FPGA and Verilog

Safwan Wshah Project TA

srwshah@buffalo.edu

Office hours :

- Tuesday 11-12
- Friday 12-1

Office Location: 30 Commons -- CEDAR

Semester Plan

- Week 2 – Introduction to FPGA and Verilog
- Week 3 – Structural Verilog + The Verilog HDL Test Fixture
- Week 4 – Behavioral Modeling
- Week 5 –Example
  - » Writing Modular Code in Verilog
  - » *Managing a Large Project;*
  - » I/O on the Basys 2 Board
- Week 6- Project 1 specification and grading criteria

Projects

- Project 1
  - Will be basic project that make things easy to start the final project.
- Project 2
  - Advanced project, there are many suggested projects and you can come up with your own.

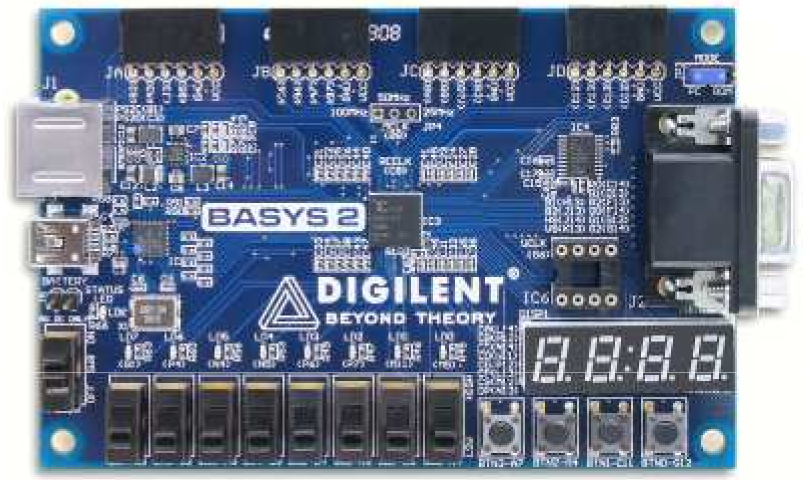
FPGA

- **Field-Programmable Gate Arrays** (FPGAs) are pre-fabricated silicon devices that can be electrically programmed to become almost any kind of digital circuit or system.

- Applications of FPGAs include
  - » digital signal processing,
  - » software-defined radio,
  - » Aerospace
  - » medical imaging, computer vision,
  - » speech recognition,
  - » cryptography, bioinformatics, computer hardware emulation, radio astronomy, metal detection and a growing range of other areas.

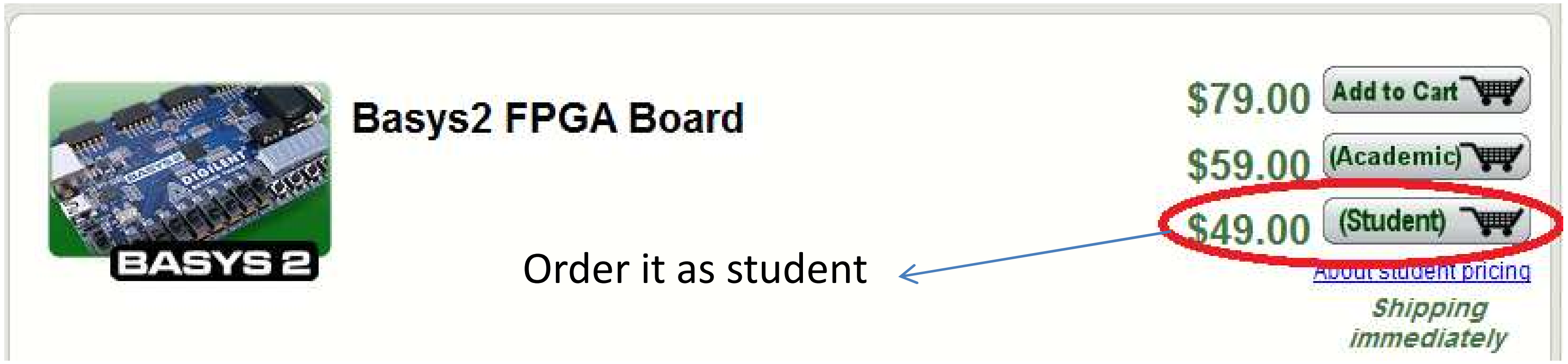
In the Final Project you will have many project options or you can come up with your own

FPGA



- We are going to use Basys 2 FPGA from Xilnix
  - http://www.xilinx.com/

- Each student should buy his own board
  - When to buy ? ***NOW***
  - How to buy?
    - Go to xilnix website , and order it from there http://www.digilentinc.com/Products/Detail.cfm?Prod=BASYS2


