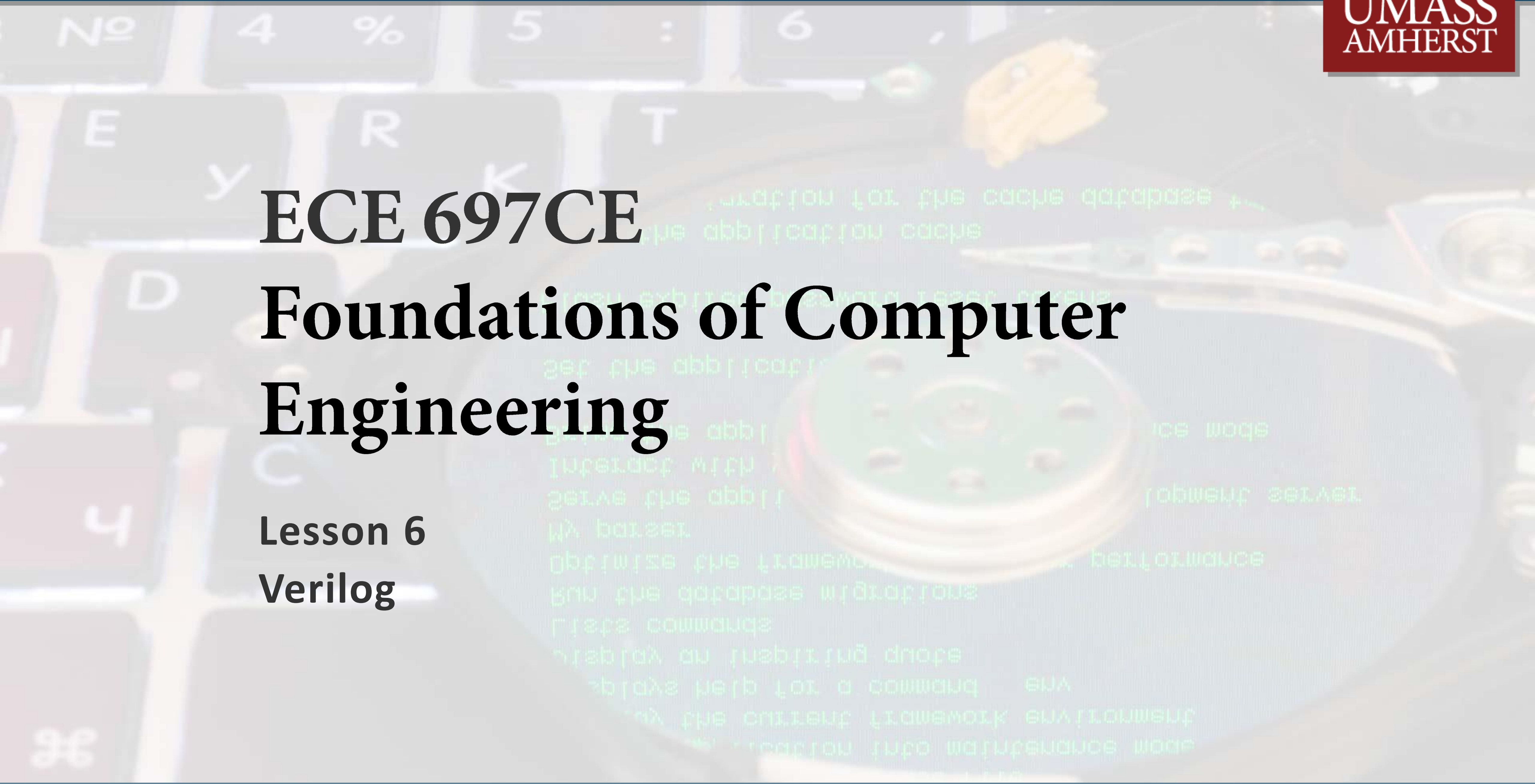
UMASS AMHERST



ECE 697CE Foundations of Computer Engineering

Lesson 6

Verilog

Rationale

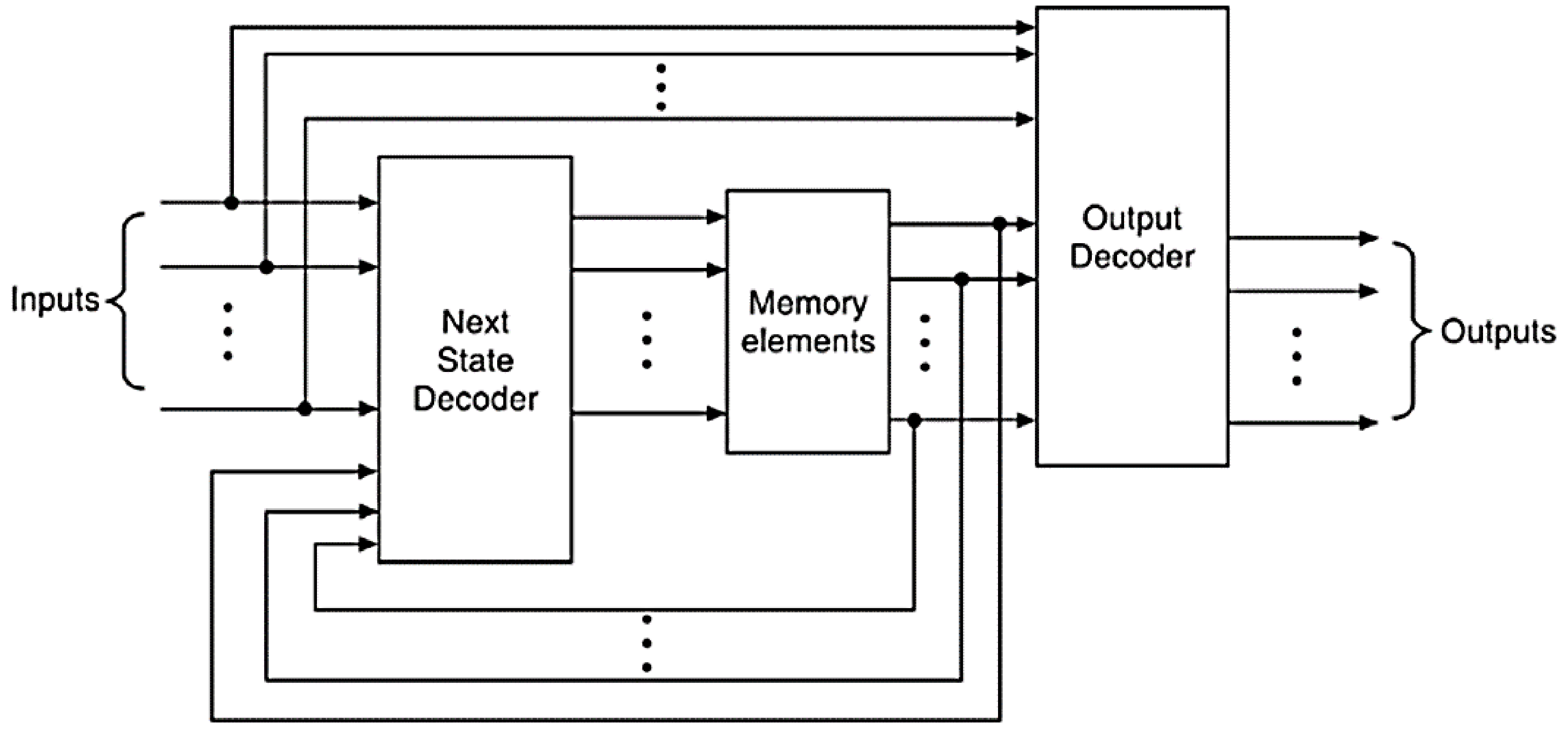
- Verilog specifies behavioral and structural definition of a digital system
- Verilog can describe exact contents of truth tables and datapath of the latest section

Objectives

- Demonstrate familiarity with hardware design language
- Discuss the role of modules and its instantiation within Verilog

Poll: Prior Knowledge

The following is an example of what type of finite state machine?



