

Lecture 9. Hardware design with Verilog. Case study: design of adders.

Dmitry Zaitsev

<http://daze.ho.ua>

Combinatorial logic

- specify functioning by truth table
- synthesize DNF for each bit of result
- minimize DNFs
- draw scheme using gates
- ✓ sometimes we use restricted set of gates
- ✓ sometimes we start with specification using formulae

Standard forms of Boolean algebra

- Disjunctive normal form

$$(A \wedge \neg B \wedge \neg C) \vee (\neg D \wedge E \wedge F)$$

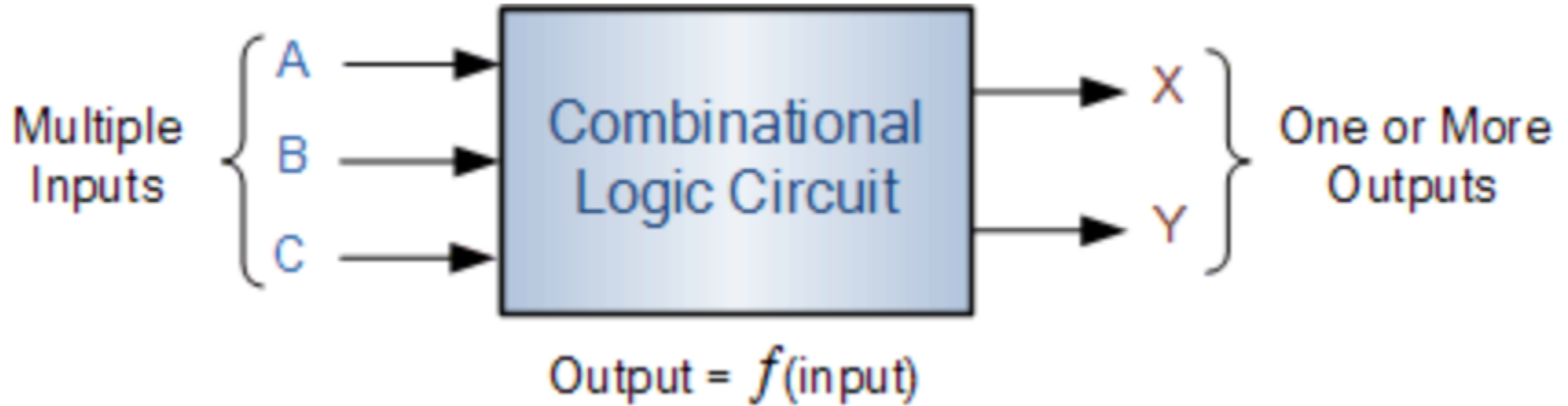
- Conjunctive normal form

$$(A \vee \neg B \vee \neg C) \wedge (\neg D \vee E \vee F)$$

- Polynomial of Zhegalkin – algebraic normal form








$$1 \oplus A \oplus ABD.$$

Combinational Logic Circuits

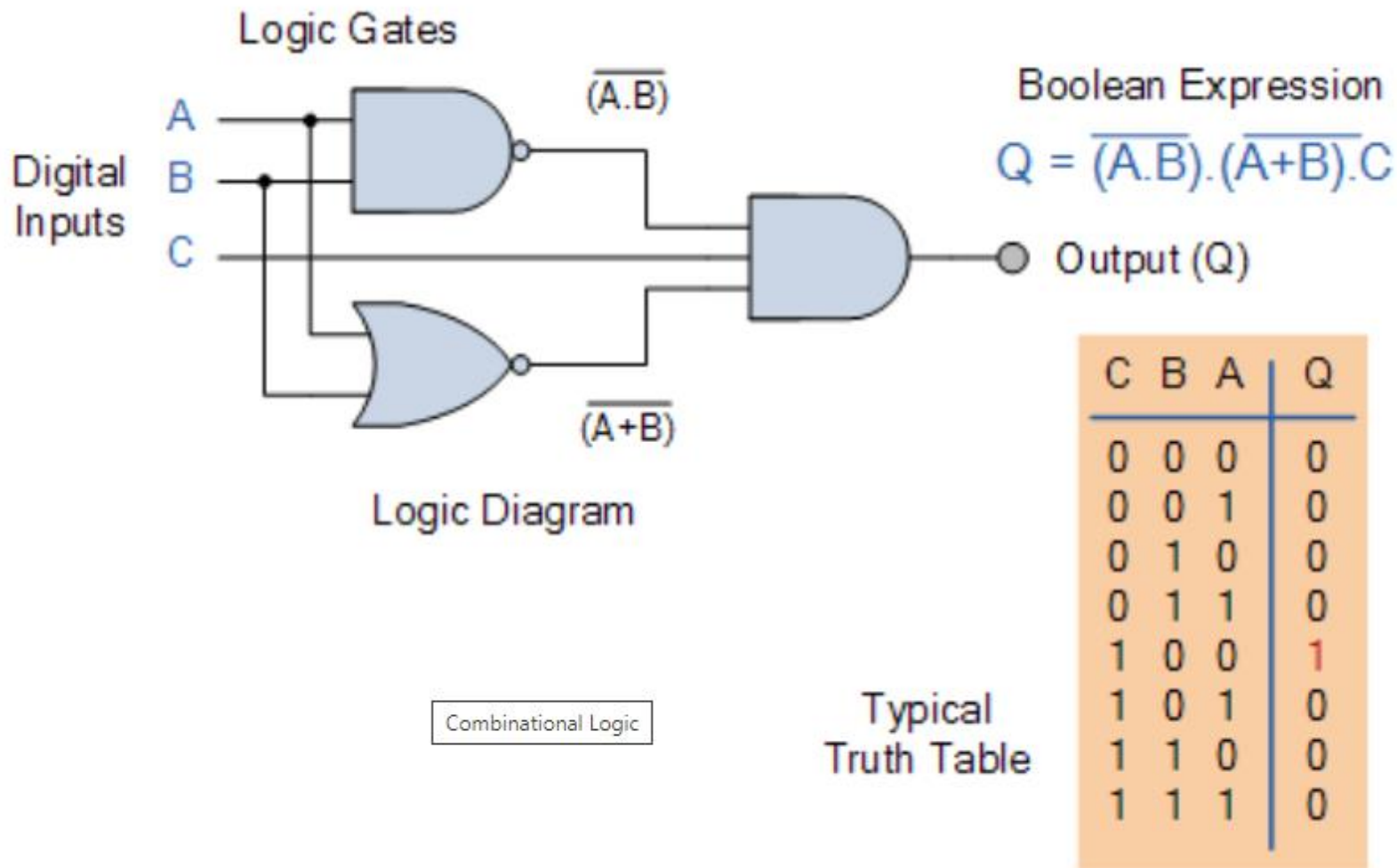


The outputs of **Combinational Logic Circuits** are only determined by the logical function of their current input state

Logic gates

| Name | NOT | AND | NAND | OR | NOR | XOR | XNOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------|--|---|---|--|---|---|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Alg. Expr. | \overline{A} | AB | \overline{AB} | $A+B$ | $\overline{A+B}$ | $A\oplus B$ | $\overline{A\oplus B}$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Symbol |  |  |  |  |  |  |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Truth Table | <table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table> | A | X | 0 | 1 | 1 | 0 | <table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table> | B | A | X | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | <table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table> | B | A | X | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | <table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table> | B | A | X | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | <table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table> | B | A | X | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | <table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table> | B | A | X | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | <table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table> | B | A | X | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| A | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B | A | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B | A | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B | A | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B | A | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B | A | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| B | A | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Logic diagram



Combinational Logic Circuit

```
graph TD; A[Combinational Logic Circuit] --> B[Arithmetic & Logical Functions]; A --> C[Data Transmission]; A --> D[Code Converters]; B --> B1[Adders<br/>Subtractors<br/>Comparators<br/>PLD's]; C --> C1[Multiplexers<br/>Demultiplexers<br/>Encoders<br/>Decoders]; D --> D1[Binary<br/>BCD<br/>7-segment];
```

Arithmetic &
Logical Functions

Adders
Subtractors
Comparators
PLD's

Data
Transmission

Multiplexers
Demultiplexers
Encoders
Decoders

Code
Converters

Binary
BCD
7-segment

Verilog

- hardware description language (HDL)
- design and verification of digital circuits
- syntax similar to C
- standardized as IEEE 1364
- tools: ModelSim (siemens), VeriLogger, Verilator etc

Verilog features

- specify schemes with modules and formulae
- hierarchical design: using modules inside modules
- automatic graphical layout of scheme using gates
- automated simulation and testing
- final generation of technological specifications to produce chips

Verilog module

A Verilog module is a building block that defines a design or testbench component, by defining the building block's ports and internal behaviour. Higher-level modules can embed lower-level modules to create hierarchical designs. Different Verilog modules communicate with each other through Verilog port.

