# 



**Lecture 01: INTRODUCTION**

**PROF. INDRANIL SENGUPTA**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**Main Objectives of the Course**

**Hardware Modeling Using Verilog**

1. Learn about the Verilog hardware description language.
2. Understand the diﬀerence between behavioral and structural design styles.
3. Learn to write test benches and analyze simulation results.
4. Learn to model combinational and sequential circuits.
5. Distinguish between good and bad coding practices.
6. Case studies with some complex designs.

Hardware Modeling Using Verilog

2

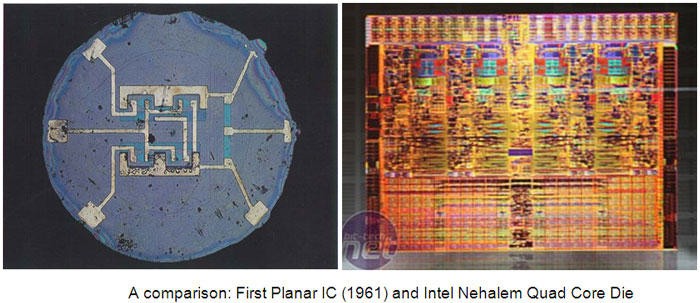


**VLSI Design Process**

* Design complexity increasing rapidly
  + Increased size and complexity
  + Fabrication technology improving
  + CAD tools are essential
  + Conﬂicting requirements like area, speed, and energy consumption
* The present trend
  + Standardize the design ﬂow
  + Emphasis on low-power design, and increased performance

