SRM Institute of Science and Technology Tiruchirappalli
Faculty of Engineering and Technology
Dept. of Electronics and Communication Engineering

21ECC303T
VLSI Design and Technology

Unit I – Introduction to Verilog HDL and Coding

Part I - Introduction

Prepared by
Dr. S. Aditya,
AP/Dept. of ECE,
SRMIST Tiruchirappalli

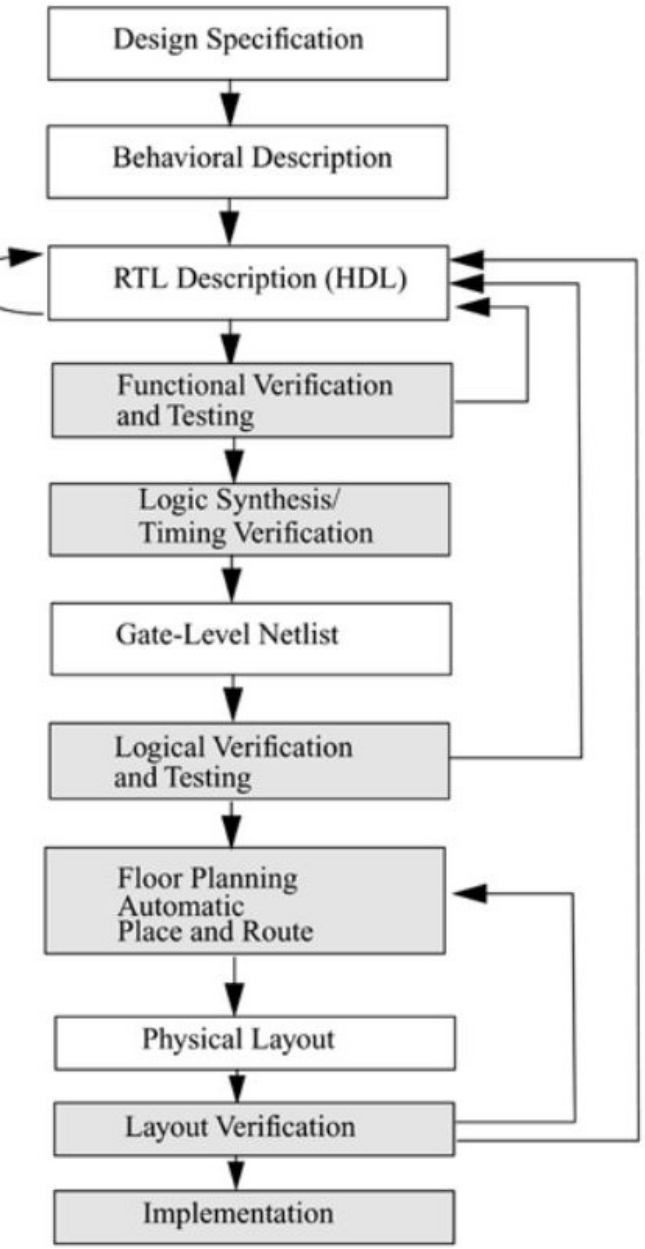
Hardware Description Language (HDL)

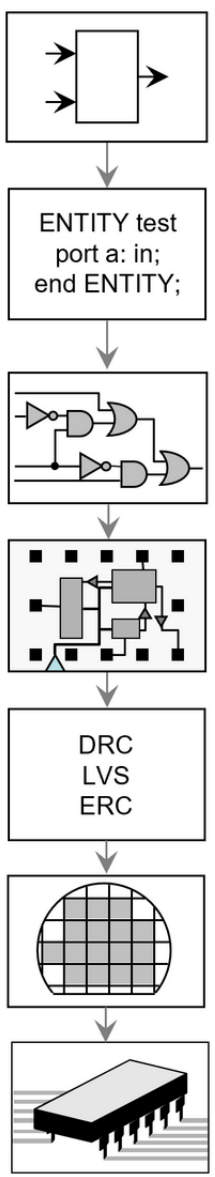
- 100,000 transistors
- Computer aided techniques for verification and design of VLSI digital circuits.
- HDL describes the hardware as programs and also verify the functionality of the design before they are actually fabricated on chip.
- HDLs are used for system level design
- Modeling language rather than a Computational language
- HDLs, were used for simulation of system boards, Interconnect buses, FPGAs (Field Programmable Gate Arrays) and PALs (Programmable Array Logic)