Defining a Verilog Module

- Keyword *module* to begin the definition
- Identifier that is the name of the module
- Optional list of parameters
- Optional list of ports (to be addressed more deeply in a future article)
- Module item
- Keyword *endmodule* to end the definition

Half adder

- One bit adder
- Input:
 - bit x
 - bit y
- Output ($s=x+y$):
 - bit s – sum
 - bit c – carry

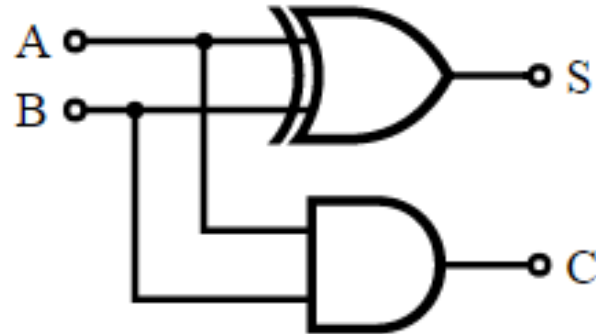
$$\begin{array}{r} 1 \\ + \\ 1 \\ \hline 10 \\ \text{CS} \end{array}$$

Half adder design

| Half Adder Truth Table | | | |
|------------------------|---|-------|-----|
| A | B | Carry | Sum |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$$S = A \oplus B$$

$$C = A \wedge B$$



Half adder module

```
module half_adder1
(  input x,
    input y,
    output s,
    output c  );

    assign s = x^y;
    assign c = x&y;

endmodule
```

Verilog online: <http://digitaljs.tilk.eu>

Verilog Module for Design and Te x Half Adder - Nandland x DigitalJS Online x +

← → ↻ Not secure | digitaljs.tilk.eu 🔍 📄 ☆ 🌐 ⚙️ 🖨️ 👤 ⋮

🔊 ⏸ ⏩ → ↪ ⌚ 3205 📄 💾 🔗

Setup I/O unnamed.sv 🗑️ X

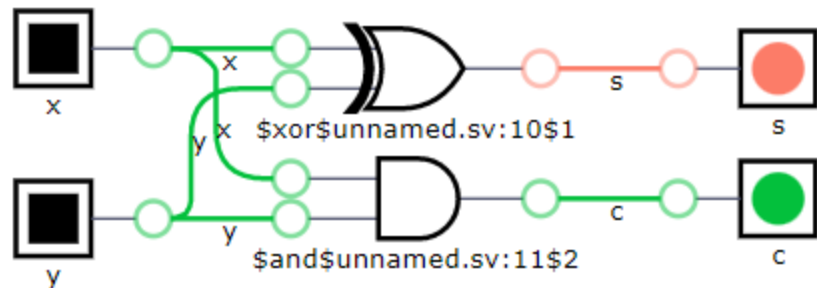
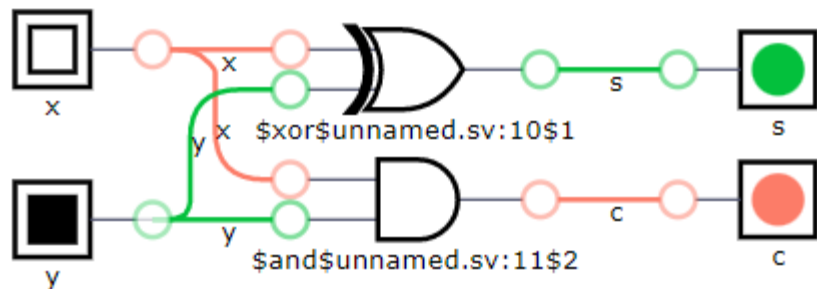
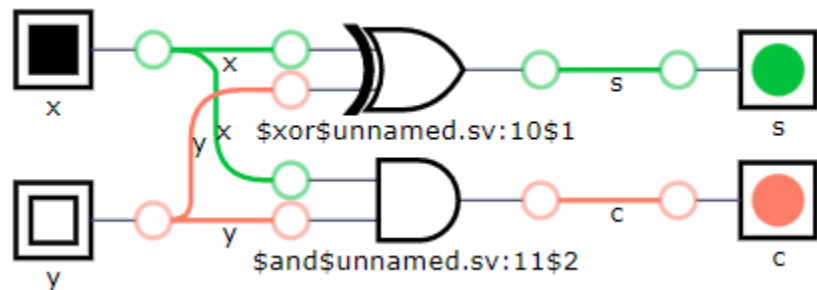
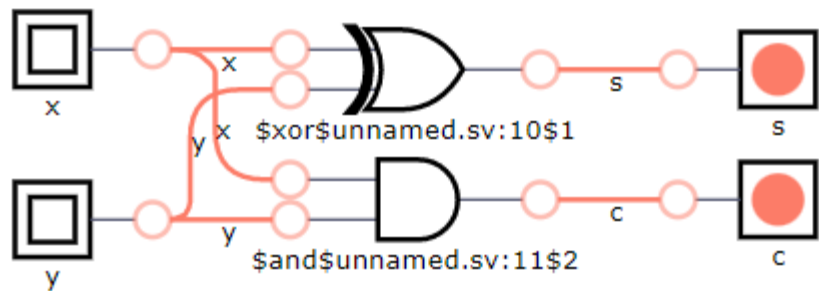
```
1 module half_adder1
2 (
3   input x,
4   input y,
5
6   output s,
7   output c
8 );
9
10 assign s = x^y;
11 assign c = x&y;
12
13 endmodule
14
```