1) Mealy Machine
2) Moore Machine

Prior Knowledge

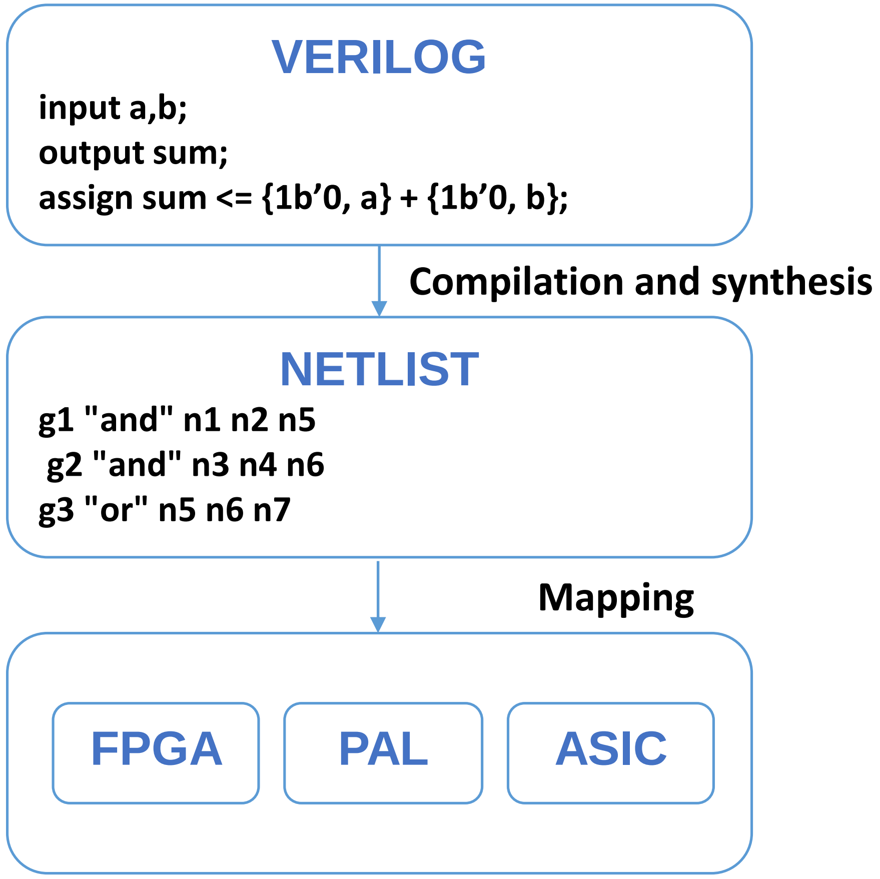
- Let’s review from Lesson 4 & 5: Finite State Machines
  - Sequential circuits
  - Mealy & Moore Machines
  - Synchronizing & Distinguishing Sequences

Orchestrated Discussion (Hand Raise): Critical Thinking Exercise

- Discuss 2 questions about Lesson 5

Introduction

- Hardware description language: Device independent representation of digital logic
- Describe hardware in
  - Behavioral level or algorithmic level
  - Data flow level
  - Gate level
  - Switch level
- HDL description compiled into a netlist
- Synthesis optimizes the logic
- Mapping targets a specific hardware platform

