Binary Adders

ECE 2372 | Modern Digital System Design | Texas Tech University

Lecture Overview

- Binary Addition
- Designing a 1-bit Binary Adder
- The Ripple-Carry Adder
- Implementation in Verilog HDL

Binary Addition

Binary Adders | Modern Digital System Design

Binary Addition

Binary addition works the same as decimal addition

```
 147
+23
----
```

Binary Addition

Binary addition works the same as decimal addition

```
  1   Carry
 147
+23
----
   0  SUM
```

Binary Addition

Binary addition works the same as decimal addition

```
 01     Carry
 147
+23
----
  70    SUM
```

Binary Addition

Binary addition works the same as decimal addition

01	Carry
147	
+23	
170	SUM

Binary Addition

Binary addition works the same as decimal addition

```
 1100
+1010
-----
```

Binary Addition

Binary addition works the same as decimal addition

```
   0   Carry
 1100
+1010
-----
    0  SUM
```

Binary Addition

Binary addition works the same as decimal addition

```
   00   Carry
 1100
+1010
-----
   10   SUM
```

Binary Addition

Binary addition works the same as decimal addition

```
 000   Carry
 1100
+1010
-----
  110  SUM
```

Binary Addition

Binary addition works the same as decimal addition

```
1000   Carry
 1100
+1010
-----
 0110  SUM
```

Binary Addition

Binary addition works the same as decimal addition

```
 1000   Carry
  1100
 +1010
 -----
 10110  SUM
```

Binary Addition

Using decimal addition is a great way to double-check your work.

```
  1100       ??
+1010      +??
-----      ---
10110        ??
```

Binary Addition

Using decimal addition is a great way to double-check your work.

```
  1100        12
+1010       +??
-----       ---
10110        ??
```

Binary Addition

Using decimal addition is a great way to double-check your work.

```
  1100        12
+1010       +10
-----       ---
10110        ??
```

Binary Addition

Using decimal addition is a great way to double-check your work.

$$\begin{array}{r} 1100 \\ +1010 \\ \hline 10110 \end{array} \qquad \begin{array}{r} 12 \\ +10 \\ \hline 22 \end{array}$$

What questions do you have about binary addition?

Exam Problem | Binary Addition

What is the result of the following binary addition operation?

A. 011010

B. 110001

C. 110111

D. 111101

$$\begin{array}{r} 100110 \\ +010111 \\ \hline \end{array}$$

Exam Problem | Binary Addition

What is the result of the following binary addition operation?

~~A. 011010~~

~~B. 110001~~

~~C. 110111~~

D. 111101

$$\begin{array}{r} 100110 \\ +010111 \\ \hline \end{array}$$

Exam Problem | Binary Addition

What is the result of the following binary addition operation?

~~A. 011010~~

~~B. 110001~~

~~C. 110111~~

D. 111101

$$
\begin{array}{r}
0 \\
100110 \\
+010111 \\
\hline
1
\end{array}
$$

Exam Problem | Binary Addition

What is the result of the following binary addition operation?

~~A. 011010~~

~~B. 110001~~

~~C. 110111~~

D. 111101

$$\begin{array}{r} 10 \\ 100110 \\ +010111 \\ \hline 01 \end{array}$$

Exam Problem | Binary Addition

What is the result of the following binary addition operation?

~~A. 011010~~

~~B. 110001~~

~~C. 110111~~

D. 111101

$$
\begin{array}{r}
110 \\
100110 \\
+010111 \\
\hline
101
\end{array}
$$

Exam Problem | Binary Addition

What is the result of the following binary addition operation?

~~A. 011010~~

~~B. 110001~~

~~C. 110111~~

D. 111101

$$
\begin{array}{r}
0110 \\
100110 \\
+010111 \\
\hline
1101
\end{array}
$$

Exam Problem | Binary Addition

What is the result of the following binary addition operation?

