Lect 17

Hardware Programming (Verilog HDL)

CS221: Digital Design

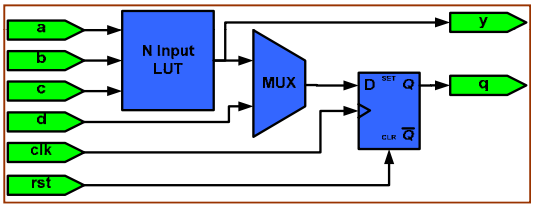
Dr. A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

Outline

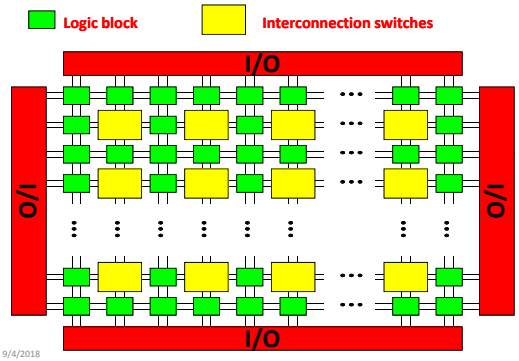
- FPGA/ASIC Design Flow
- HDL Programming : Verilog HDL
- HDL Rules
- HDL Module and Examples
- HDL levels : Data flow, Structural and Behavioral, UDP
- Testing and Simulation

FPGA - Field Programmable Gate Array

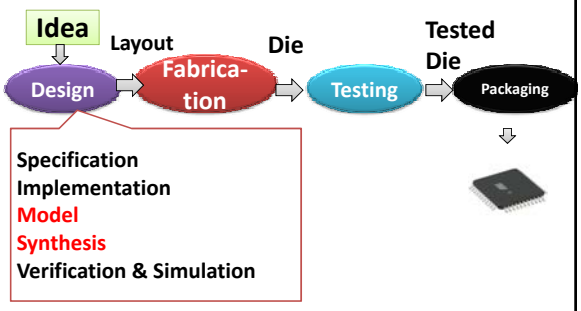
- Programmable logic blocks or CLB
  - (Logic Element "LE")
  - Implement combinatorial and sequential logic. Based on LUT and DFF.



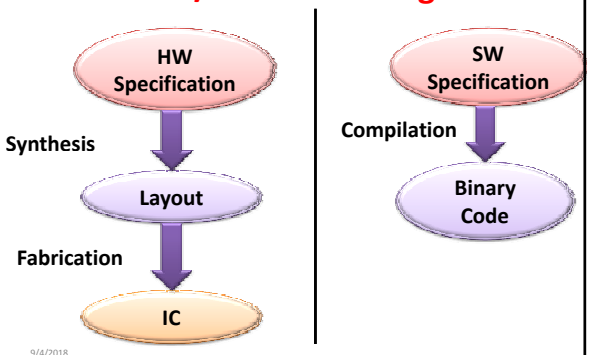
FPGA - Field Programmable Gate Array



IC Design Process



Hardware/Software Design Flow

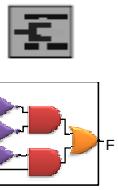


Model

- Representation of abstract view of the System
- Varying abstractions
  - functional only
  - timing only
  - functional + timing

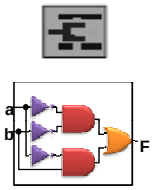
Hardware Specification

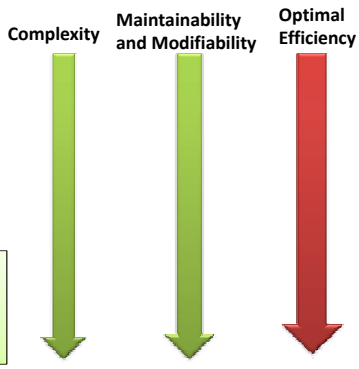
- Layout editor
  - directly enter layout
  - Up to ~10^2 of unique transistors
  - Complex circuits
  - Memory, aided by generators
- Schematic Capture
  - Enter gates and interconnections
  - Up to ~10^4 transistors
- Hardware Description Languages
  - Enter text description
  - 10^7 transistors




```
module ..
If (x < y) then
Y=x and z;
....
```


Hardware Specification






```
Entity ..
If (x < y) then
Y=x and z;
....
```


Modelling: level of detail

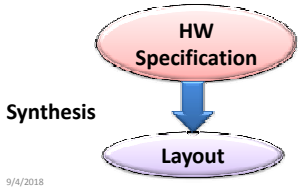
- Behavioral Level
  - no clock cycle level commitment
- Register-Transfer Level (RTL)
  - Operations committed to clock cycles
- Gate level
  - structural netlist

```
for (i=0;i <4;i++)
  S = S+ A[i]
```