Hardware Modeling Using Verilog

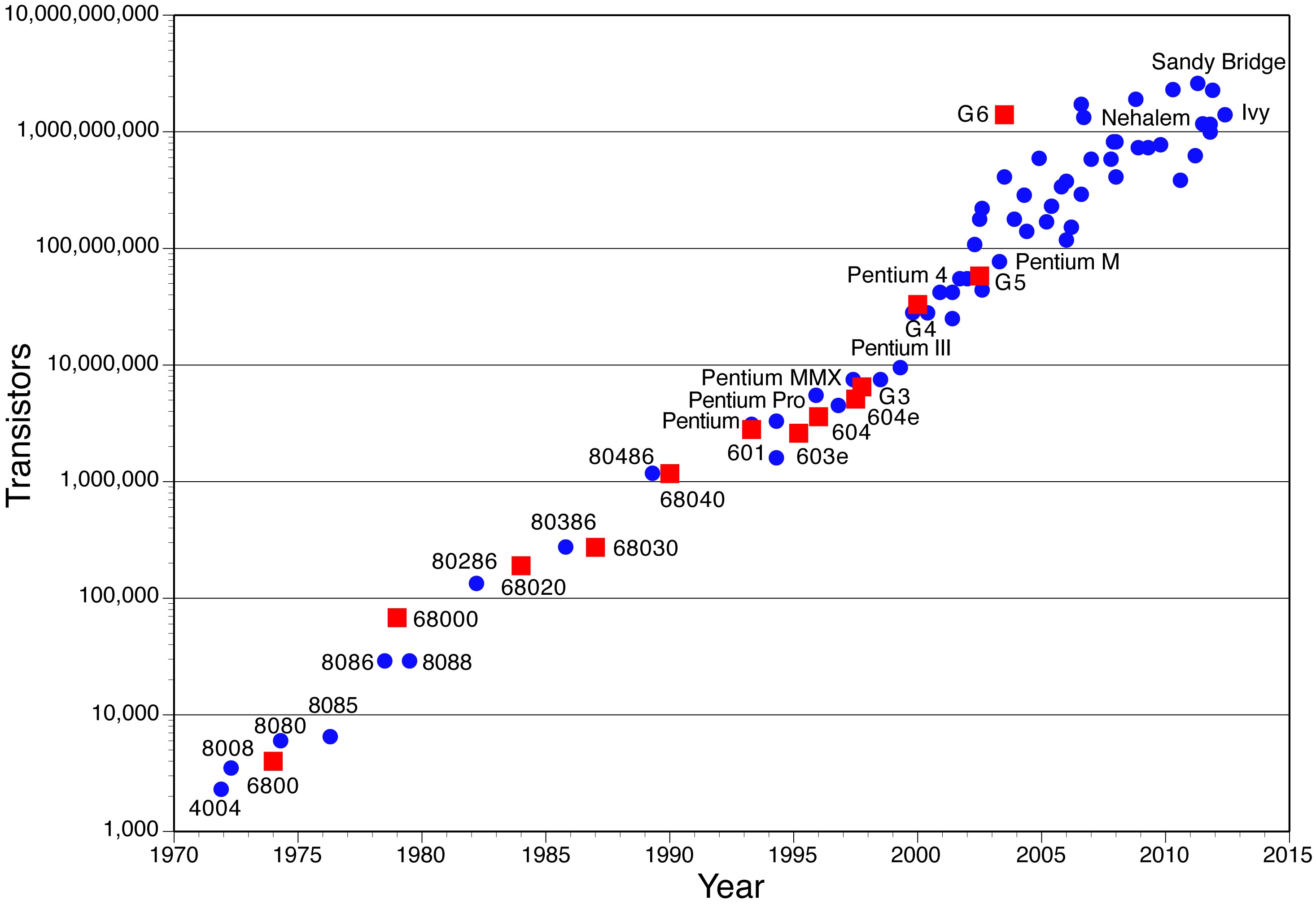
3



First Planar IC (1961) and Intel Nehalem Quad Core Die

Hardware Modeling Using Verilog

4

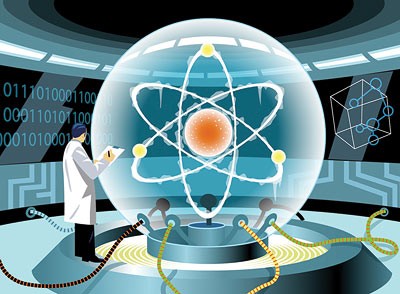
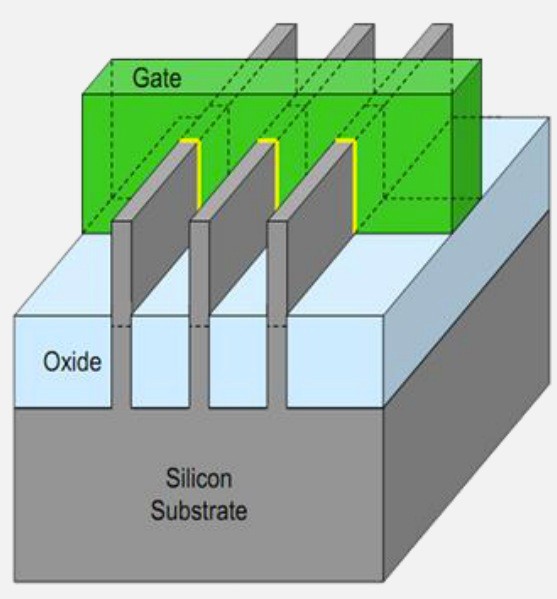
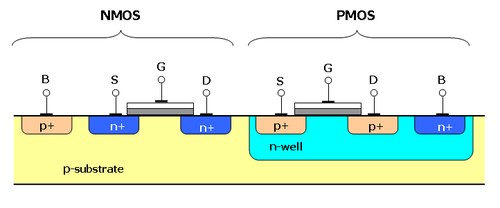


**Moore’s Law**

* Exponential growth
* Design complexity increases rapidly
* Automated tools are essential
* Must follow well- deﬁned design ﬂow

Hardware Modeling Using Verilog

5



**CMOS**

**(up to 22nm)**

**FinFET (14nm)**

**QUANTUM?**

Hardware Modeling Using Verilog

6

***binit***

*2018-08-02 08:52:17*

--------------------------------------------

We just say what we want, we don't specify how it is doing it.



**VLSI Design Flow**

* Standardized design procedure
  + Starting from the design idea down to the actual implementation.
* Encompasses many steps:
  + Speciﬁcation
  + Synthesis
  + Simulation
  + Layout
  + Testability analysis
  + and many more ……

Hardware Modeling Using Verilog

7



* Need to use Computer Aided Design (CAD) tools.
  + Based on Hardware Description Language (HDL).
  + HDLs provide formats for representing the outputs of various design steps.
  + A CAD tool transforms its HDL input into a HDL output that contains more detailed information about the hardware.
    - Behavioral level to register transfer level
    - Register transfer level to gate level
    - Gate level to transistor level
    - Transistor level to the layout level

Hardware Modeling Using Verilog

8



**Two Competing HDLs**

1. Verilog
2. VHDL

*Designs are created typically using HDLs, which get transformed from one level of abstraction to the next as the design ﬂow progresses.*

There are other HDLs like SystemC, SystemVerilog, and many more …

Hardware Modeling Using Verilog

9



**Simplistic View of Design Flow**

Design Idea

Flow Graph, Pseudo Code Bus/Register Structure Gate/F-F Netlist

Transistor Layout

Chip / Board

Hardware Modeling Using Verilog

10

Manufacturing

Physical Design

Logic Design

Data Path Design

Behavioral Design



**Steps in the Design Flow**

* Behavioral design
  + Specify the functionality of the design in terms of its *behavior*.
  + Various ways of specifying:
    - Boolean expression or truth table.
    - Finite-state machine behavior (e.g. state transition diagram or table).
    - In the form of a high-level algorithm.
  + Needs to be synthesized into more detailed speciﬁcations for hardware realization.

Hardware Modeling Using Verilog

11



* Data path design
  + Generate a netlist of register transfer level components, like registers, adders, multipliers, multiplexers, decoders, etc.
  + A *netlist* is a directed graph, where the vertices indicate components, and the edges indicate interconnections.
  + A netlist speciﬁcation is also referred to as *structural design*.
    - Netlist may be speciﬁed at various levels, where the components may be functional modules, gates or transistors.
    - Systematically transformed from one level to the next.

Hardware Modeling Using Verilog

12



* Logic design
  + Generate a netlist of gates/ﬂip-ﬂops or standard cells.
  + A standard cell is a pre-designed circuit module (like gates, ﬂip-ﬂops, multiplexer, etc.) at the layout level.
  + Various logic optimization techniques are used to obtain a cost eﬀective design.
  + There may be conﬂicting requirements during optimization:
    - Minimize number of gates.
    - Minimize number of gate levels (i.e. delay).
    - Minimize signal transition activities (i.e. dynamic power).

Hardware Modeling Using Verilog

13



* Physical design and Manufacturing
  + Generate the ﬁnal layout that can be sent for fabrication.
  + The layout contains a large number of regular geometric shapes corresponding to the diﬀerent fabrication layers.
  + Alternatively, the ﬁnal target may be Field Programmable Gate Array (FPGA), where technology mapping from the gate level netlist is used.
    - Can be programmed in-ﬁeld.
    - Much greater ﬂexibility, but less speed.

Hardware Modeling Using Verilog

14



**Other Steps in the Design Flow**

* Simulation for veriﬁcation
  + At various levels: logic level, switch level, circuit level
* Formal veriﬁcation
  + Used to verify the designs through formal techniques
* Testability analysis and Test pattern generation
  + Required for testing the manufactured devices

Hardware Modeling Using Verilog

15



**END OF LECTURE 01**

Hardware Modeling Using Verilog

16



**Lecture 02: DESIGN REPRESENTATION**

**PROF. INDRANIL SENGUPTA**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**Design Representation**

* A design can be represented at various levels from three diﬀerent points of view:
  1. Behavioral
  2. Structural
  3. Physical
* Can be conveniently expressed by Y-diagram.