~~A. 011010~~

~~B. 110001~~

~~C. 110111~~

D. 111101

$$
\begin{array}{r}
00110 \\
100110 \\
+010111 \\
\hline
11101
\end{array}
$$

Exam Problem | Binary Addition

What is the result of the following binary addition operation?

~~A. 011010~~

~~B. 110001~~

~~C. 110111~~

D. 111101

```
 0000110
  100110
 +010111
 -------
  111101
```

Designing a 1-bit Binary Adder

Binary Adders | Modern Digital System Design

Designing a 1-bit Binary Adder

We will treat this problem like any other combination logic problem.

1. Derive the truth table.
2. Find the generalized minterm expansion
3. Populate a Karnaugh Map
4. Use the K-Map to get the optimized Boolean expression.
5. Draw the Logic Circuit

Designing a 1-bit Binary Adder

1. Derive the Truth Table

A	B	Cin	Cout	Sum
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Designing a 1-bit Binary Adder

1. Derive the Truth Table

A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Designing a 1-bit Binary Adder

1. Derive the Truth Table

A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Designing a 1-bit Binary Adder

1. Derive the Truth Table

A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Designing a 1-bit Binary Adder

1. Derive the Truth Table

A	B	Cin	Cout	Sum
0	**0**	**0**	0	0
0	**0**	**1**	0	1
0	**1**	**0**	0	1
0	**1**	**1**	1	0
1	**0**	**0**		
1	**0**	**1**		
1	**1**	**0**		
1	**1**	**1**		

Designing a 1-bit Binary Adder

1. Derive the Truth Table

A	B	Cin	Cout	Sum
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	**1**	**0**		
1	**1**	**1**		

Designing a 1-bit Binary Adder

1. Derive the Truth Table

A	B	Cin	Cout	Sum
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	**1**	**1**		

Designing a 1-bit Binary Adder

1. Derive the Truth Table

A	B	Cin	Cout	Sum
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**		

Designing a 1-bit Binary Adder

1. Derive the Truth Table

A	B	Cin	Cout	Sum
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

Designing a 1-bit Binary Adder

2. Find the generalized minterm expansion

Cout	Sum
0	0
0	1
0	1
1	0
0	1
1	0
1	0
1	1

Designing a 1-bit Binary Adder

2. Find the generalized minterm expansion

Cout	Sum
0	0
0	1
0	1
1	0
0	1
1	0
1	0
1	1

$$C_{out}(A, B, C_{in}) = \sum_m (3, 5, 6, 7)$$

Designing a 1-bit Binary Adder

2. Find the generalized minterm expansion

Cout	Sum
0	0
0	1
0	1
1	0
0	1
1	0
1	0
1	1

$$C_{out}(A, B, C_{in}) = \sum_m (3, 5, 6, 7)$$

$$Sum(A, B, C_{in}) = \sum_m (1, 2, 4, 7)$$

Designing a 1-bit Binary Adder

3. Populate the Karnaugh Map

$$C_{out}(A, B, C_{in}) = \sum_m (3, 5, 6, 7)$$

	B’		B	
A’	0	1	3	2
A	4	5	7	6
	C’	C		C’