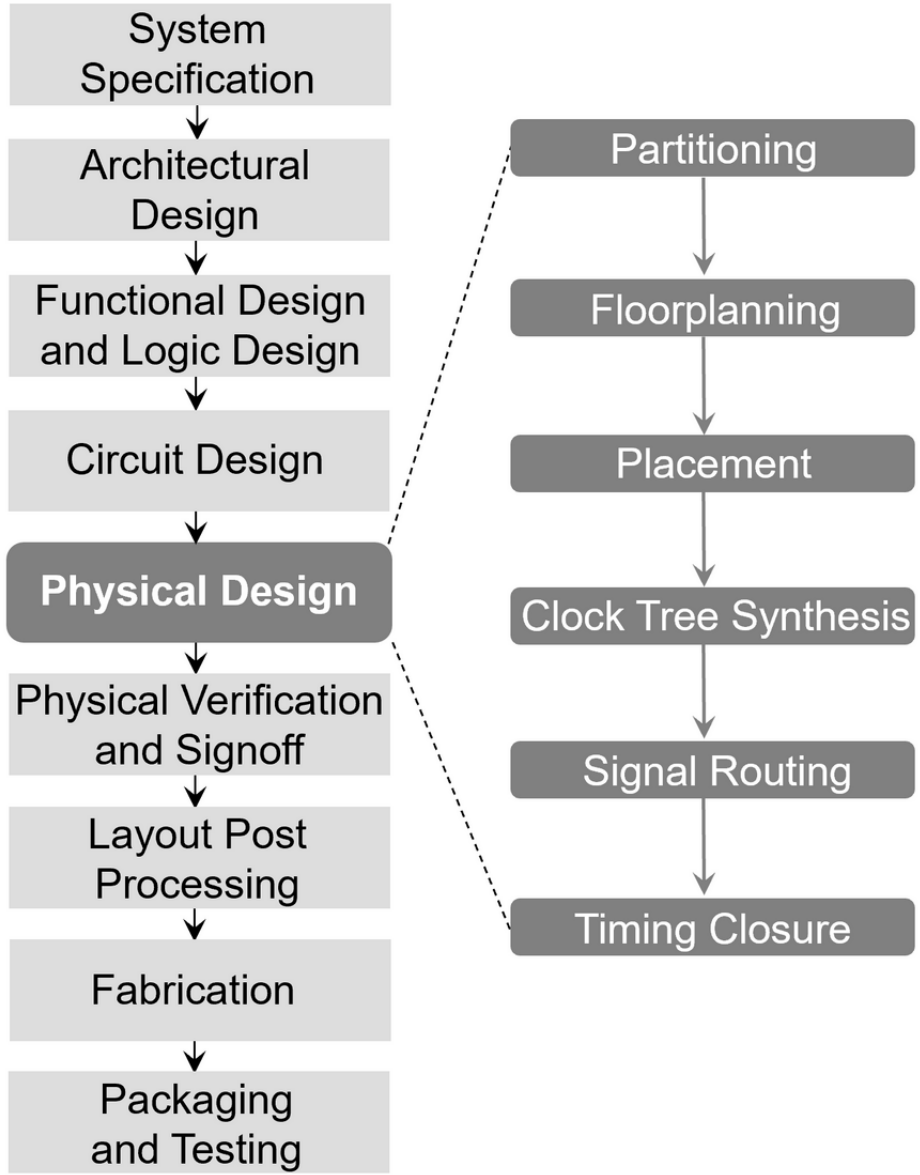
HDL Types:

- Verilog HDL (Verification of Logic)
  - easier and simpler to learn
  - weakly typed, concise with efficient notation and deterministic
- VHDL (Very High-speed Integrated Circuit Hardware Description Language (VHSIC))
  - more capable in modeling abstract behavior
  - more difficult to learn
  - strongly typed, verbose and deterministic

Typical Design Flow





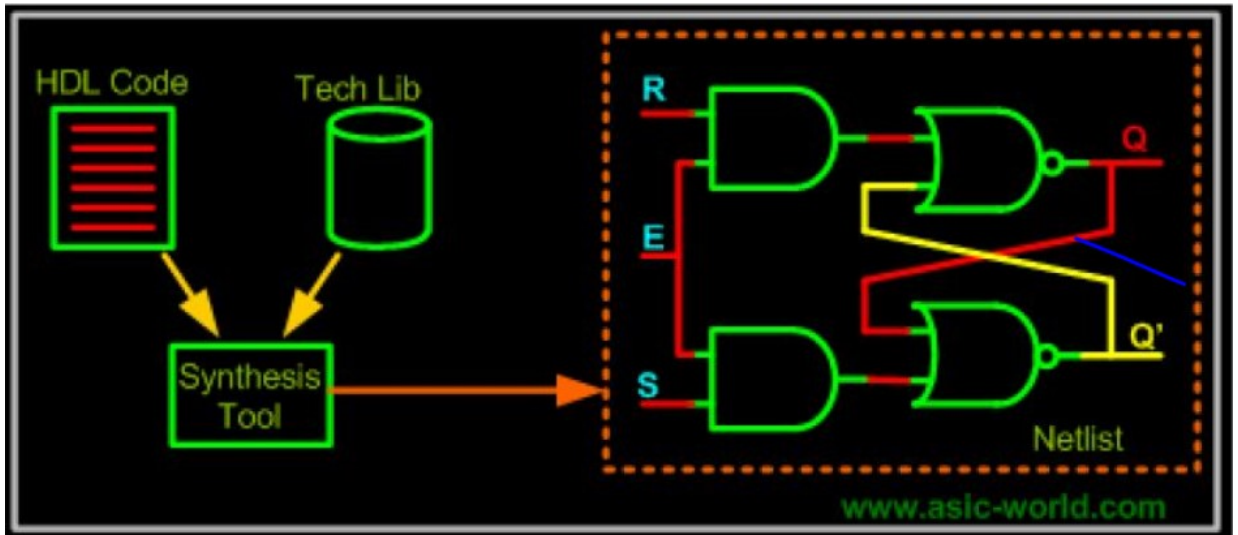


Introduction to Verilog HDL

- Standard Hardware description Language
- Similar in syntax to the C programming language
- Different Levels of abstraction to be mixed in the same model
- Hardware model can be defined in terms of switches, gates, (Register transfer Level) RTL or behavioral code

Logic Synthesis: converting high-level description of design into an optimized gate level representation.