Hardware Modeling Using Verilog

18



BEHAVIORAL DOMAIN

Programs Specifications

Truth table

STRUCTURAL DOMAIN

Gates Adders Registers

Transistors / Layouts

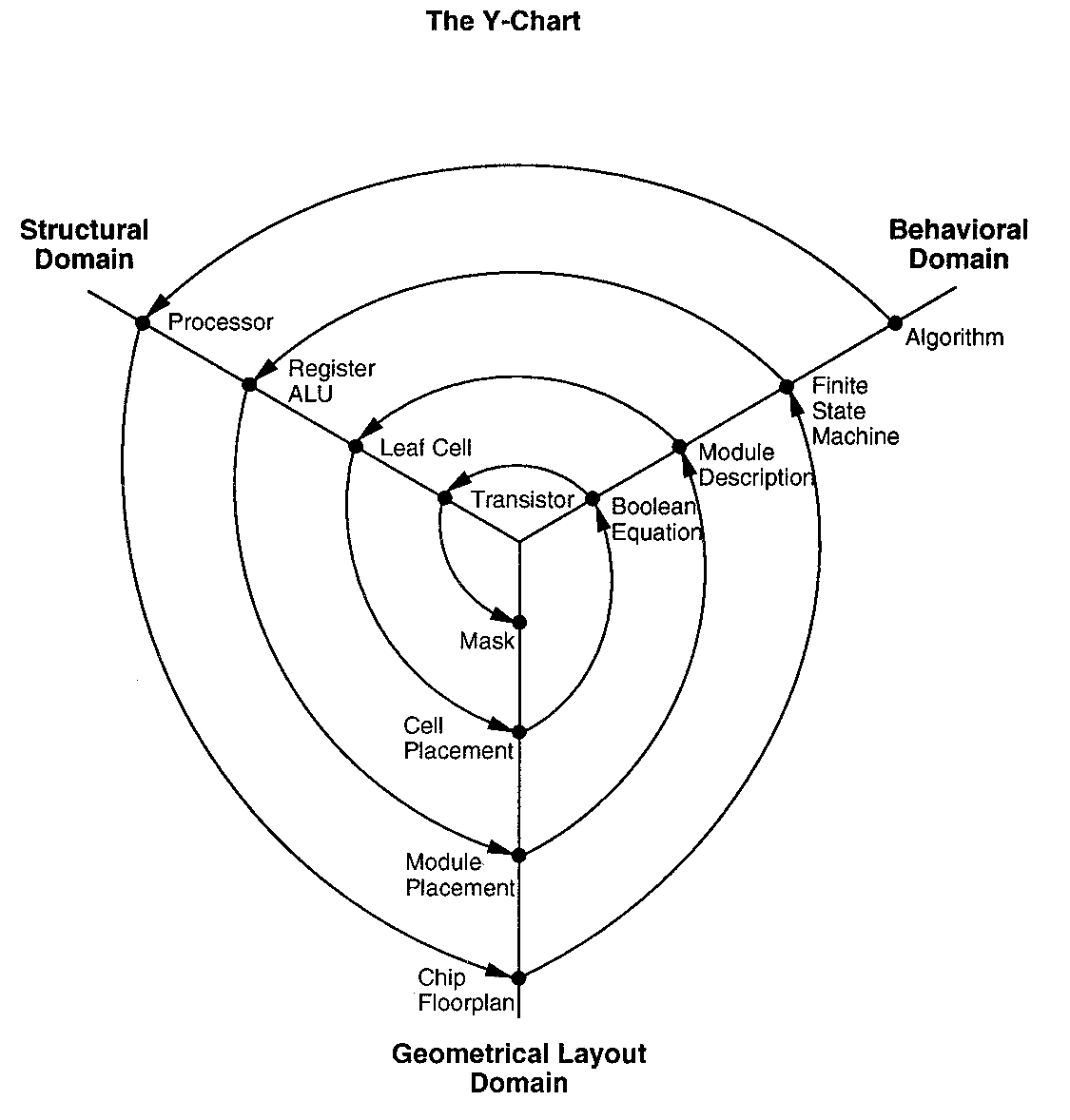
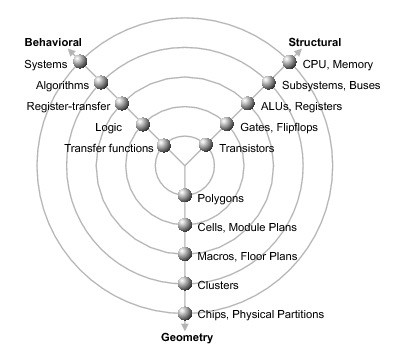
PHYSICAL Cells

DOMAIN

Chips / Boards

Hardware Modeling Using Verilog

19



Hardware Modeling Using Verilog

20



**Behavioral Representation**

* Speciﬁes how a particular design should respond to a given set of inputs.
* May be speciﬁed by:
  + Boolean equations
  + Tables of input and output values
  + Algorithms written in standard HLL like C
  + Algorithms written in special HDL like Verilog or VHDL

Hardware Modeling Using Verilog

21



**Behavioral Representation :: Example**

Full Adder:

* two operand inputs A and B
* a carry input C
* a carry output Cy
* a sum output S

A B C

S

Cy

* Express in terms of Boolean expressions:

S = A.Bʹ.Cʹ + Aʹ.Bʹ.C + Aʹ.B.Cʹ + A.B.C = A  B  C Cy = A.B + A.C + B.C

Hardware Modeling Using Verilog

22

FA



* Express in Verilog in terms of Boolean expressions

Hardware Modeling Using Verilog

23

**module carry (S, Cy, A, B, C); input A, B, C;**

**output S, Cy;**

**assign S = A ^ B ^ C;**

**assign Cy = (A & B) | (B & C) | (C & A); endmodule**



* Express in Verilog in terms of truth table (only Cy is shown)

Hardware Modeling Using Verilog

24

**primitive carry (Cy, A, B, C); input A, B, C;**

**output Cy; table**

**// A B C Cy**

**1 1 ? : 1 ;**

**1 ? 1 : 1 ;**

**? 1 1 : 1 ;**

**0 0 ? : 0 ;**

**0 ? 0 : 0 ;**

**? 0 0 : 0 ;**

**endtable endprimitive**

**cy\_in, x, y);**



**carry = A.B + B.C + C.A**

**sum = A**  **B**  **C**

* We instantiate carry and sum circuits to create a full adder.
* We instantiate four full adders to create the 4-bit adder.

Hardware Modeling Using Verilog

28

sum

carry

sum

carry

sum

carry

sum

carry

add

add

add

add

add4



**Structural Representation**

* Speciﬁes how components are interconnected.
* In general, the description is a list of modules and their interconnection.
  + Called *netlist*.
  + Can be speciﬁed at various levels.