Designing a 1-bit Binary Adder

4. Use the K-Map to get the optimized Boolean expression.

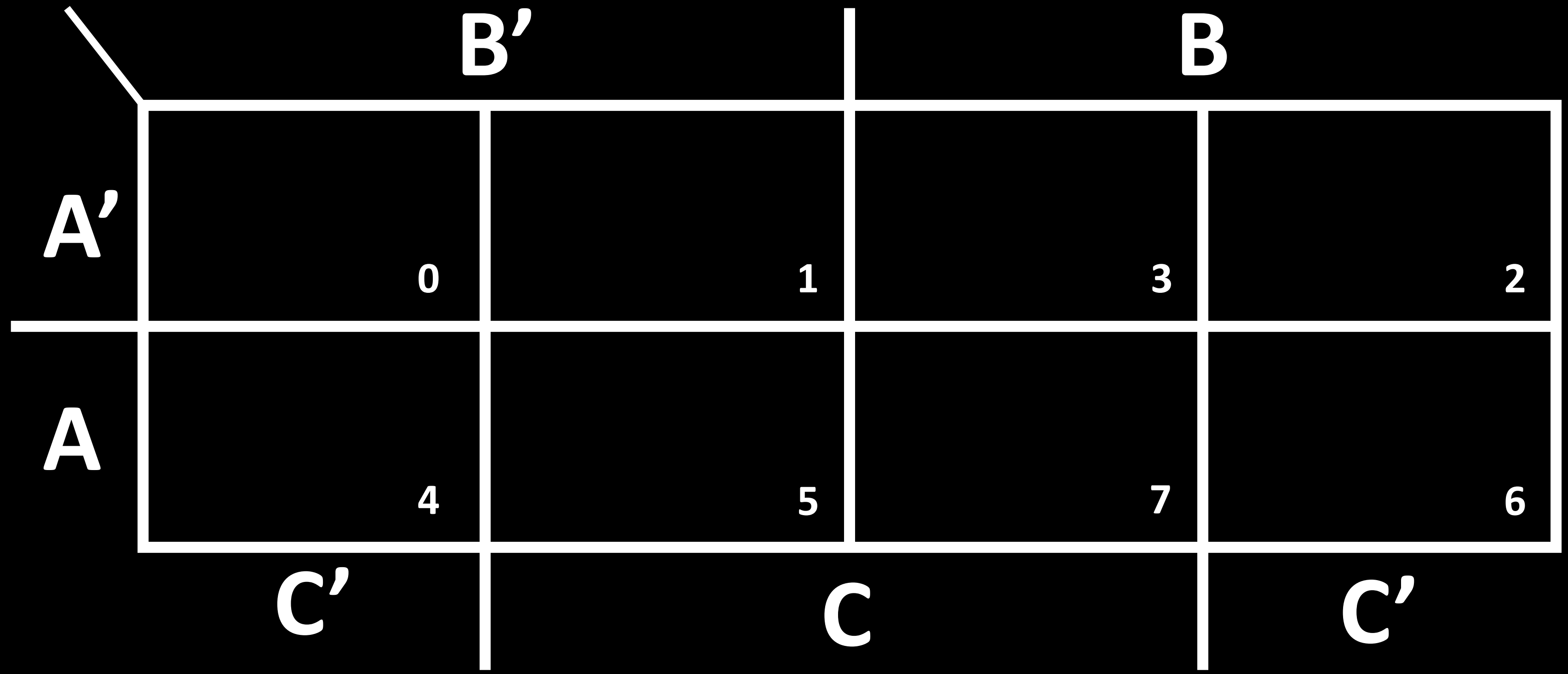
$$C_{out}(A, B, C_{in}) = \sum_m (3, 5, 6, 7)$$

	B'		B	
	C'	C	C	C'
A'	0 (0)	0 (1)	1 (3)	0 (2)
A	0 (4)	1 (5)	1 (7)	1 (6)

Designing a 1-bit Binary Adder

3. Populate the Karnaugh Map

$$Sum(A, B, C_{in}) = \sum_m (1, 2, 4, 7)$$



Designing a 1-bit Binary Adder

3. Populate the Karnaugh Map

$$Sum(A, B, C_{in}) = \sum_m (1, 2, 4, 7)$$

	B'		B	
A'	0 $_{0}$	1 $_{1}$	0 $_{3}$	1 $_{2}$
A	1 $_{4}$	0 $_{5}$	1 $_{7}$	0 $_{6}$
	C'	C		C'

Designing a 1-bit Binary Adder

4. Use the K-Map to get the optimized Boolean expression.

$$C_{out} = AC_{in}$$

	B’	B’	B	B
	C’	C	C	C’
A’	0 (0)	0 (1)	1 (3)	0 (2)
A	0 (4)	1 (5)	1 (7)	1 (6)