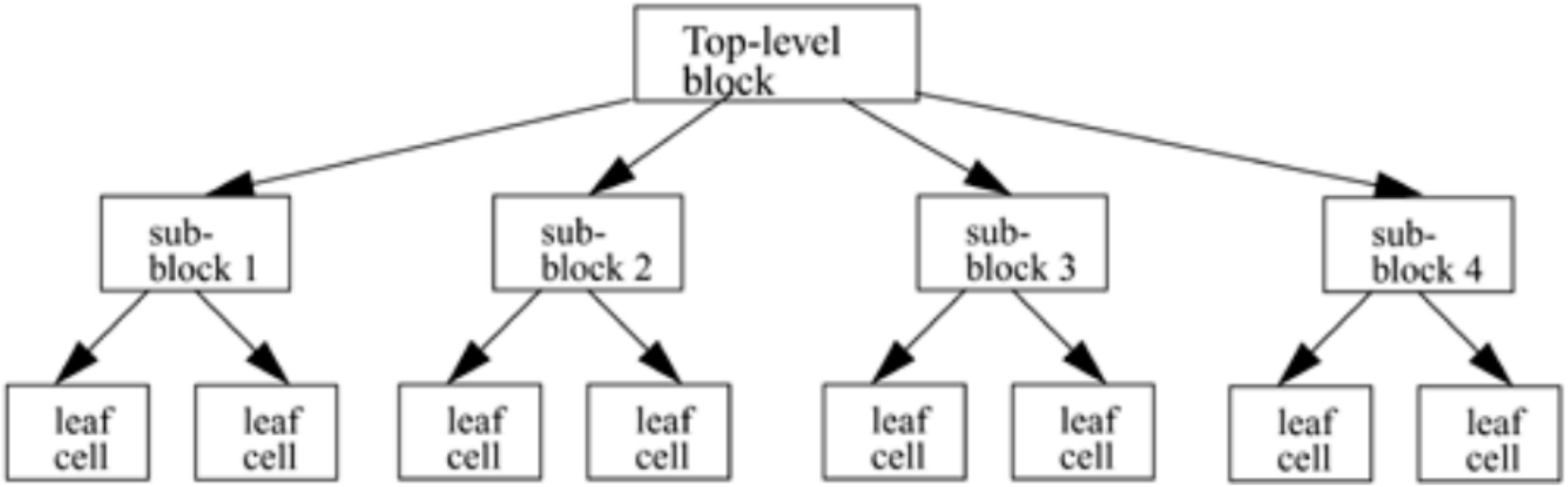
Uses a standard cell library which have simple cells such as basic gates (and, or, nand, nor, xor, xnor and not) or macro cells such as adders, muxes, memory and flip-flops.



Design Methodologies

Top-down Design Methodology:

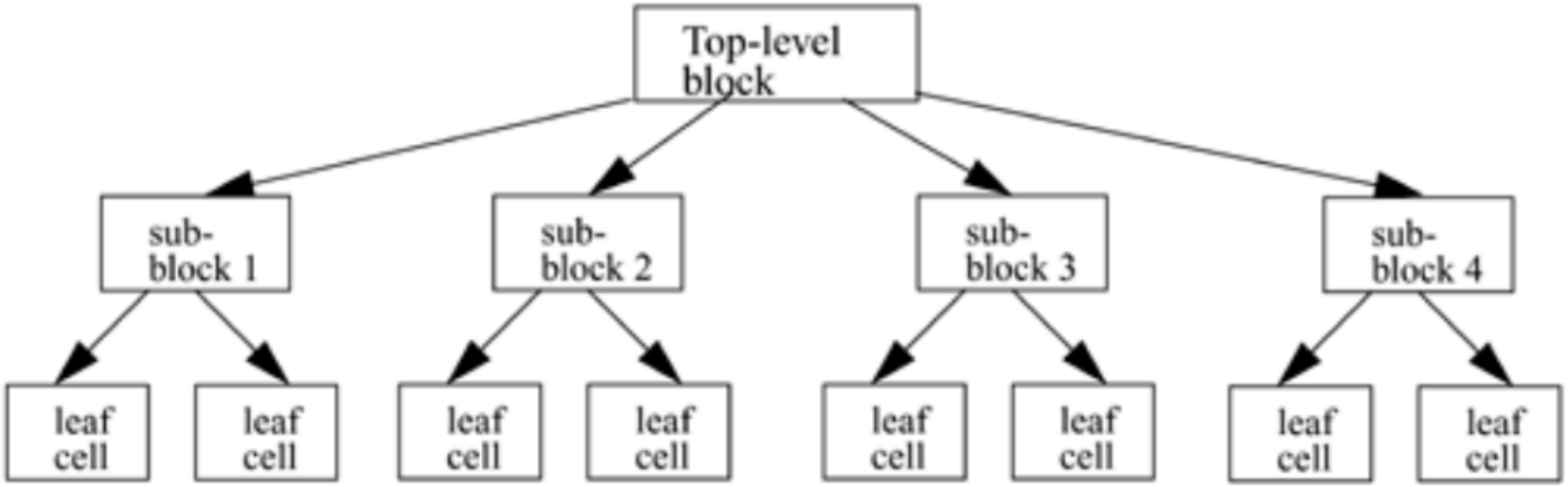
- define the top-level block and identify the sub-blocks necessary to build the top-level block.
- We further subdivide the sub-blocks until we come to leaf cells, which are the cells that cannot further be divided.