Hardware Modeling Using Verilog

25



* At the structural level, the levels of abstraction are:
  + The module (functional) level
  + The gate level
  + The transistor level
  + Any combination of above
* In each successive level more detail is revealed about the implementation.

Hardware Modeling Using Verilog

26



**Example: A 4-bit Ripple Carry Adder**

A3 B2

A2 B2 A1 B1

A0 B0

C

C3

C2

C1

C0

4

S3

S2

S1

S0

* Consists of four full adders.
* Each full adder consists of a sum circuit and a carry circuit.

Hardware Modeling Using Verilog

27

FA0

FA1

FA2

FA2



**module add (cy\_out, sum, a, b, cy\_in); input a, b, cy\_in;**

**module add4 (s, cy4, input [3:0] x, y; input cy\_in;**

**\_in);**

Hardware Modeling Using Verilog

29

**output [3:0] s;**

**output cy4;**

**wire [2:0] cy\_out;**

**add B0 (cy\_out[0], s[0], x[0], y[0], ci);**

**add B1 (cy\_out[1], s[1], x[1], y[1], cy\_out[0]);**

**add B2 (cy\_out[2], s[2], x[2], y[2], cy\_out[1]);**

**add B3 (cy4, s[3], x[3], y[3], cy\_out[2]); endmodule**

**output sum, cy\_out;**

**sum s1 (sum, a, b, cy\_in); carry c1 (cy\_out, a, b, cy**

**endmodule**



Hardware Modeling Using Verilog

30

**module sum (sum, a, b, cy\_in);**

**input a, b, cy\_in; output sum;**

**wire t;**

**xor x1 (t, a, b);**

**xor x2 (sum, t, cy\_in); endmodule**

**module carry (cy\_out, a, b, cy\_in); input a, b, cy\_in;**

**output cy\_out; wire t1, t2, t3;**

**and g1 (t1, a, b);**

**and g2 (t2, a, c);**

**and g3 (t3, b, c);**

**or g4 (cy\_out, t1, t2, t3); endmodule**



**Physical Representation**

* The lowest level of physical speciﬁcation.

– Photo-mask information required by the various processing steps in the fabrication process.

* At the module level, the physical layout for the 4-bit adder may be deﬁned by a rectangle or polygon, and a collection of ports.
* At the layout level, there can be a large number of rectangles or polygons.

Hardware Modeling Using Verilog

31



* Partial physical description for 4-bit adder in Verilog

Hardware Modeling Using Verilog

32

**module add4;**

**input x[3:0], y[3:0], cy\_in; output s[3:0], cy4;**

**boundary [0, 0, 130, 500];**

**port x[0] aluminum width = 1 origin = [0, 35]; port y[0] aluminum width = 1 origin = [0, 85]; port cy\_in polysilicon width = 2 origin = [70, 0]; port s[0] aluminum width = 1 origin = [120, 65];**

**add a0 origin = [0, 0];**

**add a1 origin = [0, 120]; endmodule**



**Digital IC Design Flow: A quick look**

Logical design (front-end CAD)

Physical design (back-end CAD)

Hardware Modeling Using Verilog

33

Routing

Circuit Extraction

Placement

Floorplanning

Post-layout Simulation

Partitioning

Logic Synthesis

Design Entry

Pre-layout Simulation



**END OF LECTURE 02**

Hardware Modeling Using Verilog

34



**Lecture 03: GETTING STARTED WITH VERILOG**

**PROF. INDRANIL SENGUPTA**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**Why do we use Verilog?**

* To describe a digital system as a set of *modules*.
  + Each of the modules will have an interface to other modules, in addition to its description.
  + Two ways to specify a module:
    1. By specifying its internal logical structure (called *structural representation*).
    2. By describing its behavior in a program-like manner (called *behavioral representation*).
  + The modules are interconnected using *nets*, which allow them to work with each other.

Hardware Modeling Using Verilog

36



Evaluate Result

Hardware Modeling Using Verilog

40

FPGA

ASIC

SYNTHESIS

SIMULATE

Verilog Module(s)

Test Bench

Verilog Module(s)



**What next?**

* Aier specifying the system in Verilog, we can do two things:
  1. Simulate the system and verify the operation.
     + Just like running a program written in some high-level language.
     + Requires a *test bench* or *test harness*, that speciﬁes the inputs that are to be applied and the way the outputs are to be displayed.
  2. Use a synthesis tool to map it to hardware.
     + Converts it to a netlist of low-level primitives.
     + The hardware can be *Application Speciﬁc Integrated Circuit* (ASIC).
     + Or else, it can be *Field Programmable Gate Array* (FPGA).

Hardware Modeling Using Verilog

37



* When the design is mapped to hardware, we do not need test bench for simulation any more.
* Signals can be actually applied from some source (e.g. signal generator), and response evaluated by some equipment (e.g. oscilloscope or logic analyzer).

Hardware Modeling Using Verilog

38



* Using ASIC as hardware target?
  + When high performance and high packing density is required.
  + When the manufactured hardware is expected to be used in large numbers (e.g. processor chips).
* Using FPGA as hardware target?
  + When fast turnaround time is required to validate the design.
  + The mapping can be done in the laboratory itself with a FPGA kit and associated soiware.
  + There is a tradeoﬀ in performance.

Hardware Modeling Using Verilog

39



**How to Simulate?**

* Free soiware available:
  + Icarus Verilog (http://iverilog.icarus.com)
  + GTKWave (http://gtkwave.sourceforge.net)
* Commercial soiware (with free versions available):
  + From Xilinx (ISE, Vivado)
  + Many more.

Hardware Modeling Using Verilog

41



**How to Synthesize?**

* For FPGA as target, speciﬁc soiware is required:
  + Xilinx ISE or VIVADO for Xilinx FPGA kits.
  + Similar soiware available from other FPGA vendors.
* For ASIC as target, commercial CAD tools exist:
  + Tool suite from Cadence.
  + Tool suite from Synopsys.
  + Several others …

Hardware Modeling Using Verilog

42

***binit***

*2018-08-04 10:31:13*

--------------------------------------------

Monitor prints only when there is change in any variable.