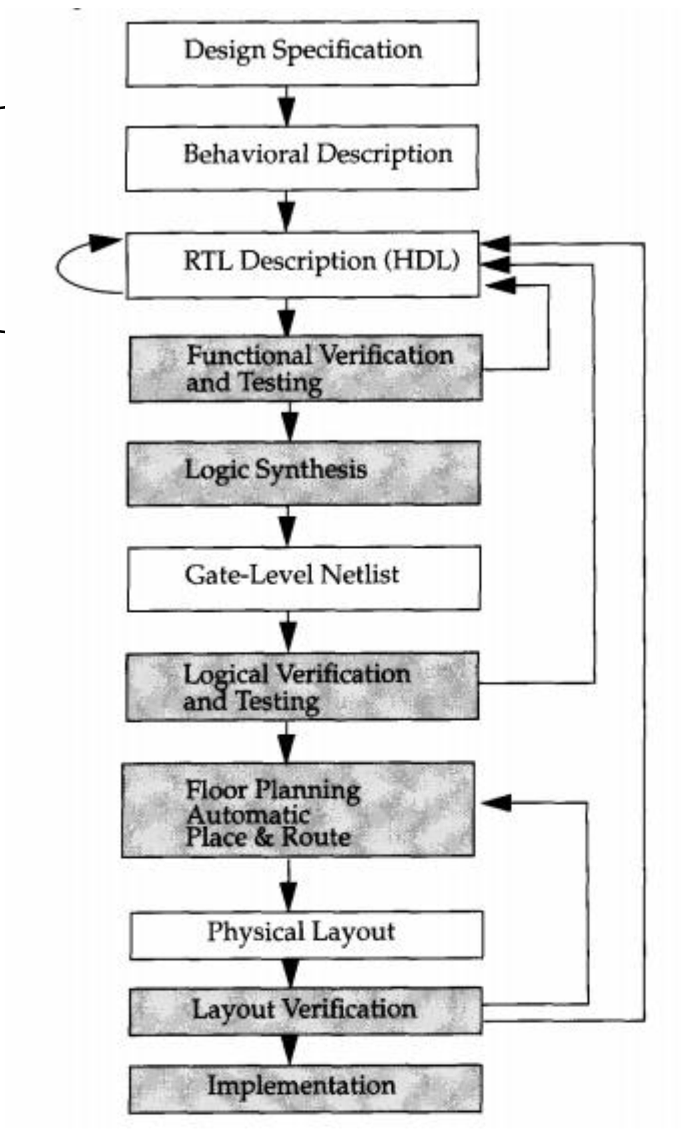


```
input a,b;
output sum;
assign sum <= {1b'0, a} + {1b'0, b};
```

```
g1 "and" n1 n2 n5
 g2 "and" n3 n4 n6
g3 "or" n5 n6 n7
```

Typical Design Flow

Using Hardware Description Languages like Verilog



Importance of HDLs

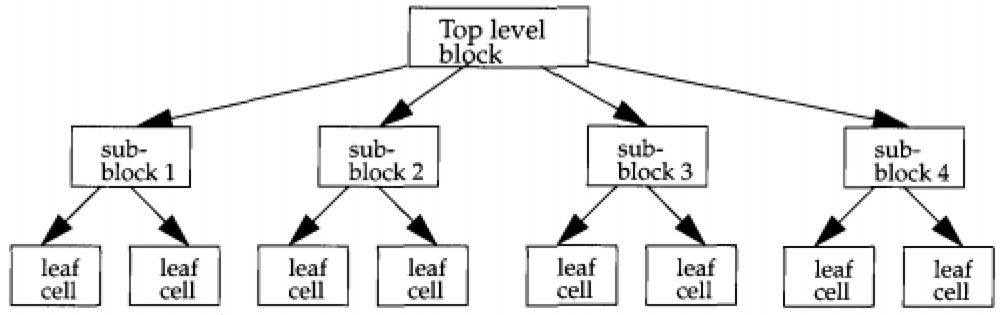
- Can write RTL description without choosing a specific fabrication technology
    - Logic synthesis tool convert the design to any fabrication technology
    - Need not redesign the circuit if a new technology emerges
- Functional verification of the design can be done at an early stage
    - Designers working at the RTL level can modify description till it meets desired functionality
    - Most design bugs can be eliminated
    - Cuts down design cycle time significantly
- Analogous to computer programming
- Concise representation of the design compared to gate level schematics

Popularity of Verilog

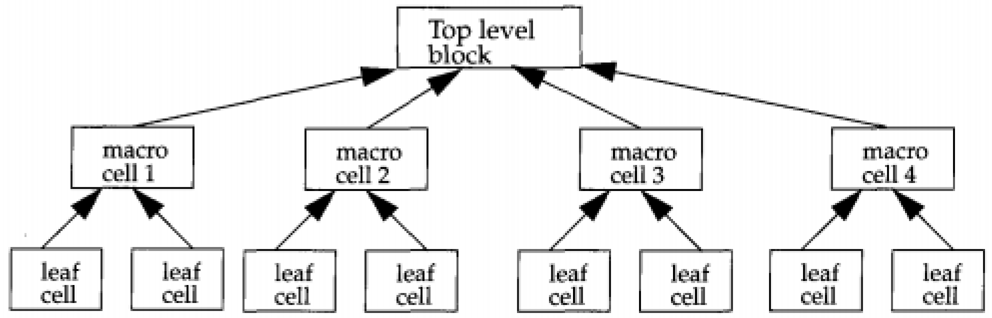
- Easy to learn and use
- Syntax similar to C programming language
- Allow different levels of abstraction to be mixed in the same model
- Only one language for stimulus and hierarchical design
- Most popular logic synthesis tools support Verilog HDL
- All fabrication vendors provide Verilog HDL libraries for post logic synthesis simulation.