Design Methodologies

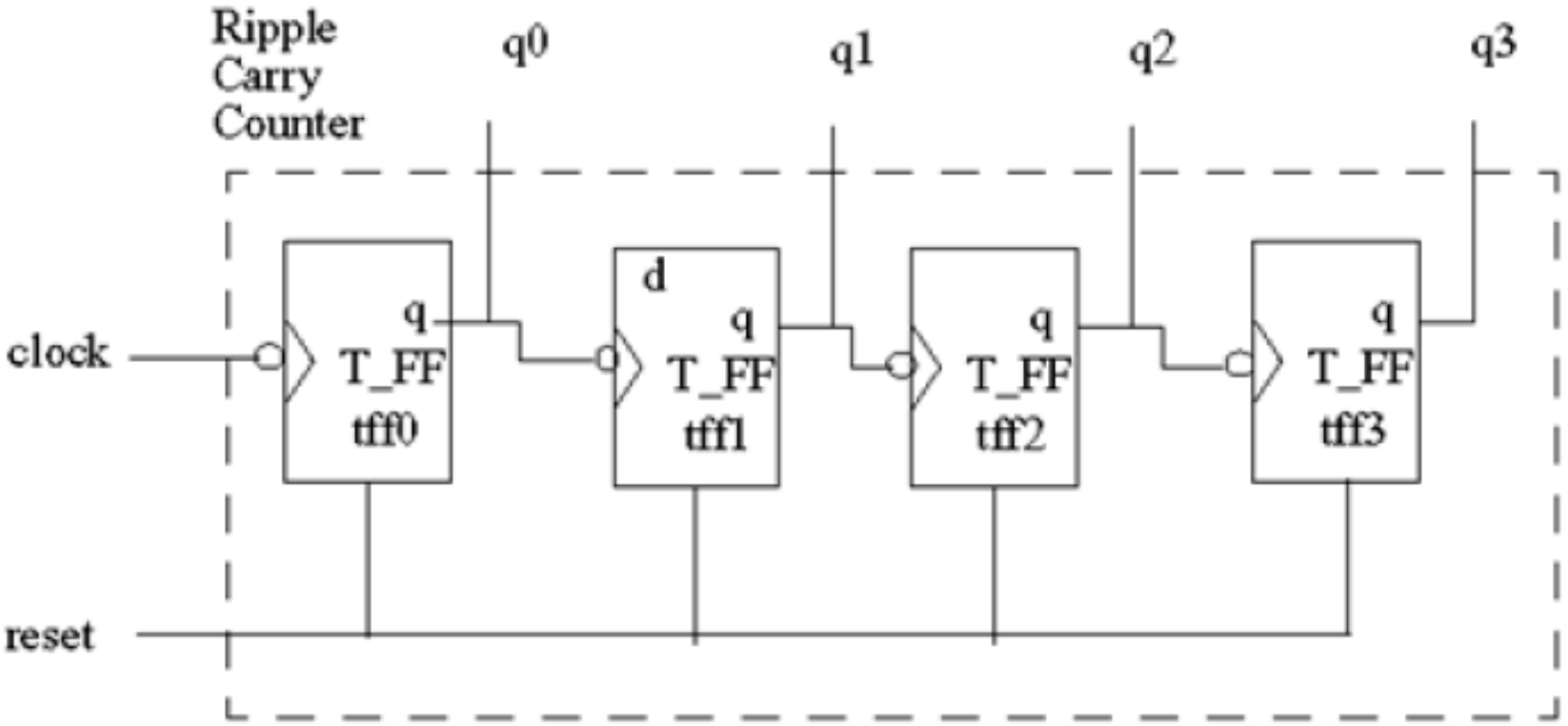
Bottom-up Design Methodology:

- First identify the building blocks that are available to us.
- We build bigger cells, using these building blocks. These cells are then used for higher-level blocks until we build the top-level block in the design.