Synthesize and simulate!

00000000

Windows Type here to search 🐧 📁 🌐 📄 📄 📄 📄 📄 📄 65°F Partly sunny ⬆ 🗣 ENG 5:44 PM 4/25/2023 🗨

Test half adder



Full adder

- One bit adder
- Input:
 - bit x
 - bit y
 - input carry c_{in}
- Output ($s=x+y+c_{in}$):
 - bit s – sum
 - bit c_{out} – output carry

$$\begin{array}{r} 1 c_{in} \\ + \\ 1 x \\ + \\ 1 y \\ \hline 11 \\ c_s \end{array}$$

Full adder design

| x | y | c_in | c_out | s |
|---|---|------|-------|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

$$s = (x \oplus y) \oplus c_{in}$$

$$c_{out} = ((x \oplus y) \wedge c_{in}) \oplus (x \wedge y)$$

```
module fulladder1
( input x,
  input y,
  input c_in,

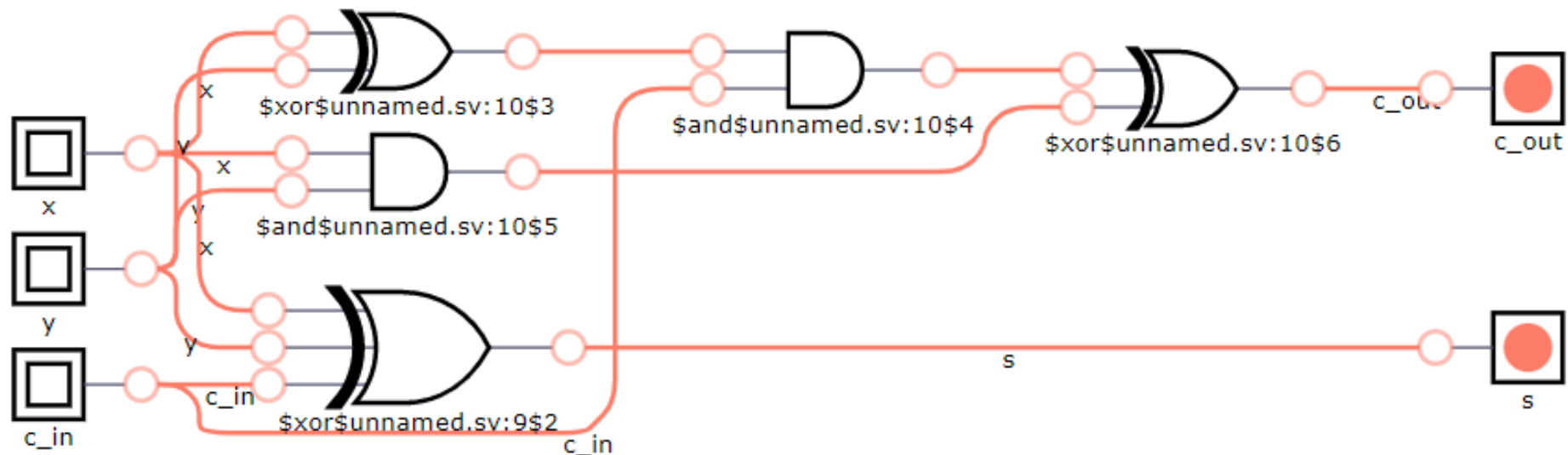
  output s,
  output c_out );

  assign s = (x^y) ^ c_in;
  assign c_out = ((x^y) & c_in) ^ (x&y);

endmodule
```

Full adder module

Full adder scheme



Hierarchical design

- specify a bus as an array of wires
- specify components
- use components
- connect input and output wires of components

Four bits sequential adder

- idea: connect 4 full adders sequentially
- c_out of an adder to c_in of the next adder
- c_in of the first feed with 0
- use c_out of the last as the resulting carry
- example:

```
    1001
    1101
  -----
    10110
```