Designing a 1-bit Binary Adder

4. Use the K-Map to get the optimized Boolean expression.

$$C_{out} = AC_{in} + BC_{in}$$

	B’		B	
A’	0_{0}	0_{1}	1_{3}	0_{2}
A	0_{4}	1_{5}	1_{7}	1_{6}
	C’	C		C’

Designing a 1-bit Binary Adder

4. Use the K-Map to get the optimized Boolean expression.

$$C_{out} = AC_{in} + BC_{in} + AB$$

	B’		B	
A’	0 0	0 1	1 3	0 2
A	0 4	1 5	1 7	1 6
	C’	C	C	C’

Designing a 1-bit Binary Adder

4. Use the K-Map to get the optimized Boolean expression.

$$Sum(A, B, C_{in}) = \mathrm{AB}'C_{in}'$$

	B’		B	
A’	0 [0]	1 [1]	0 [3]	1 [2]
A	$\boxed{1}$ [4]	0 [5]	1 [7]	0 [6]
	C’	C		C’

Designing a 1-bit Binary Adder

4. Use the K-Map to get the optimized Boolean expression.

$$Sum(A, B, C_{in}) = \mathrm{AB'C'_{in}} + \mathrm{A'B'C_{in}}$$

	B'		B	
	C'	C	C	C'
A'	0 0	$\boxed{1}$ 1	0 3	1 2
A	$\boxed{1}$ 4	0 5	1 7	0 6

Designing a 1-bit Binary Adder

$$Sum(A, B, C_{in}) = \mathrm{AB'C'_{in}} + \mathrm{A'B'C_{in}}$$

	B'		B	
A'	0 0	$\boxed{1}$ 1	0 3	1 2
A	$\boxed{1}$ 4	0 5	1 7	0 6
	C'	C		C'

Designing a 1-bit Binary Adder

4. Use the K-Map to get the optimized Boolean expression.

$$Sum(A,B,C_{in}) = \mathrm{AB'C'_{in}} + \mathrm{A'B'C_{in}} + \mathrm{ABC_{in}}$$

	B'		B	
A'	0 (0)	$\boxed{1}$ (1)	0 (3)	1 (2)
A	$\boxed{1}$ (4)	0 (5)	$\boxed{1}$ (7)	0 (6)
	C'	C	C	C'

Designing a 1-bit Binary Adder

4. Use the K-Map to get the optimized Boolean expression.

$$Sum(A, B, C_{in}) = \mathrm{AB'C'_{in}} + \mathrm{A'B'C_{in}} + \mathrm{ABC_{in}} + \mathrm{A'BC'_{in}}$$

	B'		B	
A'	0 (0)	$\boxed{1}$ (1)	0 (3)	$\boxed{1}$ (2)
A	$\boxed{1}$ (4)	0 (5)	$\boxed{1}$ (7)	0 (6)
	C'	C	C	C'

Designing a 1-bit Binary Adder

$$Sum(A, B, C_{in}) = \mathrm{AB'C'_{\mathit{in}}} + \mathrm{A'B'C_{in}} + \mathrm{ABC_{in}} + \mathrm{A'BC'_{in}}$$

We can utilize some Boolean algebra to simplify this even more.

Designing a 1-bit Binary Adder

$$Sum(A, B, C_{in}) = \mathrm{AB'C'} + \mathrm{A'B'C} + \mathrm{ABC} + \mathrm{A'BC'}$$

We can utilize some Boolean algebra to simplify this even more.

$$Sum(A, B, C_{in}) = AB'C' + A'BC' + A'B'C + ABC$$

Designing a 1-bit Binary Adder

$$Sum(A, B, C_{in}) = \mathrm{AB'C' + A'B'C + ABC + A'BC'}$$

We can utilize some Boolean algebra to simplify this even more.

$$Sum(A, B, C_{in}) = AB'C' + A'BC' + A'B'C + ABC$$

$$Sum(A, B, C_{in}) = C'(AB' + A'B) + C(A'B' + AB)$$

Designing a 1-bit Binary Adder

$$Sum(A, B, C_{in}) = \mathrm{AB'C'} + \mathrm{A'B'C} + \mathrm{ABC} + \mathrm{A'BC'}$$