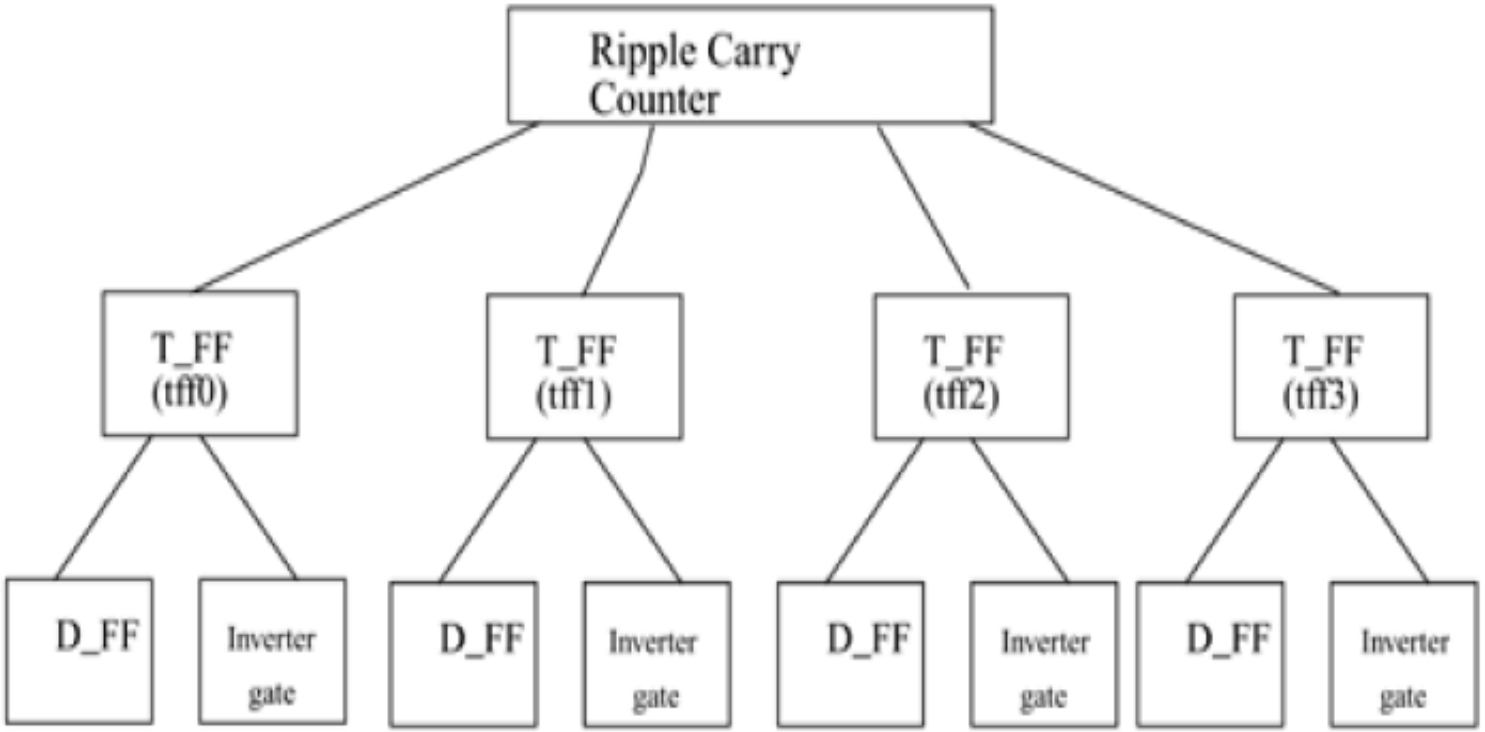
Hierarchical modeling with an example

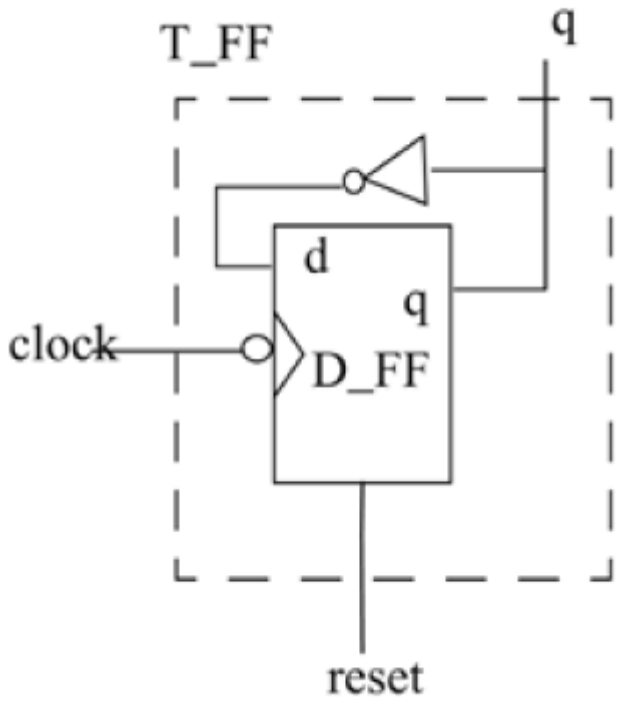
4-bit Ripple Carry Counter:



Hierarchical modeling with an example

4-bit Ripple Carry Counter:





Hierarchical Mixed modeling

Mixed design Methodology:

- a combination of top-down and bottom-up flows is used.
- Design architects define the specifications of the top-level block.
- Logic designers decide how the design should be structured by breaking up the functionality into blocks and sub-blocks.
- At the same time, circuit designers are designing optimized circuits for leaf-level cells.
- They build higher-level cells by using these leaf cells.
- The flow meets at an intermediate point where the switch-level circuit designers have created a library of leaf cells by using switches, and the logic level designers have designed from top-down until all modules are defined in terms of leaf cells.

Module

- A module is the basic building block in Verilog.
- It can be an element or a collection of lower-level design block.
- A module provides the necessary functionality to the higher-level block through its port interface (inputs and outputs), but hides the internal implementation.
- **Syntax:**

```
module <module_name> (<module_terminal_list>);
...
<module internals>
...
...
endmodule
```

- **Example: T-Flipflop module definition**

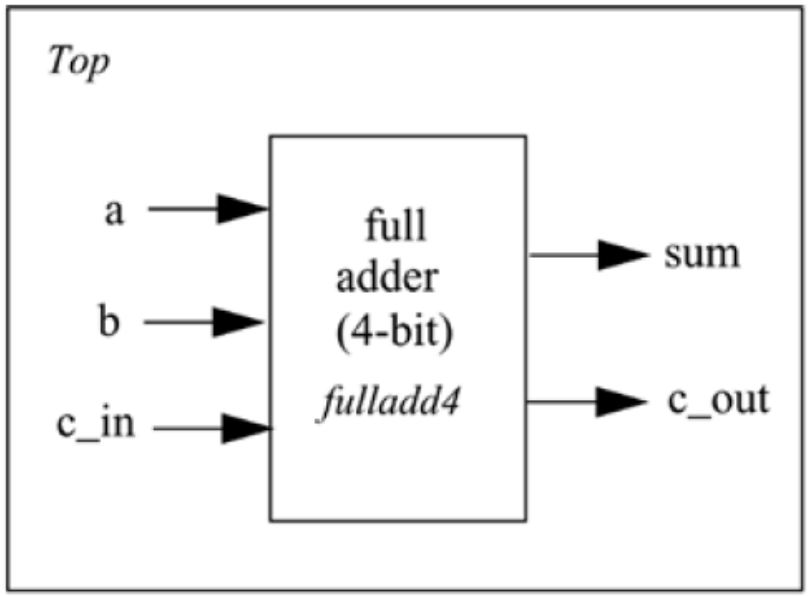
```
module T_FF (q, clock, reset);
.
.
<functionality of T-flipflop>
.
.
endmodule
```

Ports

- Ports provide the interface by which a module can communicate with its environment.
- **List of Ports:**
- A module definition contains an optional list of ports.
- If the module does not exchange any signals with the environment, there are no ports in the list.

I/O Ports for Top and Full Adder

```
module fulladd4(sum, c_out, a, b, c_in);
module Top;
```