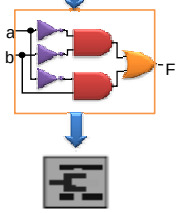
```
Cycle 1:  T1 = A[0] + A[1]
          T2 = A[2] + A[3]
Cycle 2:  S = T1 + T2
```


Synthesis

- HDL → Layout
  - HDL → Gates
  - Gates → Layout



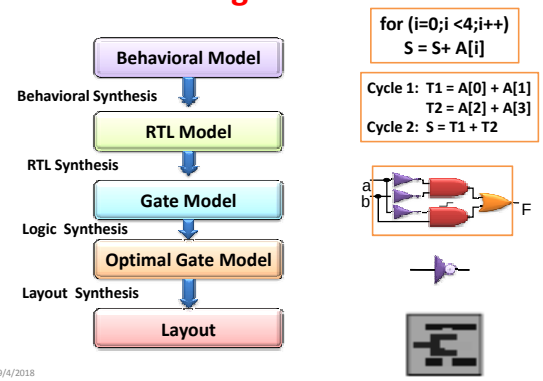

```
module ..
If (x < y) then
Y=x and z;
....
```



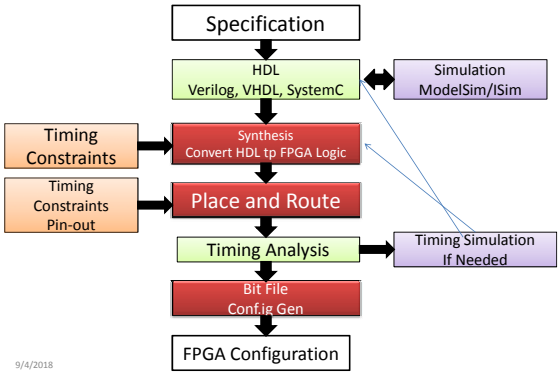
Synthesis

- Behavioral Synthesis (Process & Sequential)
  - Behavioral HDL → RTL HDL
  - No notion of clock to Clocked
- RTL Synthesis
  - RTL HDL → Gates
- Layout Synthesis
  - Gates → Layout

Design Flow



FPGA Design flow

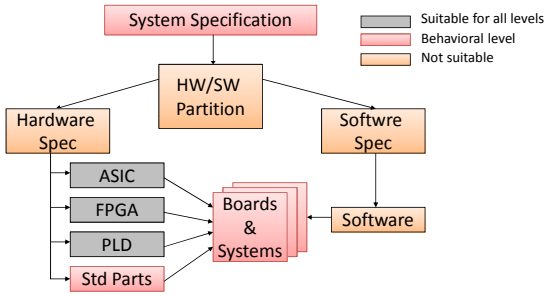


Verilog
HDL

What is Verilog

- Hardware Description Language (HDL)
- Developed in 1984
- Standard: IEEE 1364, Dec 1995

Application Areas of Verilog

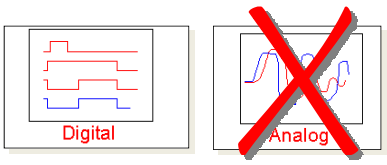


HDL, Area of Application

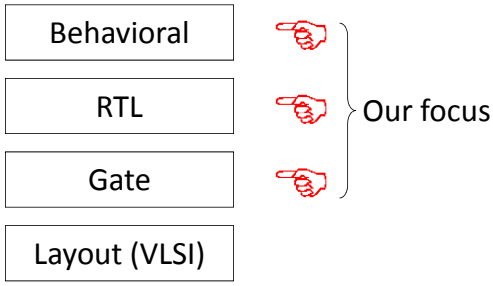
- Design Entry
- Logic Simulation
- Functional Verification
- Digital Circuit Synthesis
- Timing Verification
- Fault Simulation
- Documntation

Basic Limitation of Verilog

Description of digital systems only

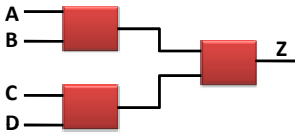


Abstraction Levels in Verilog

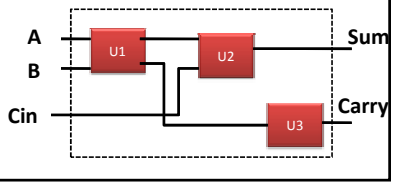


Main Language Concepts (i)