We can utilize some Boolean algebra to simplify this even more.

$$Sum(A, B, C_{in}) = AB'C' + A'BC' + A'B'C + ABC$$

$$Sum(A, B, C_{in}) = C'(AB' + A'B) + C(A'B' + AB)$$

$$Sum(A, B, C_{in}) = C'(A \oplus B) + C(A \oplus B)'$$

Designing a 1-bit Binary Adder

$$Sum(A, B, C_{in}) = \mathrm{AB'C'} + \mathrm{A'B'C} + \mathrm{ABC} + \mathrm{A'BC'}$$

We can utilize some Boolean algebra to simplify this even more.

$$Sum(A, B, C_{in}) = AB'C' + A'BC' + A'B'C + ABC$$

$$Sum(A, B, C_{in}) = C'(AB' + A'B) + C(A'B' + AB)$$

$$Sum(A, B, C_{in}) = C'(A \oplus B) + C(A \oplus B)'$$

$$Sum(A, B, C_{in}) = C \oplus (A \oplus B) = A \oplus B \oplus C$$

Designing a 1-bit Binary Adder

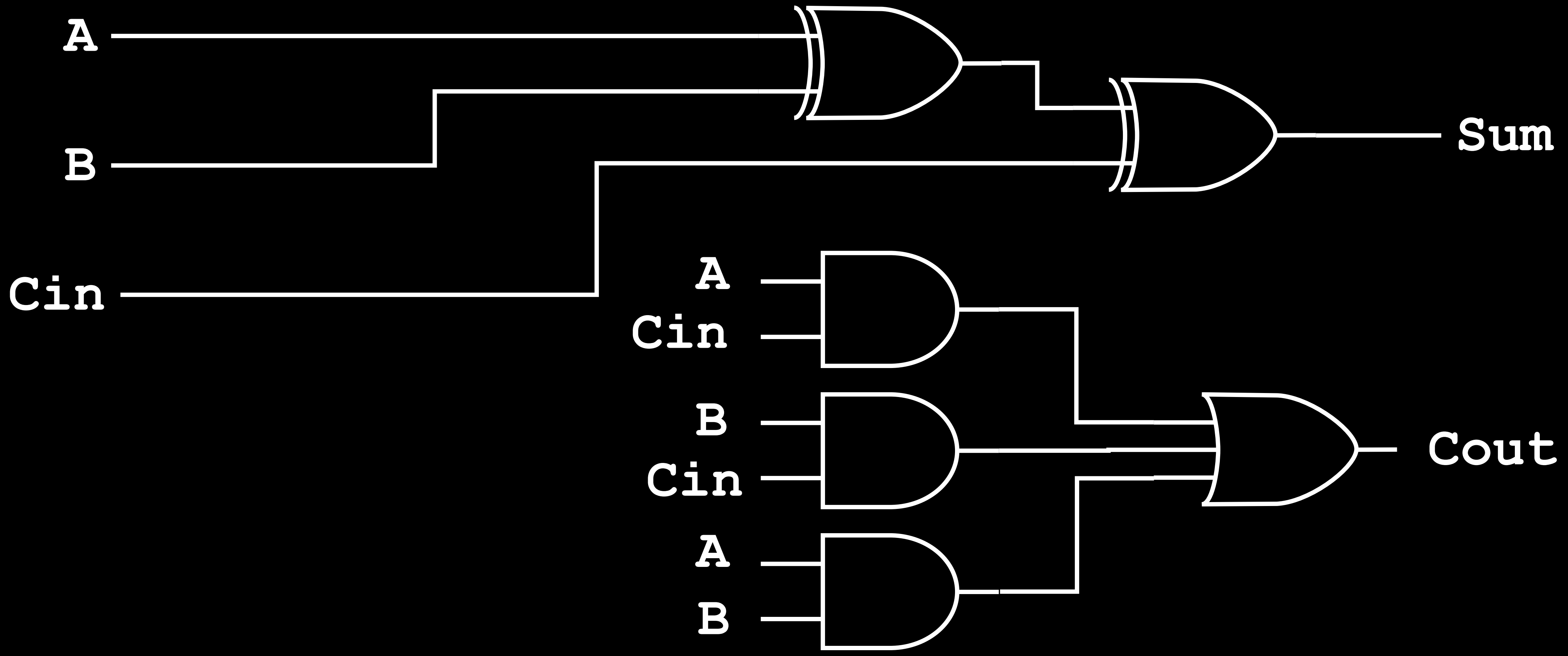
4. Use the K-Map to get the optimized Boolean expression.