Basys2 FPGA Board

$79.00 Add to Cart

$59.00 (Academic)

$49.00 (Student)

About student pricing

Shipping immediately

Order it as student

Overview of Hardware Description Languages

Hardware Description Languages (HDL):

- Verilog (What we are going to use)
  - Similar syntax to C
  - Commonly used in Industry (USA & Japan)
- VHDL (VHSIC hardware description language)
  - Similar syntax to ADA(extended from Pascal)
  - Commonly used in Government contract work ,Academia and Europe.
- Both are
  - IEEE standards
  - Supported by ASIC(Application-specific integrated circuit) & FPGA synthesis tools

Advantages of HDLs

- Descriptions are portable & independent of technology
  - Allows for easy modification and reuse of previous designs
- Efficient utilization
  - Simply resynthesize
  - New implementation is faster (Design is the same)
  - Decreases design cycle time
- Automatic synthesis of HDL into a working implementation
  - Bypasses some steps such as manual minimization techniques
- Testing
  - Behavioral description can be quickly & easily synthesized prior to implementation
  - Very advantageous early in design process
  - Provides for early evaluation of alternative structures

HDL Use in Design Tools & the Design Process

- Design Entry
- Design Verification
- Test Generation
- Fault Analysis & Simulation
- Timing Analysis & Verification
- Synthesis
- Automatic Schematic Generation
  - Improves efficiency of design flow by eliminating translations of design description as we move through design process

Digital system Abstractions*

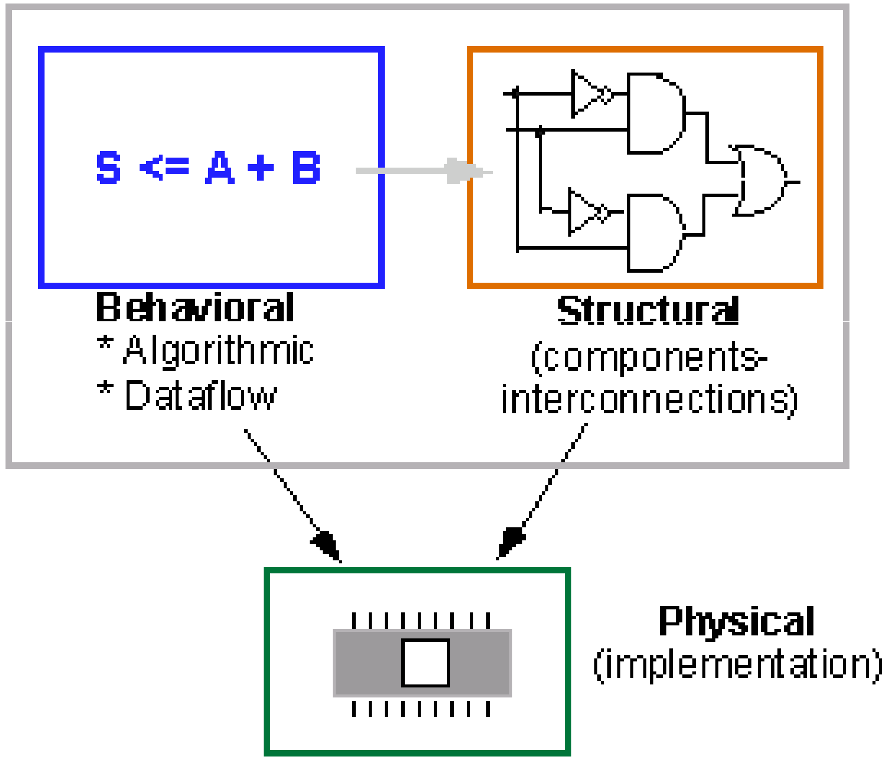


Figure 1: Levels of abstraction: Behavioral, Structural and Physical

Behavioral Modeling

- Abstract description of how the circuit works
- Does not include any indication of structure (implementation) details
- Useful early in design process
  - Allows designer to get a sense of circuit's characteristics before embarking on design process.
  - After functionality is well defined, structural design may follow

- Procedural Models
  - Synthesis Tools
    - Take a behavioral specification & generate the implementation

Structural Modeling

- System is described as a set of modules consisting of:
  - Interface
  - Description