**Scope of this Course**

* We shall be discussing features of the Verilog language, and verifying the design through simulation.
  + How to design combinational and sequential digital circuits?
  + How to verify the functionality through simulation?
* We shall not be discussing the synthesis tools; however:
  + Shall discuss various tricks that help in synthesis.
  + Shall discuss the common design ﬂow – ﬁrst code the modules in behavioral design style, and then translate selected subset of modules to structural design style.

Hardware Modeling Using Verilog

43



**How to Simulate Verilog Module(s)**

* Using a test bench to verify the functionality of a design coded in Verilog (called Design-under-Test or DUT), comprising of:
  + A set of stimulus for the DUT.
  + A monitor, which captures or analyzes the outputs of the DUT.
* Requirement:
  + The inputs of the DUT need to be connected to the test bench.
  + The outputs of the DUT needs also to be connected to the test bench.

Hardware Modeling Using Verilog

44



Stimulus

Monitor

Hardware Modeling Using Verilog

45

Design Under Test (DUT)

TEST BENCH



**An Example**

A G1

B

t1

C B’ D

G2

t2

G4

Y

E G3

F

t3

We can combine declarations of same type of gate together:

Hardware Modeling Using Verilog

46

**nand #1 G1 (t1,A,B),**

**G4 (Y,t1,t2,t3);**

**module example (A,B,C,D,E,F,Y); input A,B,C,D,E,F;**

**output Y;**

**wire t1, t2, t3, Y;**

**nand #1 G1 (t1,A,B); and #2 G2 (t2,C,~B,D);**

**nor #1 G3 (t3,E,F);**

**nand #1 G4 (Y,t1,t2,t3); endmodule**



**example.v**

**example-test.v**

Hardware Modeling Using Verilog

47

**module example**

**(A,B,C,D,E,F,Y);**

**wire t1, t2, t3, Y;**

**nand #1 G1 (t1,A,B); and #2 G2 (t2,C,~B,D);**

**nor #1 G3 (t3,E,F);**

**nand #1 G4 (Y,t1,t2,t3); endmodule**

**module testbench;**

**reg A,B,C,D,E,F; wire Y;**

**example DUT(A,B,C,D,E,F,Y);**

**initial begin**

**$monitor ($time,” A=%b, B=%b, C=%b,**

**D=%b, E=%b, F=%b, Y=%b”, A,B,C,D,E,F,Y);**

**#5 A=1; B=0; C=0; D=1; E=0; F=0; #5 A=0; B=0; C=1; D=1; E=0; F=0; #5 A=1; C=0;**

**#5 F=1;**

**#5 $finish; end**

**endmodule**



* Simulation results:

**Command in iVerilog:**

1. iverilog -o mysim example.v example-test.v
2. vvp mysim

Hardware Modeling Using Verilog

48

**0 A=x, B=x, C=x, D=x, E=x, F=x, Y=x 5 A=1, B=0, C=0, D=1, E=0, F=0, Y=x 8 A=1, B=0, C=0, D=1, E=0, F=0, Y=1 10 A=0, B=0, C=1, D=1, E=0, F=0, Y=1 13 A=0, B=0, C=1, D=1, E=0, F=0, Y=0 15 A=1, B=0, C=0, D=1, E=0, F=0, Y=0 18 A=1, B=0, C=0, D=1, E=0, F=0, Y=1 20 A=1, B=0, C=0, D=1, E=0, F=1, Y=1**

Hardware Modeling Using Verilog 50



**To display the waveforms**

Run the command:

gtkwave example.vcd

Hardware Modeling Using Verilog

49

**module testbench;**

**reg A,B,C,D,E,F; wire Y;**

**example DUT(A,B,C,D,E,F,Y);**

**initial begin**

**$dumpfile (“example.vcd”);**

**$dumpvars (0,testbench);**

**$monitor ($time,” A=%b, B=%b, C=%b, D=%b, E=%b, F=%b, Y=%b”, A,B,C,D,E,F,Y);**

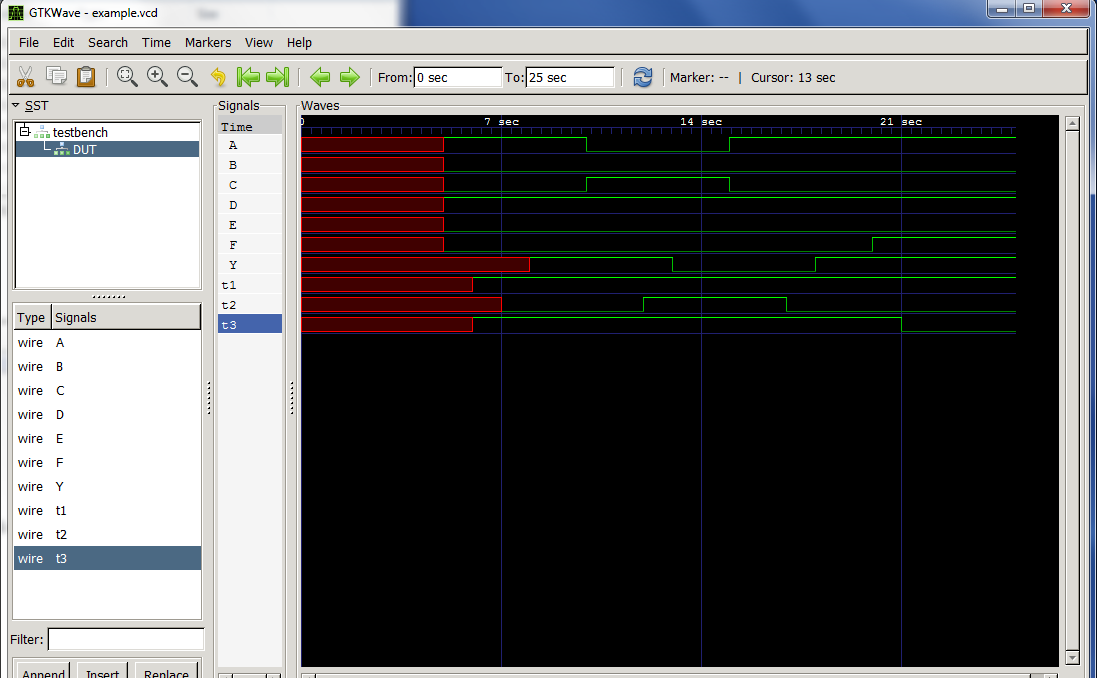
**#5 A=1; B=0; C=0; D=1; E=0; F=0;**

**#5 A=0; B=0; C=1; D=1; E=0; F=0; #5 A=1; C=0;**

**#5 F=1;**

**#5 $finish; end**

**endmodule**



**END OF LECTURE 03**

Hardware Modeling Using Verilog

51



**Lecture 04: VLSI DESIGN STYLES (PART 1)**

**PROF. INDRANIL SENGUPTA**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**VLSI Design Cycle**

* Large number of devices
* Optimization requirements for high performance
* Time-to-market competition
* Cost