Design Methodologies for Verilog

- Top down Design Methodology

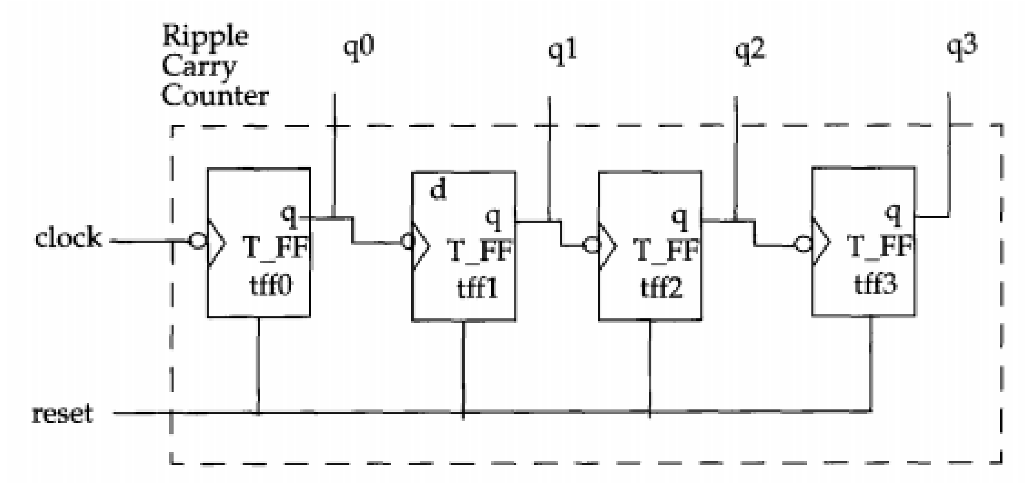


- Bottom up Design Methodology

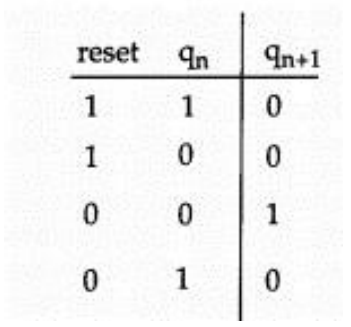


- Typically a combination of both is used

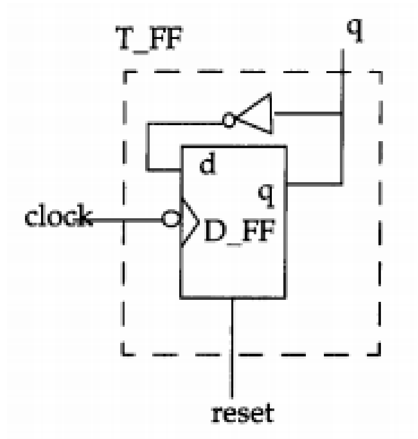
4 Bit Ripple Carry Counter



- **Made of negative edge triggered Flip flops**
- **T Flip flops can be made from negative edge triggered D Flip flops and inverters**