$$Sum(A, B, C_{in}) = A \oplus B \oplus C$$

$$C_{out} = AC_{in} + BC_{in} + AB$$

5. Draw the Logic Circuit.

$$Sum(A, B, C_{in}) = A \oplus B \oplus C \qquad C_{out} = AC_{in} + BC_{in} + AB$$



What questions do you have about designing the 1-bit adder?

Exam Question | Designing a 1-bit Adder

What is the result of $1 \oplus 1 \oplus 1$?

A. 1

B. 0

Exam Question | Designing a 1-bit Adder

What is the result of $1 \oplus 1 \oplus 1$?

A. 1

~~B. 0~~

Exam Question | Designing a 1-bit Adder

What is the result of $1 \oplus 1 \oplus 1$?

A. 1

~~B. 0~~

$$1 \oplus 1 \oplus 1$$

Exam Question | Designing a 1-bit Adder

What is the result of $1 \oplus 1 \oplus 1$?

A. 1

~~B. 0~~

$$1 \oplus 1 \oplus 1$$

$$1 \oplus (1 \oplus 1)$$

Exam Question | Designing a 1-bit Adder

What is the result of $1 \oplus 1 \ \oplus 1$?

A. 1

~~B. 0~~

$$1 \oplus 1 \oplus 1$$

$$1 \oplus (1 \oplus 1)$$

$$1 \oplus 0$$

Exam Question | Designing a 1-bit Adder

What is the result of $1 \oplus 1 \oplus 1$?

A. 1

~~B. 0~~

$$1 \oplus 1 \oplus 1$$

$$1 \oplus (1 \oplus 1)$$

$$1 \oplus 0$$

$$1$$

The Ripple-Carry Adder

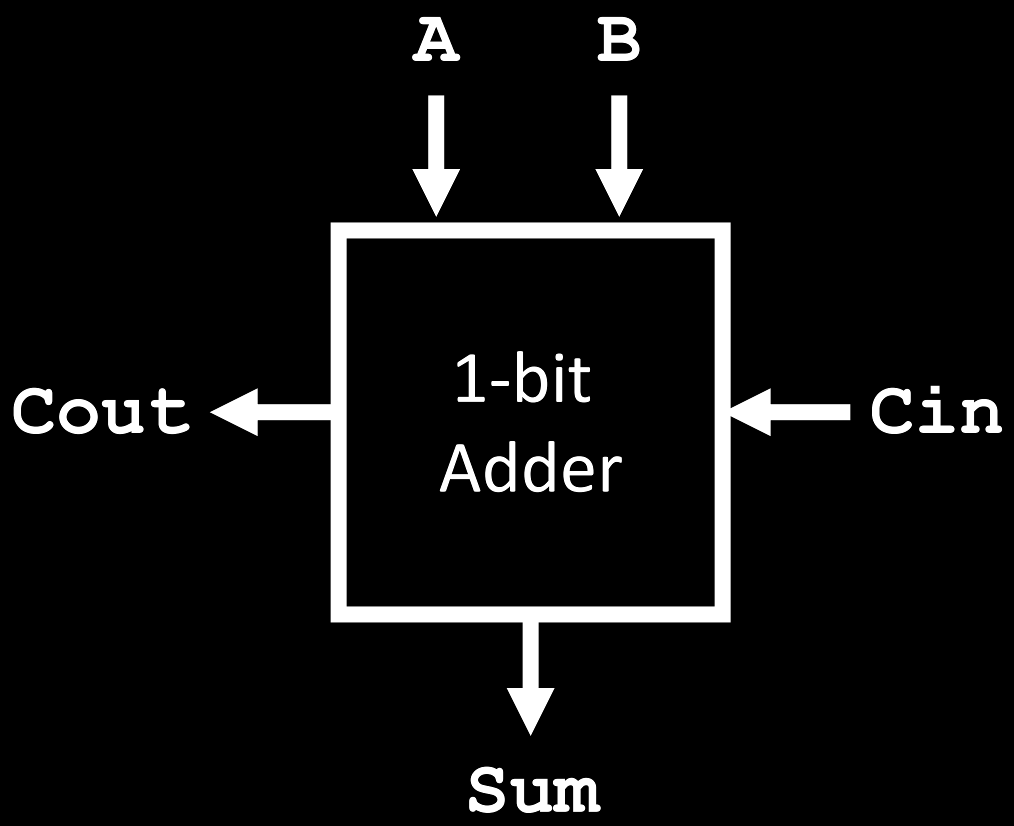
Binary Adders | Modern Digital System Design