- Modules interconnected to create system

Structural vs. Behavioral

- Behavioral
  - Functional
  - No implementation details

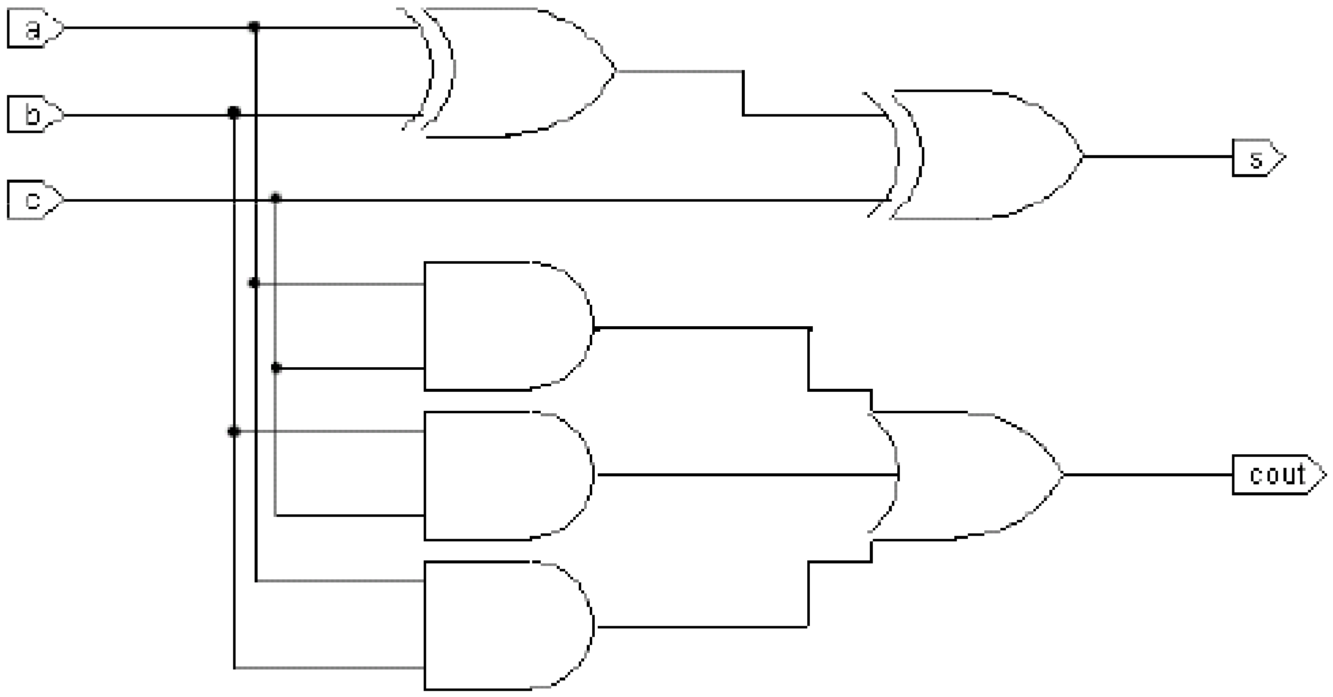
Structural Modeling

- System is described as a set of modules consisting of:
  - Interface:
    - Declares nets & registers which comprise the two fundamental data types in Verilog
      - Nets
        - » **Wire, wired-AND, wired-OR, trireg**
        - » **Used to connect structures (eg. - gates)**
  - Description

- **Basic Module**
  - Basic Module Format

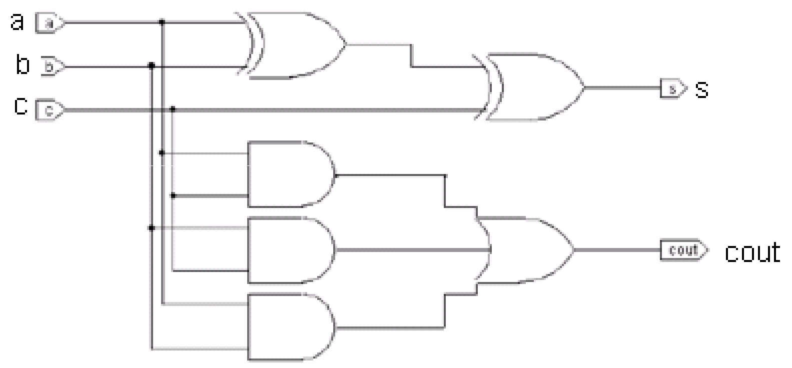
```
module  name;
   Interface

   Description

endmodule
```

- **The Full Adder**



Basic Module

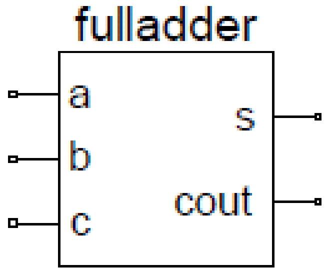
```
module fulladder() ;

        wire w1, w2, w3, w4, s, cout;
        reg a, b, c;

        xor
                g1(w1, a, b),
                g2(s, w1, c);
        and
                g3(w2, c, b),
                g4(w3, c, a),
                g5(w4, a, b);
        or
                g6(cout, w2, w3, w4);
endmodule
```