Manual

Automation

53

Chip

System Specifications



**VLSI Design Cycle (contd.)**

1. System speciﬁcation
2. Functional design
3. Logic design
4. Circuit design
5. Physical design
6. Design veriﬁcation
7. Fabrication
8. Packaging, testing, and debugging

Hardware Modeling Using Verilog

54



**Physical Design**

* Converts a circuit description into a geometric description.

– This description is used for fabrication of the chip.

* Basic steps in the physical design cycle:
  1. Partitioning, ﬂoorplanning and placement
  2. Routing
  3. Static timing analysis
  4. Signal integrity and crosstalk analysis
  5. Physical veriﬁcation and signoﬀ

Hardware Modeling Using Verilog

55



**Various Design Styles**

* Programmable Logic Devices
  + Field Programmable Gate Array (FPGA)
  + Gate Array
* Standard Cell (Semi-Custom Design)
* Full-Custom Design

Hardware Modeling Using Verilog

56



**Which Design Style to Use?**

* Basically a tradeoﬀ among several design parameters.
  + Hardware cost
  + Circuit delay
  + Time required
* Optimizing on these parameters is oien conﬂicting.

Hardware Modeling Using Verilog

57



**Field Programmable Gate Array (FPGA)**



**What does FPGA oﬀer?**

* User / Field Programmability.
  + Array of logic cells connected via routing channels.
  + Diﬀerent types of cells:
    - Special I/O cells.
    - Logic cells (Mainly lookup tables (LUT) with associated registers).
  + Interconnection between cells:
    - Using SRAM based switches.
    - Using anti-fuse elements.

Hardware Modeling Using Verilog

59



**Ease of Use**

* FPGA chips are manufactured by a number of vendors:
  + Xilinx, Altera, Actel, etc.
  + Products vary widely in capability.
* FPGA development boards and CAD soiware available from many sellers.
  + Allows rapid prototyping in laboratory.

Hardware Modeling Using Verilog

60

### Xilinx XC4000 Architecture

**XC4000E Conﬁgurable Logic Block**

CLB CLB

Switch

Matrix

CLB

CLB

Programmable

Interconnect I/O Blocks (IOBs)

Configurable Logic Blocks (CLBs)

Hardware Modeling Using Verilog

61 Hardware Modeling Using Verilog 62

**CLB Functionalities**

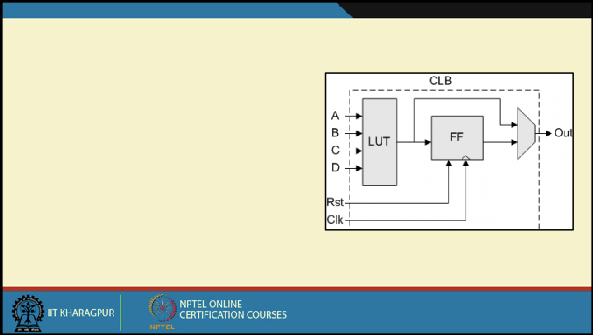
#### Two 4-input function generators

* + Implemented using Lookup Tables using 16x1 RAM.
  + Can also implement 16x1 memory.

#### Two 1-bit registers

* + Each can be conﬁgured as ﬂip-ﬂop or latch.
  + Independent clock polarity.
  + Synchronous and asynchronous Set / Reset.

## Look Up Tables (LUT)

* Combinatorial Logic is stored in 16x1 SRAM Look Up Tables (LUTs) in a CLB.
* Capacity is limited by number of inputs, not complexity.
* Choose to use each function generator as 4-input logic (LUT) or as high-speed RAM.

Hardware Modeling Using Verilog

63 Hardware Modeling Using Verilog 64

## LUT Mapping: An Example

* A function: *f = A’.B + B’.C.D*

#### The mapping process:

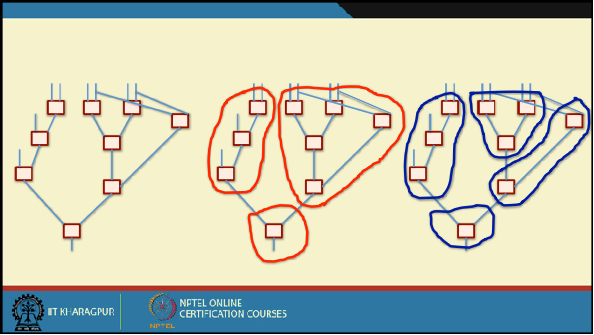
* + Create the truth table of the 4-variable function.
  + Load the output column into the SRAM corresponding to the LUT.
  + Apply the function inputs to the LUT inputs.

#### Any 4-variable function can be realized.

* + Netlist to LUT mapping is an interesting design tradeoﬀ.