- Concurrency

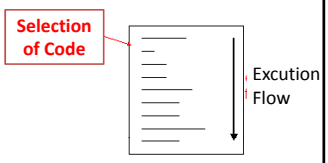


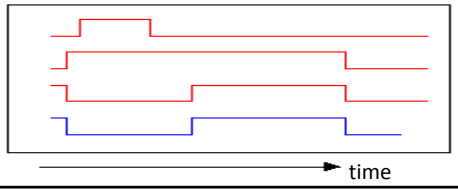
- Structure



Main Language Concepts (ii)

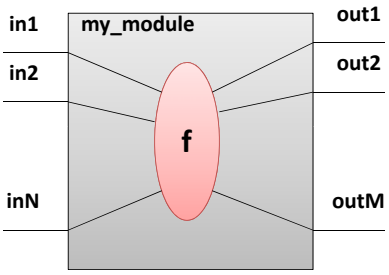
- Procedural Statements
- Time





Lets Start with an Example of Verilog HDL module

Module



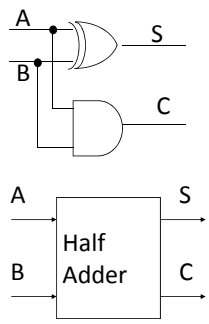
Everything you write in Verilog must be inside a module
exception: compiler directives

Module

```
module my_module(out1, .., inN);
output out1, .., outM;
input in1, .., inN;
.. // declarations
.. // description of f (maybe
.. // sequential)
endmodule
```

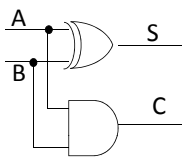
Everything you write in Verilog must be inside a module
exception: compiler directives

Example: Half Adder
(Data Flow Model: using Boolean Equations)




```
module half_adder(S,C,A,B);
output S, C;
input A, B;

wire S, C, A, B;

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

endmodule
```


Example: Half Adder
(Data Flow Model: using Boolean Equations)



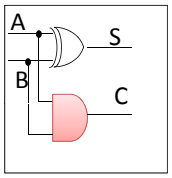

```
module half_adder(S,C,A,B);
output S, C;
input A, B;

wire S, C, A, B;

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

endmodule
```

```
assign S = A ^ B;
assign C = A & B;
```

Same

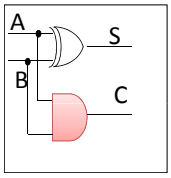
```
assign C = A & B;
assign S = A ^ B;
```


Example: Half Adder, 2nd Implementation Using Structural/gate primitive inter connection




```
module half_adder(S,C,A,B);
output S, C;
input A, B;

wire S, C, A, B;

xor  (S, A, B);
and  (C, A, B);

endmodule
```


Example: Half Adder, 2nd Implementation Using Structural/gate primitive inter connection




```
module half_adder(S,C,A,B);
output S, C;
input A, B;

wire S, C, A, B;

xor  (S, A, B);
and  (C, A, B);

endmodule
```

```
xor  (S, A, B);
and  (C, A, B);
```

Same

```
and  (C, A, B);
xor  (S, A, B);
```


Verilog HDL Languages

User Identifiers

- Formed from {[A-Z], [a-z], [0-9], _, $}
- Can't begin with $ or [0-9]
  - `myidentifier`
  - `m_y_identifier`
  - `3my_identifier`
  - `$my_identifier`
  - `_myidentifier$`
- Case sensitivity : `myid` ≠ `Myid`

9/4/2018

Comments : same as C++ Style

- `// The rest of the line is a comment`
- `/* Multiple line comment */`
- `/* Nesting /* comments */ do NOT work */`

9/4/2018

Verilog Value Set

- ***0*** represents low logic level or false condition
- ***1*** represents high logic level or true condition
- ***x*** represents unknown logic level
- ***z*** represents high impedance logic level == > open circuit

9/4/2018

Truth Tables (Updated..)

AND	0	1	X	Z
0	0	0	0	0
1	0	1	X	X
X	0	x	x	X
Z	0	x	x	x

XOR	0	1	X	Z
0	0	**1**	**X**	**X**
1	**1**	**0**	X	X
X	**X**	x	x	X
Z	x	x	x	x

OR	0	1	X	Z
0	0	**1**	**X**	**X**
1	**1**	1	**1**	**1**
X	**X**	**1**	**X**	X
Z	**X**	**1**	**X**	**X**

NOT	0	1	X	Z
OUT	**1**	**0**	**X**	**X**

Sorry: There were two mistakes in this Slide, now corrected