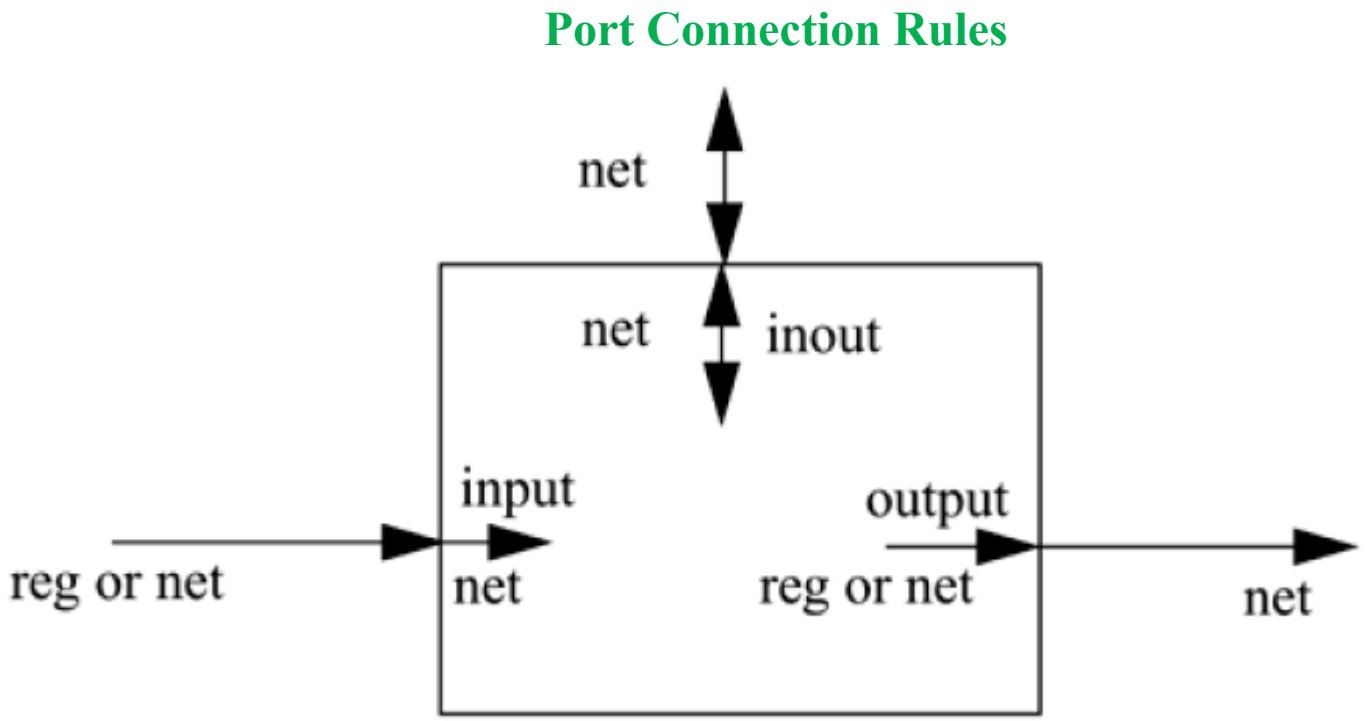
- **Port Declarations:**

```
module fulladd4(sum, c_out, a, b, c_in);
output[3:0] sum;
output c_cout;
input [3:0] a, b;
input c_in;
...
<module internals>
...
endmodule
```

Verilog Keyword	Type of Port
input	Input port
output	Output port
inout	Bidirectional port

Port connection rules

- A port consist of two units.
  - *one unit that is internal to the module*
  - *another that is external to the module.*
- There are rules governing port connections when modules are instantiated within other modules.



Module Internals: four levels of abstraction

- **Behavioral or algorithmic level**
  - highest level of abstraction
  - Design as algorithm without concern of hardware implementation details.
- **Dataflow level**
  - Design is aware of the dataflow between the hardware registers.
- **Gate level**
  - The module is implemented in terms of logic gates and interconnections between these gates.
  - Gate level logic diagram
- **Switch level**
  - Lowest level of abstraction
  - Module is implemented in terms of switches, storage nodes, and the interconnections between them.

- Verilog allows the designer to mix and match all four levels of abstractions in a design.
- The term register transfer level (RTL) is a Verilog description that uses a combination of behavioral and dataflow constructs and is acceptable to logic synthesis tools.
- If a design contains four modules, Verilog allows each of the modules to be written at a different level of abstraction.

Instances

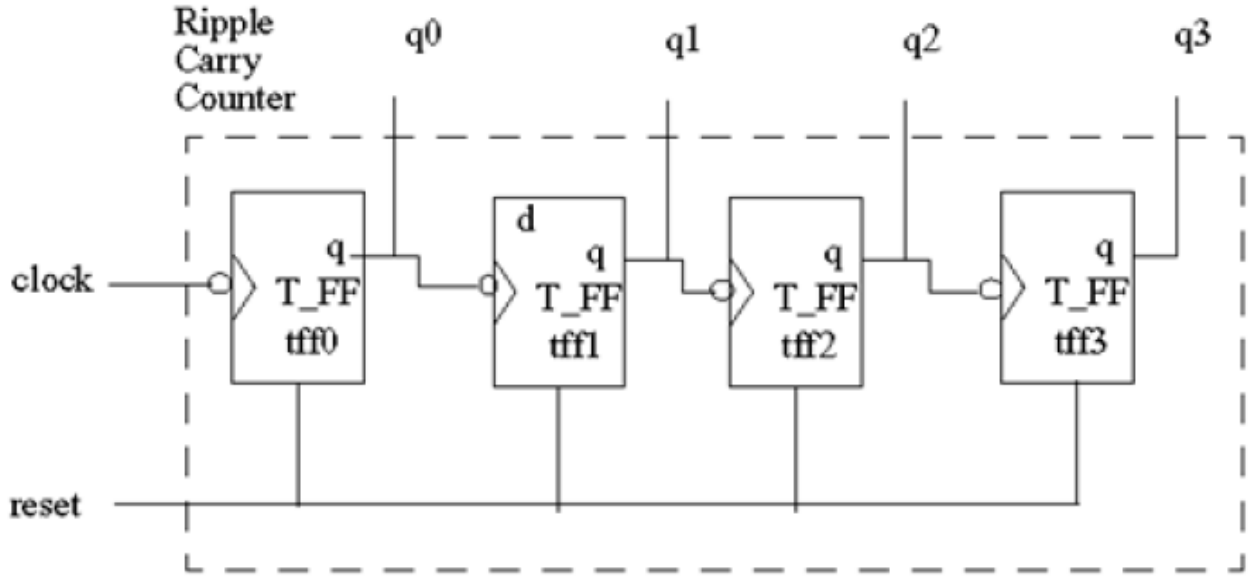
- Instances are the Objects created from a module template.
- When a module is invoked, Verilog creates a unique object from the template.
- Each object has its own name, variables, parameters, and I/O interface.
- The process of creating objects from a module template is called instantiation, and the objects are called instances.

Note:

- In Verilog, it is illegal to nest modules.
- Instead, a module definition can incorporate copies of other modules by instantiating them.