Given netlist

## Area - Delay Tradeoﬀ

3 LUTs

Delay = 2

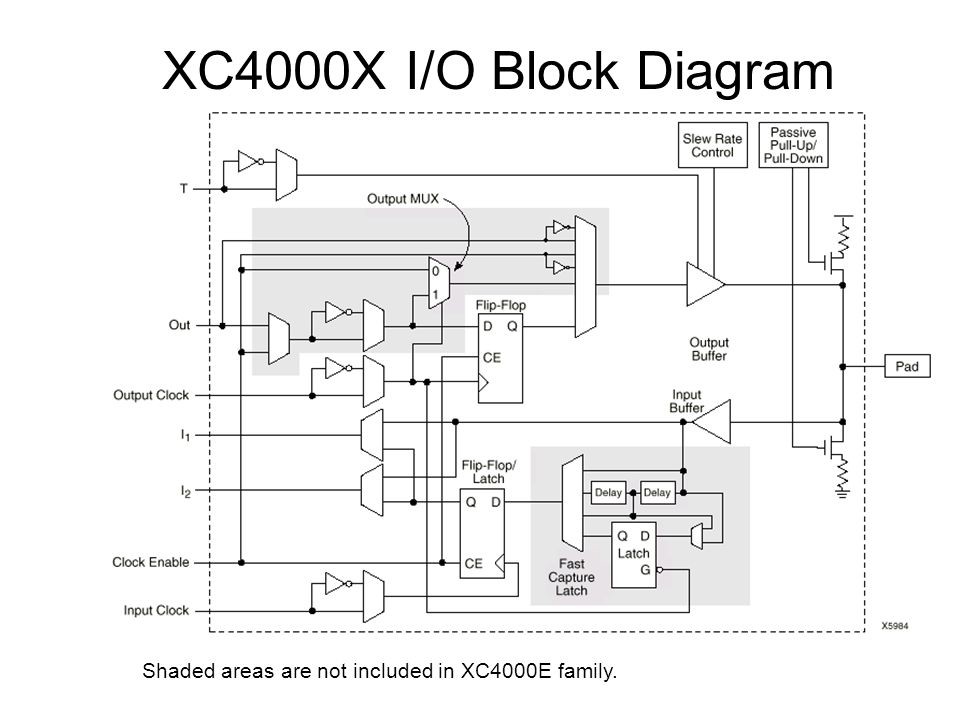
4 LUTs

Delay = 3

Hardware Modeling Using Verilog

65 Hardware Modeling Using Verilog 66

# 



**XC4000X I/O**

**Block Diagram**

Hardware Modeling Using Verilog

67



**Xilinx FPGA Routing**

1. Fast Direct Interconnect – CLB to CLB
2. General Purpose Interconnect

– Uses switch matrix

Hardware Modeling Using Verilog

68

CLB

CLB

Switch Matrix

Switch Matrix

CLB

CLB



**FPGA Design Flow**

* Design Entry
  + In schematic, VHDL, or Verilog.
* Implementation
  + Placement & Routing
  + Bitstream generation
  + Analyze timing, view layout, simulation, etc.
* Download
  + Directly to Xilinx hardware devices with unlimited reconﬁgurations.

Hardware Modeling Using Verilog

69



**END OF LECTURE 04**

Hardware Modeling Using Verilog

70



**Lecture 05: VLSI DESIGN STYLES (PART 2)**

**PROF. INDRANIL SENGUPTA**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**Gate Array**



**Introduction**

* In view of the speed of prototyping capability, the gate array (GA) comes aier the FPGA.
* Design implementation of
  + FPGA chip is done with user programming,
  + Gate array is done with metal mask design and processing.

Hardware Modeling Using Verilog

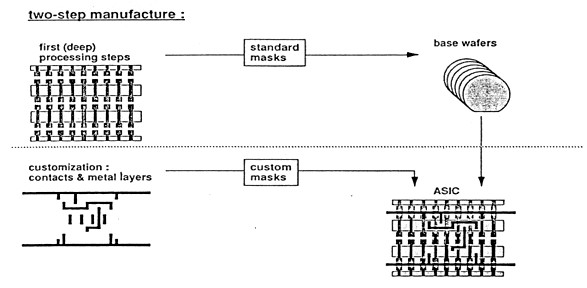
73



* Gate array implementation requires a two-step manufacturing process:
  1. The ﬁrst phase, which is based on generic (standard) masks, results in an array of uncommitted transistors on each GA chip.
  2. These uncommitted chips can be customized later, which is completed by deﬁning the metal interconnects between the transistors of the array.

Hardware Modeling Using Verilog

74



Hardware Modeling Using Verilog

75



* The GA chip utilization factor is higher than that of FPGA.
  + The used chip area divided by the total chip area.
* Chip speed is also higher.
  + More customized design can be achieved with metal mask designs.
* Typical gate array chips can implement millions of logic gates.

Hardware Modeling Using Verilog

76



**Standard Cell Based Design**

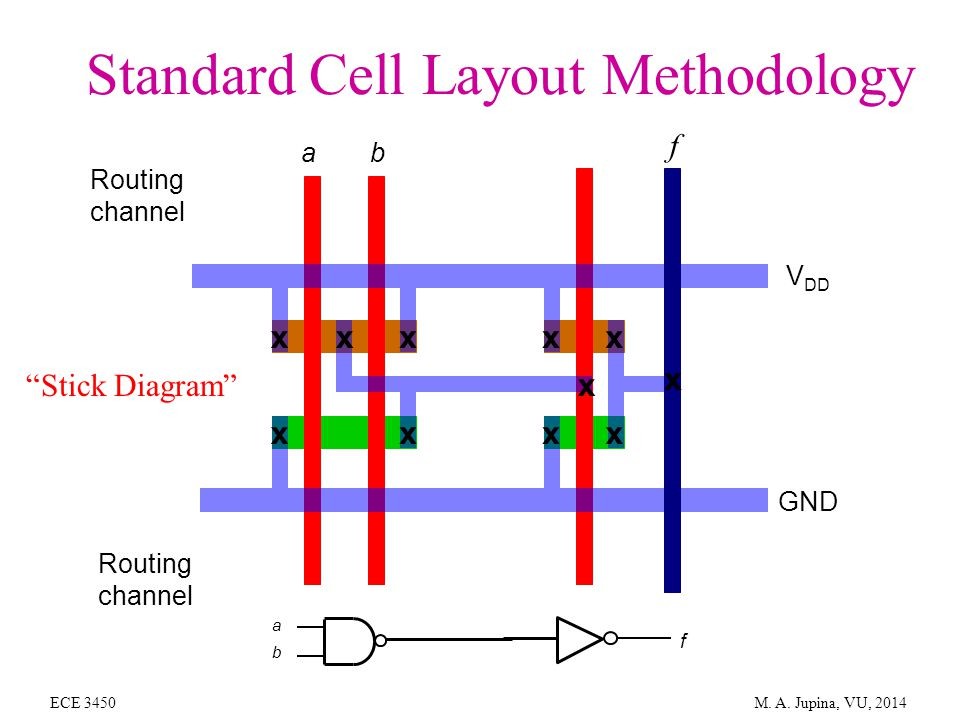


**Introduction**

* One of the most prevalent design styles.
  + Also called semi-custom design style.
  + Requires developing full custom mask set.
* Basic idea:
  + Commonly used logic cells are developed, and stored in a standard cell library.
  + Typical library may contain a few hundred cells (*Inverters, NAND gates, NOR gates, AOI gates, OAI gates, 2-to-1 MUX, D-latches, ﬂip- ﬂops, etc.*).

