads level

Constructs Not Supported in Synthesis (1)

Construct Type	Notes
Initial	Used only in test benches.
Events	Events make more sense for syncing test bench components.
Real	Real data type not supported.
Time	Time data type not supported.
Force and Release	Force and release of data types not supported.

Constructs Not Supported in Synthesis (2)

Construct Type	Notes
Assign and Deassign	assign and deassign of reg data types is not supported. But assign on wire data type is supported.
Fork Join	Use nonblocking assignments to get same effect.
Primitives and Table	Only gate level primitives are supported. UDP and tables are not supported.

Example of Non-Synthesizable Verilog Construct (1)

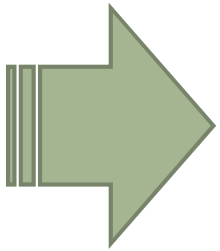
- Initial Statement

```
module synthesis_initial(clk,q,d);
input clk,d;
output q;
reg q;
   initial
   begin
      q <= 0;
   end
   always @ (posedge clk)
   begin
      q <= d;
   end
endmodule
```

Example of Non-Synthesizable Verilog Construct (2)

- Delays

```
a=#10 b
```



```
a=b
```

Just suits to simulation

After synthesis delays are eliminated

Example of Non-Synthesizable Verilog Construct (3)

- Comparison to x and z are always ignored

```
module synthesis_compare_xz (a,b);
output a;
input b;
reg a;
always @ (b)
begin
      if ((b == 1'bz) || (b == 1'bx))
            a = 1;
      else
            a = 0;
end
endmodule
```

Constructs Supported in Synthesis (1)

Construct Type	Keyword Description	Notes
ports	input, inout, output	Use inout only at IO level.
parameters	parameter	This makes design more generic
module definition	module	
signals and variables	wire, reg	Vectors are allowed

Constructs Supported in Synthesis (2)

Construct Type	Keyword Description	Notes
instantiation	module instances / primitive gate instances	E.g.: nand (out,a,b), bad idea to code RTL this way.
function and tasks	function , task	Timing constructs ignored
procedural	always, if, else, case, casex, casez	initial is not supported
procedural blocks	begin, end, named blocks, disable	Disabling of named blocks allowed

Constructs Supported in Synthesis (2)

Construct Type	Keyword Description	Notes
data flow	Assign	Delay information is ignored
named Blocks	Disable	Disabling of named block supported.
loops	for, while, forever	forever loops must contain @(posedge clk) or @(negedge clk)

Operators and Their Effect (1)

Symbol	Operation Performed	Notes
*	Multiply	Not Recommended
/	Division	Not Recommended
+	Add	Allowed
-	Subtract	Allowed
%	Modulus	Not Recommended
+	Unary plus	Allowed
-	Unary minus	Allowed

Operators and Their Effect (2)

Symbol	Operation Performed	Notes
!	Logical negation	Allowed
&&	Logical AND	Allowed
||	Logical OR	Allowed
<	Greater than	Allowed
>	Less than	Allowed
=<	Greater than or equal	Allowed
=>	Less than or equal	Allowed
==	Equality	Allowed
=!	inequality	Allowed

Operators and Their Effect (3)

Symbol	Operation Performed	Notes
&	Bitwise AND	Allowed
&~	Bitwise NAND	Allowed
|	Bitwise OR	Allowed
|~	Bitwise NOR	Allowed
^	Bitwise XOR	Allowed
^~ ~^	Bitwise XNOR	Allowed
<<	Right shift	Allowed
>>	Left shift	Allowed
{ }	Concatenation	Allowed
?	conditional	Allowed

Verilog Abstraction Layers

- Gate Level
  - All Synthesizable
- Dataflow Models
  - Most expressions synthesizable
  - Exceptions: *, / , %, ===, !==
- Behavioral Models
  - initials are ignored
  - always: care must be taken
  - Control statements are allowed

Combinational Circuit Modeling (1)

- Using
  - assign
  - always
- always @(list of all inputs)
- Normally blocking assignments are used for combinational circuits.

Combinational Circuit Modeling (2)

- Unintended latch after synthesis
  - Not assigning a value to a variable in some cases
    - case statements
    - if else statements
  - Synthesis tool look for an initial value
  - When does not find, assigns it's last value
    - so latch is created!
- Always drive a value to the LHS variable in the beginning of always code

Example 1


```
module decoder (in,out);
input [2:0] in;
output [7:0] out;
wire [7:0] out;
assign out =
      (in == 3'b000 ) ? 8'b0000_0001 :
      (in == 3'b001 ) ? 8'b0000_0010 :
      (in == 3'b010 ) ? 8'b0000_0100 :
      (in == 3'b011 ) ? 8'b0000_1000 :
      (in == 3'b100 ) ? 8'b0001_0000 :
      (in == 3'b101 ) ? 8'b0010_0000 :
      (in == 3'b110 ) ? 8'b0100_0000 :
      (in == 3'b111 ) ? 8'b1000_0000 :8'h00;
endmodule
```

Example 2


```
module decoder_always (in,out);
input [2:0] in; output [7:0] out;
reg [7:0] out;
always @ (in)
      begin
            out = 0;
            case (in)
            3'b000 : out = 8'b0000_0001;
            3'b001 : out = 8'b0000_0010;
            3'b010 : out = 8'b0000_0100;
            3'b011 : out = 8'b0000_1000;
            3'b100 : out = 8'b0001_0000;
            3'b101 : out = 8'b0010_0000;
            3'b110 : out = 8'b0100_0000;
            3'b111 : out = 8'b1000_0000;
            endcase
      end
```

Example 3


```
module addbit (a, b, ci, sum, co);
     input a, b, ci;
     output sum, co;
     assign {co,sum} = a + b + ci;
endmodule
```

Example 4


```
module mux_21 (a,b,sel,y);
    input a, b;
    output y;
    input sel;
    wire y;
    assign y = (sel) ? b : a;
endmodule
```

Sequential Circuit Modeling

- Using edge sensitive elements in the sensitive list
- Only use always blocks
- Normally non-blocking assignments are used
- always @ (posedge clk)
  - Asynchronous inputs are also allowed
  - E.g., always @ (posedge clk, posedge reset)

Example


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

always @ (posedge clk )
begin
     if (reset == 1)
          q <= 0;
     else
          q <= d;
end
endmodule
```

Verilog Coding Style (1)

- Use meaningful names for signals and variables
- Don't mix level and edge sensitive elements in the same always block
- Avoid mixing positive and negative edge-triggered flip-flops
- Use parentheses for code readability
- Use continuous assign statements for simple combo logic

Verilog Coding Style (2)

- Use nonblocking for sequential and blocking for combo logic
- Don't mix blocking and nonblocking assignments in the same always block (even if the tool supports them!!)
- Be careful with multiple assignments to the same variable
- Define all branches of if-else or case statements explicitly

Famous Synthesis Tools

Company	Famous Synthesis Tool (or Design Environment)
Mentor Graphics	Leonardo Spectrum
Synopsys	Design Compiler, Synplify
Cadence	Encounter RTL Compiler
Altera (FPGA company)	Quartus
Xilinx (FPGA company)	ISE