The Ripple-Carry Adder

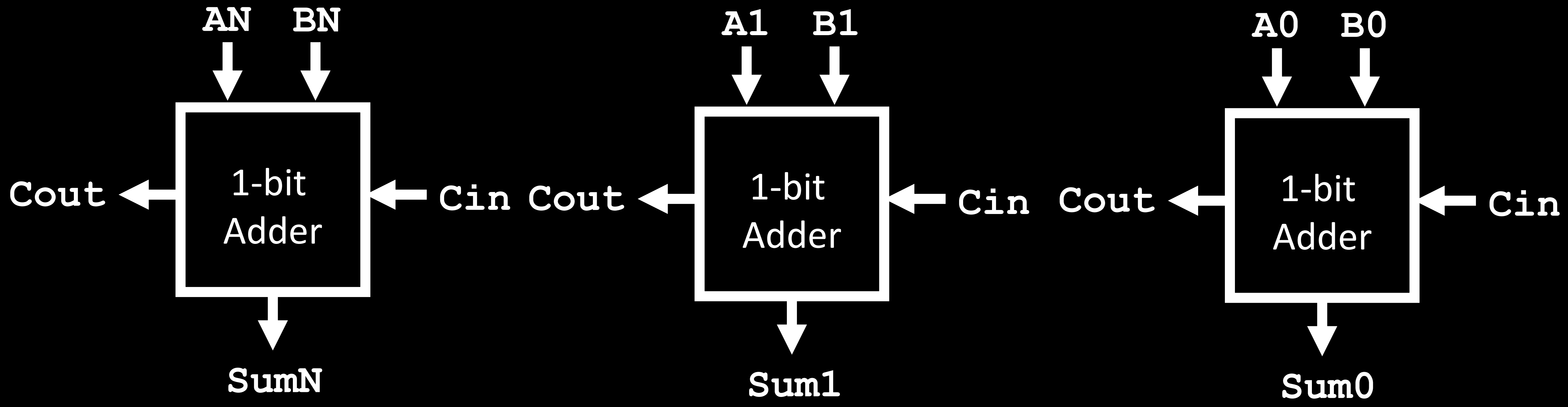
We can implement an N-bit adder by using an iterative design.

Use our 1-bit adder N-times.

The Ripple-Carry Adder



The Ripple-Carry Adder



What questions do you have about the Ripple-Carry Adder?

Exam Question | The Ripple-Carry Adder

How many wires will be required to implement a 4-bit ripple-carry adder in Verilog HDL?

A. 4
B. 3
C. 1
D. 0

Exam Question | The Ripple-Carry Adder

How many wires will be required to implement a 4-bit ripple-carry adder in Verilog HDL?