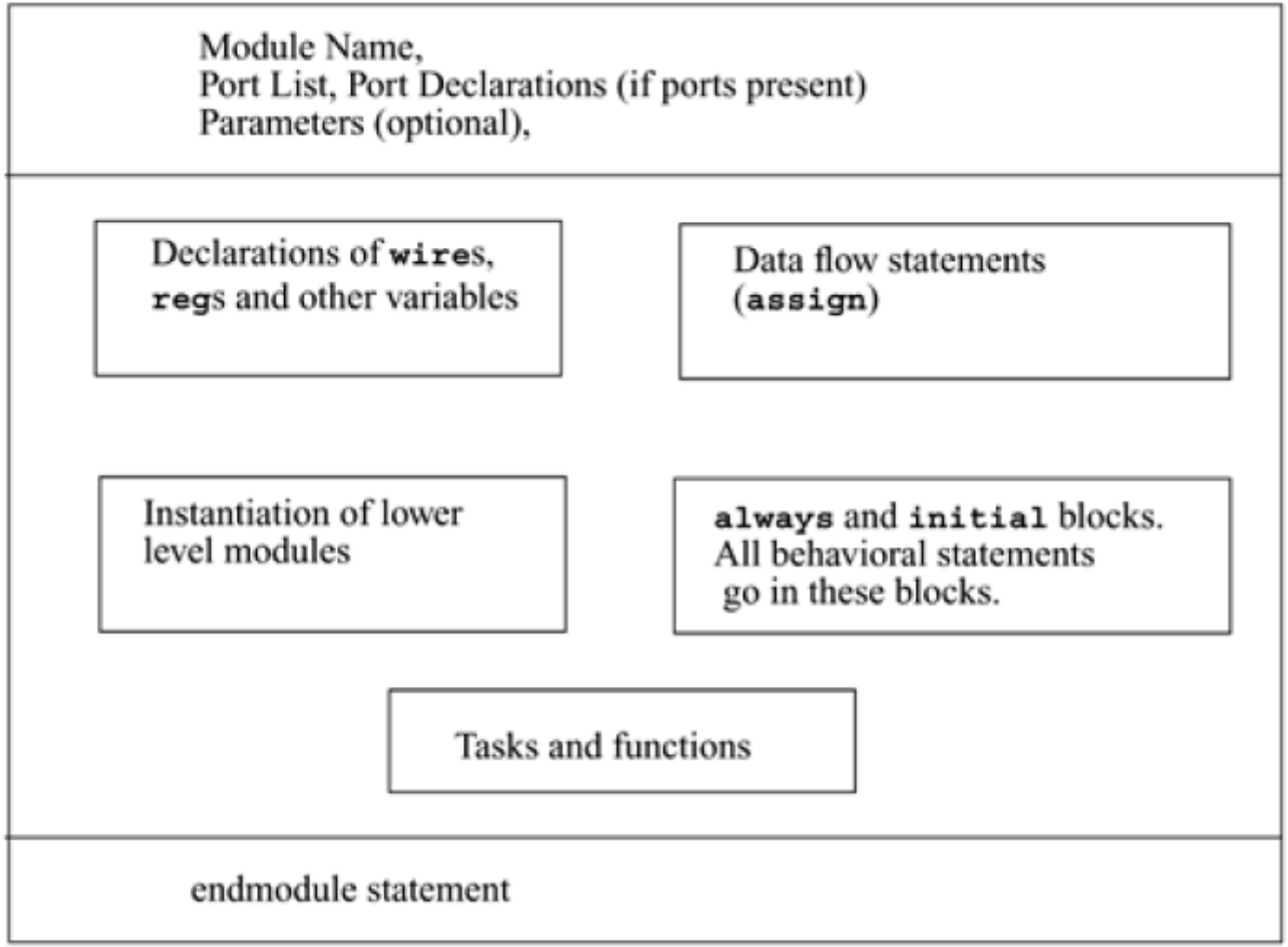
Example Instances

```
module ripple_carry_counter(q, clk, reset);
output [3:0] q; input clk, reset;
T_FF tff0(q[0],clk, reset);
T_FF tff1(q[1],q[0], reset);
T_FF tff2(q[2],q[1], reset);
T_FF tff3(q[3],q[2], reset);
endmodule
```




```
module T_FF(q, clk, reset);
output q;
input clk, reset;
wire d;
D_FF dff0(q, d, clk, reset);
not n1(d, q);
endmodule
```

Components of Verilog Module



Example : Design Block for RCC

Ripple Carry Counter Top Block Design

```
module ripple_carry_counter(q, clk, reset);
output [3:0] q;
input clk, reset;
T_FF tff0(q[0],clk, reset);
T_FF tff1(q[1],q[0], reset);
T_FF tff2(q[2],q[1], reset);
T_FF tff3(q[3],q[2], reset);
endmodule
```