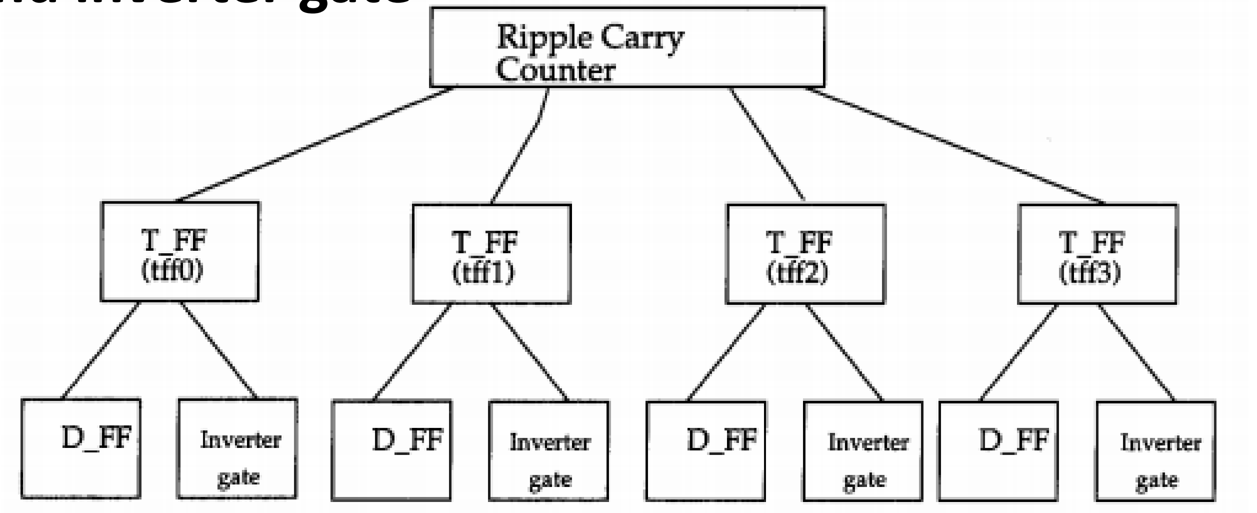


reset	q_n	q_{n+1}
1	1	0
1	0	0
0	0	1
0	1	0



Design Hierarchy for Ripple Carry Counter

- Top down design
  - Specify functionality of Ripple carry counter (Top level block)
  - Implement counter with T flip flops
  - Build the T flip flop from D flip flop and inverter gate

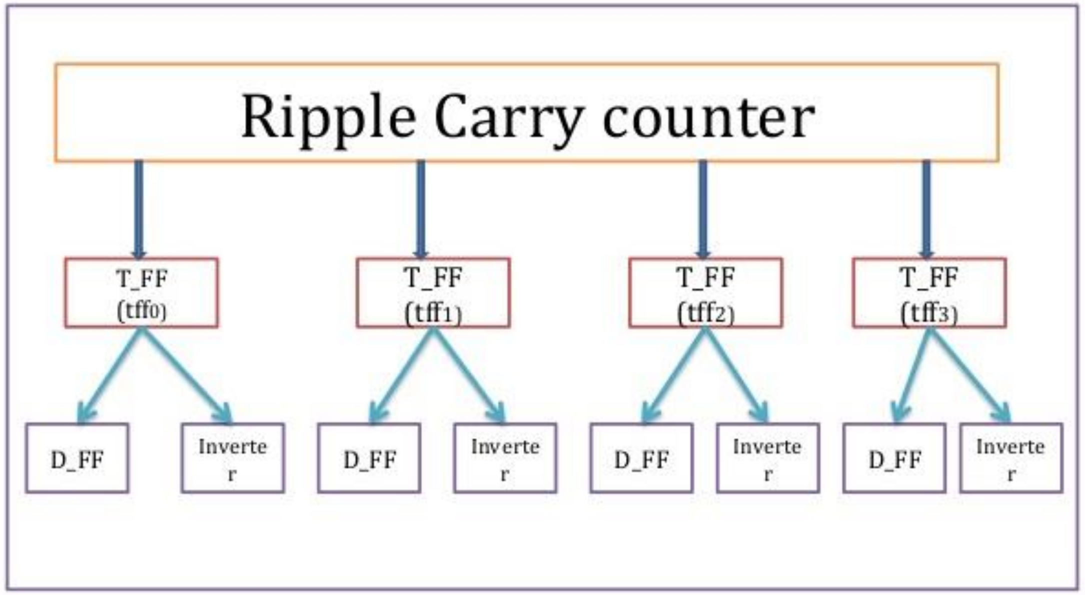


- Bottom up design
  - Flows in opposite direction
  - Build D Flipflops from <u>and, or</u> gates

- Bottom up flow meets the top down flow at the level of D Flip flop

Orchestrated Discussion (Hand Raise): Design Hierarchy

- Identify whether the following example represents a top down or a bottom up design methodology