Four bits sequential adder module

```
module Adder4(  
    input [3:0] a,  
    input [3:0] b,  
    output [3:0] sum,  
    output carry  
);  
    wire cin;  
  
    assign cin = 1'b0;  
    fulladder1 s0( .x( a[0] ), .y( b[0]), .c_in( cin ), .s( sum[0]), .c_out( ripple0 ) );  
    fulladder1 s1( .x( a[1] ), .y( b[1]), .c_in( ripple0 ), .s( sum[1]), .c_out( ripple1 ) );  
    fulladder1 s2( .x( a[2] ), .y( b[2]), .c_in( ripple1 ), .s( sum[2]), .c_out( ripple2 ) );  
    fulladder1 s3( .x( a[3] ), .y( b[3]), .c_in( ripple2 ), .s( sum[3]), .c_out( carry ) );  
  
endmodule
```

Automatic layout

DigitalJS Online

Not secure | digitaljs.tilk.eu

535

Setup I/O unnamed.sv

```
1 module fulladder1
2 (
3   input x,
4   input y,
5   input c_in,
6
7   output s,
8   output c_out
9 );
10
11 assign s = (x^y) ^ c_in;
12 assign c_out = ((x^y) & c_in
13
14 endmodule
15
16 module Adder4(
17   input [3:0] a,
18   input [3:0] b,
19   output [3:0] sum,
20   output carry
21 );
22 wire cin;
23
24 assign cin = 1'b0;
25 fulladder1 s0(.x( a[0] )
26 fulladder1 s1(.x( a[1] )
27 fulladder1 s2(.x( a[2] )
```

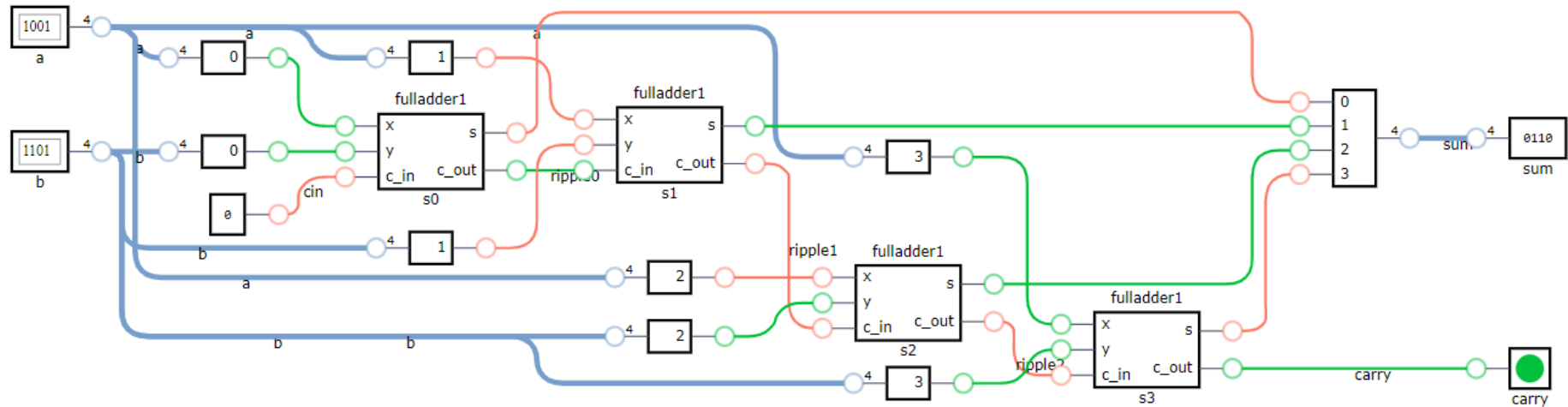
Synthesize and simulate!

The diagram shows the automatic layout of a 4-bit adder circuit. It consists of four fulladder1 modules (s0, s1, s2, s3) and a 4-bit sum output. The inputs are 4-bit buses a and b. The carry-in (cin) is set to 1'b0. The outputs are a 4-bit sum and a carry-out. The layout shows the internal connections between the modules, including the carry propagation (ripple) and the final sum output.

Activate Windows
Go to Settings to activate Windows.

66°F Sunny 2:01 PM 4/26/2023

Manual testing



Show components

DigitalJS Online

Not secure | digitaljs.tilk.eu

3601

fulladder1 s1

Activate Windows
Go to Settings to activate Windows.

Type here to search

66°F Mostly sunny 2:42 PM 4/26/2023

Pros and contras sequential scheme

- Pros:
 - simplicity, regularity of design
- Contras:
 - long time of work, total delay grows linearly with respect to number of bits