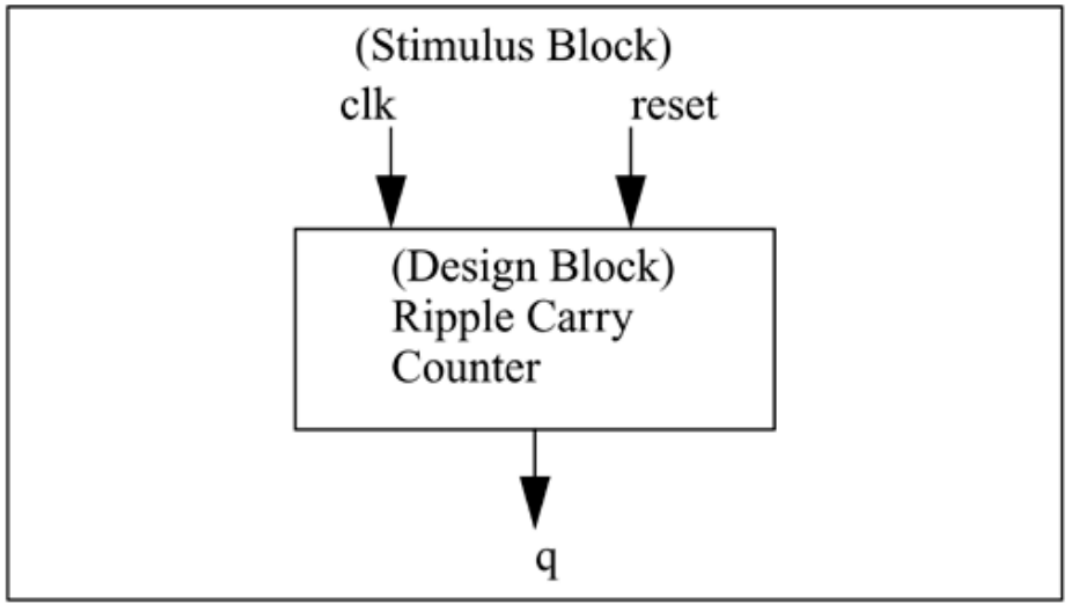
T-Flipflop Design

```
module T_FF(q, clk, reset);
output q;
input clk, reset;
wire d;
D_FF dff0(q, d, clk, reset);
not n1(d, q);
endmodule
```

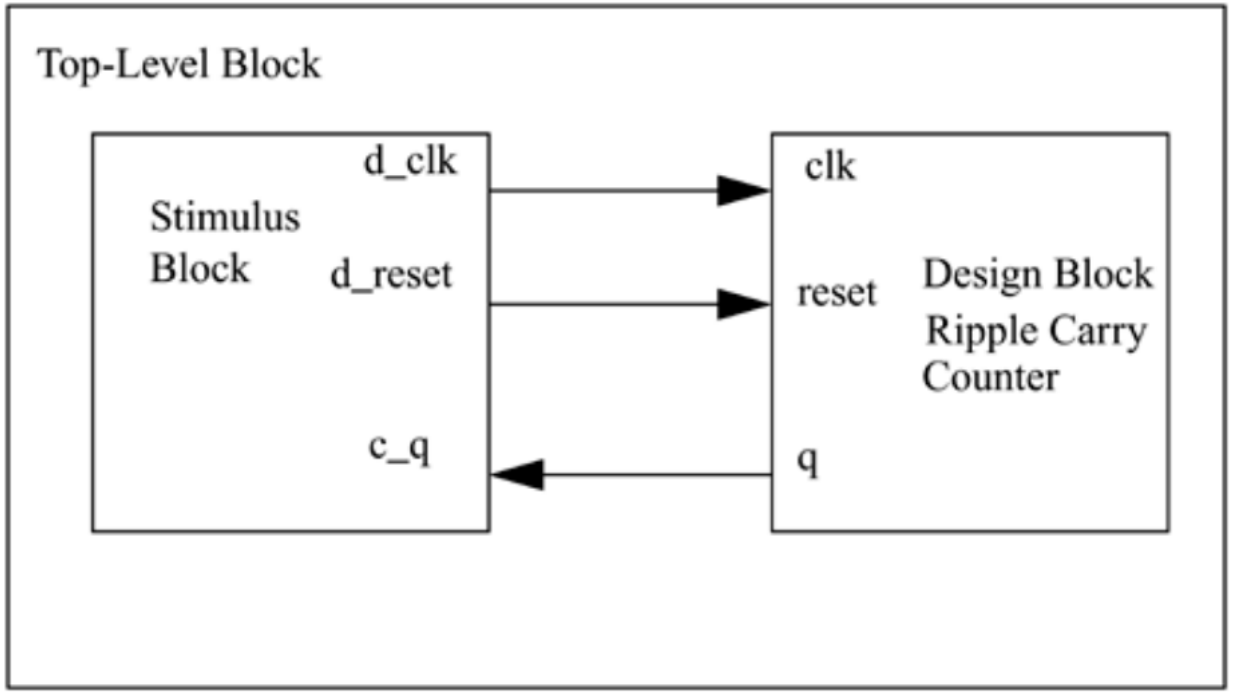
D-Flipflop Design

```
module D_FF(q, d, clk, reset);
output q;
input d, clk, reset;
reg q;
always @(posedge reset or negedge clk)
if (reset)
q <= 1'b0;
else
q <= d;
endmodule
```

Components of a Simulation



Stimulus Block Instantiates Design Block



Stimulus and Design Blocks Instantiated in a Dummy Top-Level Module

Example : Test Bench for RCC

Stimulus Block

```
module stimulus;
reg clk;
reg reset;
wire[3:0] q;
ripple_carry_counter r1(q, clk, reset);
initial
clk = 1'b0;
always
#5 clk = ~clk;
initial
begin
reset = 1'b1;
#15 reset = 1'b0;
#180 reset = 1'b1;
#10 reset = 1'b0;
#20 $finish;
end
initial
$monitor($time, " Output q = %d", q);
endmodule
```

Output of the Simulation

```
0 Output q = 0
20 Output q = 1
30 Output q = 2
40 Output q = 3
50 Output q = 4
60 Output q = 5
70 Output q = 6
80 Output q = 7
90 Output q = 8
100 Output q = 9
110 Output q = 10
120 Output q = 11
130 Output q = 12
140 Output q = 13
150 Output q = 14
160 Output q = 15
170 Output q = 0
180 Output q = 1
190 Output q = 2
195 Output q = 0
210 Output q = 1
220 Output q = 2
```

Summary

- Two kinds of design methodologies are used for digital design: top-down and bottom-up. A combination of these two methodologies is used in today's digital designs.
- Modules are the basic building blocks in Verilog. Modules are used in a design by instantiation. An instance of a module has a unique identity and is different from other instances of the same module.
- There are two distinct components in a simulation: a design block and a stimulus block.
  - *A design block contains the module definition and instantiation.*
  - *A stimulus block is used to test the design block. The stimulus block is usually the top-level block.*
  - *There are two different styles of applying stimulus to a design block*