Basic Building Block in Verilog

- **Module***:* Basic building block in Verilog
- Can be an element or a collection of lower level design blocks
  - Typically elements are grouped to modules and used at many places in design
- Provides necessary functionality to the higher level block using port interface
- Hides internal implementation from the higher level block
- Allows designer to modify module internals without affecting the rest of the design

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

Modules: Ripple Carry Counter

- T_FF, D_FF are examples of modules

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

```
module D_FF (q, d, clk, reset);
..
<functionality of D flip flop>
..
..
endmodule
```

Abstraction Levels of Verilog

- Internals of each module can be described at four levels of abstraction
- Module behavior is independent of the level of abstraction

<u>Register Transfer level (RTL)</u> :

- Combination of behavioral and dataflow constructs
- Code that can be synthesized i.e. acceptable to logic synthesis tools

Abstraction Levels of Verilog

<u>Behavioral level or algorithmic level</u>

- Highest level of abstraction
- No regard to hardware implementation details
- Design at this level : similar to C programming

<u>Data flow level</u>

- Specify data transfer between hardware registers
- Specify how data is processed in the design

Abstraction Levels of Verilog

Gate level

- Described by logical gates and interconnections
- Similar to design in terms of gate level logic diagram

Switch level

- Lowest level of abstraction
- Implementation in terms of switches, nodes and interconnections

Whiteboard: Abstraction Levels of Verilog

Correctly organize the different levels of abstraction from highest to lowest

Switch

Gate

Behavioral

Data Flow

Instances of Modules

- Module provides templates from actual objects or instances can be created
- Process of creating objects from module template is called instantiation
- Module definition can incorporate copies of other modules by instantiating them
- Module definitions specify how the module will work
- Module must be instantiated for use in the design

Module Instances: Ripple Carry Adder

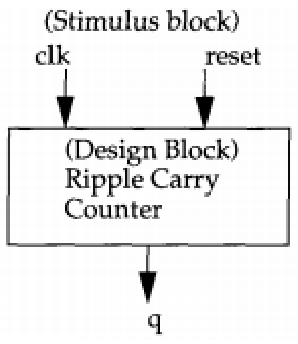
```
module ripple_carry_counter( q, clock, reset);
output [3:0] q;
input clk, reset;

T_FF tff0(q[0], clock, 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
```

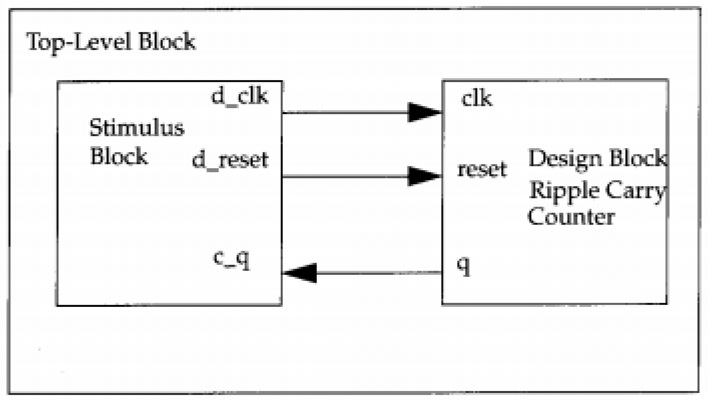
Instances of module T_FF

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

Components of Simulation

- Functionality of the design block can be tested by applying stimulus and checking results
- Design block tested using stimulus blocks (test bench)
- 2 styles of stimulus block application
  1) Stimulus block becomes the top level block
     - Stimulus block instantiates the design block
     - Directly drives the signals in the design block
  2) Instantiate and stimulus and design block in a top level dummy module
     - Stimulus block interacts with the design block through interface





Group Discussion and Report Back (Short Answer): Verilog Applications

- **What are some potential applications of Verilog in your personal or professional experiences?**

Poll: Application

In the following diagram, identify the missing component.

```
module D_FF (q, d, clk, reset);
..
<functionality of D ______>
..
..
endmodule
```

1. **“reset”**
2. **“clock”**
3. **“T”**
4. **“flip-flop”**

UMASS AMHERST

Summary of this Lesson

- Introduction to Verilog

To Prepare for the Next Lesson

- Read the Required Readings for Lesson 7.
- Complete the Pre-work for Lesson 7.
- Continue working on the Project.

Go to the online classroom for details.