Hardware Modeling Using Verilog

78



**Standard Cell Example**

Made to stack side by side

* Fixed height
* Width can vary
* Can abut at VDD and GND

Hardware Modeling Using Verilog

80



**Characteristic of the Cells**

* 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.
  + Neighboring cells share a common power and ground bus.
* The input and output pins are located on the upper and lower boundaries of the cell.

Hardware Modeling Using Verilog

79

|  |  |
| --- | --- |
|  | |
| Routing Channel |  |
|  | |
| Routing Channel |  |
|  | |
|  | |

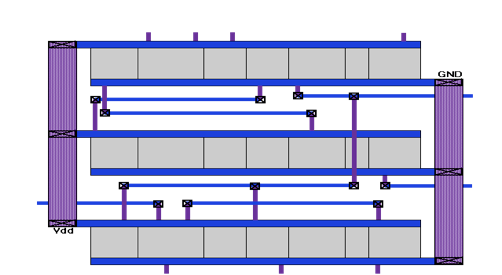


**Floorplan for Standard Cell Design**

* Inside the I/O frame which is reserved for I/O cells, the chip area contains rows or columns of standard cells.
  + Between cell rows are channels for routing.
  + Over-the-cell routing is also possible.
* The physical design and layout of logic cells ensure that
  + When placed into rows, their heights match.
  + Neighboring cells can abut side-by-side, which provides natural connections for power and ground lines in each row.

Hardware Modeling Using Verilog

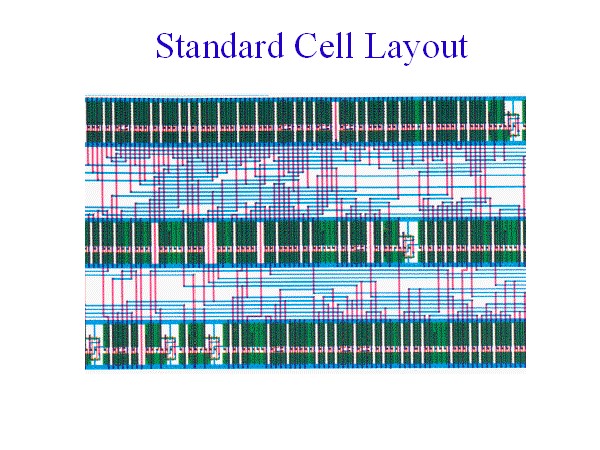
81



**Standard Cell Layout**

Hardware Modeling Using Verilog

82



Hardware Modeling Using Verilog

83

**Standard Cell Layout**



**Full Custom Design**



**Comparison Among Various Design Styles**

Hardware Modeling Using Verilog

89



**Introduction**

* Standard-cells based design is oien called semi custom design.
  + The cells are pre-designed for general use.
* In the full custom design, the entire mask design is done anew without use of any library.
  + The development cost of such a design style is prohibitively high.
  + The concept of design reuse is becoming popular to reduce design cycle time and cost.

Hardware Modeling Using Verilog

85



* The most rigorous full custom design can be the design of a memory cell.
  + Static or dynamic.
  + Since the same layout design is replicated, there would not be any alternative to high density memory chip design.
* For logic chip design, a good compromise can be achieved by combining diﬀerent design styles on the same chip.
  + Standard cells, data-path cells and PLAs.

Hardware Modeling Using Verilog

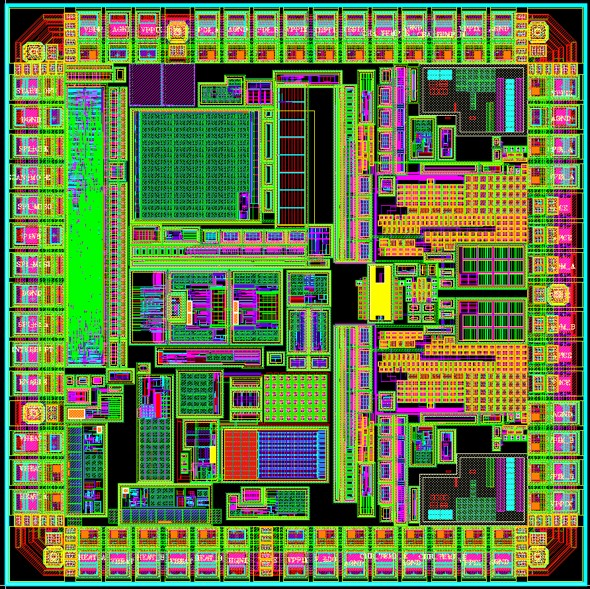
86



* In real full-custom layout in which the geometry, orientation and placement of every transistor is done individually by the designer.
  + Design productivity is usually very low (typically 10 to 20 transistors per day, per designer).
* In digital CMOS VLSI, full-custom design is rarely used due to the high labor cost.
  + Exceptions to this include the design of high-volume products such as memory chips, high-performance microprocessors and FPGA masters.

Hardware Modeling Using Verilog

87



**A full custom layout**

Hardware Modeling Using Verilog

88

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Design Style** | | | |
| **FPGA** | **Gate array** | **Standard cell** | **Full custom** |
| **Cell size** | **Fixed** | **Fixed** | **Fixed height** | **Variable** |
| **Cell type** | **Programmable** | **Fixed** | **Variable** | **Variable** |
| **Cell placement** | **Fixed** | **Fixed** | **In row** | **Variable** |
| **Interconnect** | **Programmable** | **Variable** | **Variable** | **Variable** |
| **Design time** | **Very fast** | **Fast** | **Medium** | **Slow** |



**END OF LECTURE 05**

Hardware Modeling Using Verilog

90