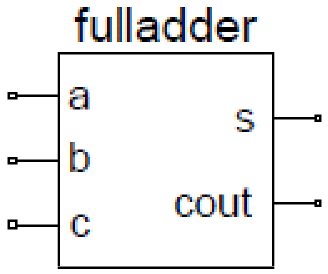
- **Comments**
  - Comments are preceded by //

Creating Ports

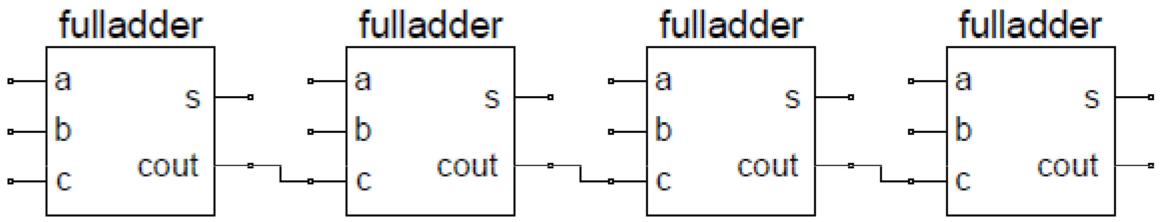
  - Port names are known only inside the module
  - Declarations
    - Input
    - Output
    - Bidirectional
- Full Adder Module



```
module fulladder(a,b,c,s,cout);
   input a,b,c;
   output s,cout;
            xor #1
               g1(w1, a, b),
               g2(s, w1, c);
            and #1
               g3(w2, c, b),
               g4(w3, c, a),
               g5(w4, a, b);
            or #1
               g6(cout, w2, w3, w4);
endmodule
```



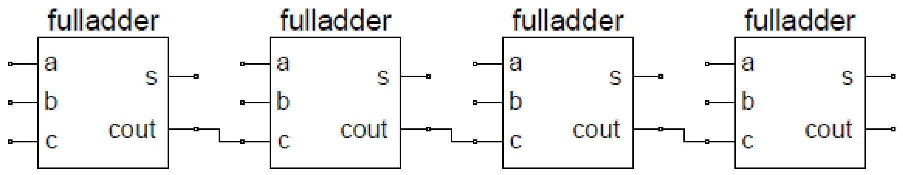
- **Instantiation**
  - Modules can be instantiated to complete a design
  - 4-bit Ripple Carry Adder



Vectors

- Scalar
  - A single bit net or reg
- Vector
  - A multiple bit net or reg
- Advantage
  - Vectors make for a more natural way of scaling up a design
- Example
  - Consider the 4-bit adder
- Using scalars:
  - A3 A2 A1 A0 + B3 B2 B1 B0 + Cin = Cout S3 S2 S1 S0
- Using vectors:
  - A + B + Cin = Cout, S
  - A[3:0] + B[3:0] + Cin = Cout, S[3:0]

```
module fulladder(a,b,c,s,cout);
input a,b,c;
output s,cout;
xor #1
    g1(w1, a, b),
    g2(s, w1, c);
and #1
    g3(w2, c, b),
    g4(w3, c, a),
    g5(w4, a, b);
or #1
    g6(cout, w2, w3, w4);
endmodule
```



```
module fourBitAdder(x,y,s,cout,cin);
    input [3:0] x,y;
    output [3:0] s;
    input cin;
    output cout;
    wire c[3:0];
    fulladder f0 (x[0],y[0],cin,s[0],c[0]);
    fulladder f1 (x[1],y[1],c[0],s[1],c[1]);
    fulladder f2 (x[2],y[2],c[1],s[2],c[2]);
    fulladder f3 (x[3],y[3],c[2],s[3],cout);
endmodule
```

References

- Donald E. Thomas and Philip R. Moorby, *The Verilog Hardware Description Language,* Kluwer Academic Publishers, 1998
- Samir Palnitkar, *Verilog HDL A Guide to Digital Design and Synthesis, Prentice Hall, Inc., 4th* Edition, 1996
- David R. Smith and Paul D. Franzon, *Verilog Styles of Digital Systems, Prentice Hall, Inc.,* 2000
- Michael D. Ciletti, *Advanced Digital Design with the Verilog HDL, Pearson Education, Inc.* (Prentice Hall), 2003
- * VHDL Tutorial, Jan Van der Spiegel, University of Pennsylvania, Department of Electrical and Systems Engineering
- Wikipeidia