~~A. 4~~

B. 3

~~C. 1~~

~~D. 0~~

Exam Question | The Ripple-Carry Adder

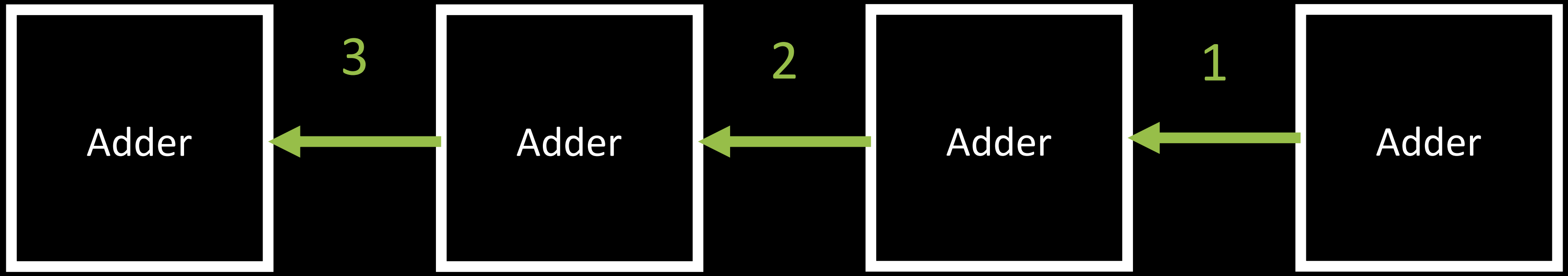
How many wires will be required to implement a 4-bit ripple-carry adder in Verilog HDL?

~~A. 4~~

B. 3

~~C. 1~~

~~D. 0~~



There is always 1 fewer connection between adders than the number of adders.

Implementation in Verilog HDL

Binary Adders | Modern Digital System Design

Implementation in Verilog HDL

Let’s implement an 8-bit ripple-carry adder.

adder.v

```
module adder(A, B, Cin, Cout, Sum);
    // ...
endmodule
```

adder.v

```
// DATA INPUTS
input A, B, Cin;

// DATA OUTPUTS
output Cout, Sum;
```

adder.v

```
// LOGIC CIRCUIT
assign Cout = A & Cin | B & Cin | A & B;
assign Sum = A ^ B ^ Cin;
```

adder.v

```
// LOGIC CIRCUIT
assign Cout = A & Cin | B & Cin | A & B;
assign Sum = A ^ B ^ Cin;
```

adder.v

```
module adder(A, B, Cin, Cout, Sum);

    // DATA INPUTS
    input A, B, Cin;

    // DATA OUTPUTS
    output Cout, Sum;

    // LOGIC CIRCUIT
    assign Cout = A & Cin | B & Cin | A & B;
    assign Sum = A ^ B ^ Cin;

endmodule
```

adder_test.v

```
Begin testbench for adder.v
A + B + Cin = Cout Sum
----------------------------
0 + 0 + 0 = 00
0 + 0 + 1 = 01
0 + 1 + 0 = 01
0 + 1 + 1 = 10
1 + 0 + 0 = 01
1 + 0 + 1 = 10
1 + 1 + 0 = 10
1 + 1 + 1 = 11
End of Test.
```

ripple_adder.v

```
`include "adder.v"
```

ripple_adder.v

```
module ripple_adder(A, B, Sum);
    // ...
endmodule
```

ripple_adder.v

```
// DATA INPUTS
input [7:0] A, B;

// DATA OUTPUTS
output [7:0] Sum;
```

ripple_adder.v

```
// Wires
wire [6:0] carry;
```

ripple_adder.v

```
// LOGIC IMPLEMENTATION
adder U0(.A(A[0]), .B(B[0]), .Cin(1'b0),      .Cout(carry[0]), .Sum(Sum[0]));
adder U1(.A(A[1]), .B(B[1]), .Cin(carry[0]), .Cout(carry[1]), .Sum(Sum[1]));
adder U2(.A(A[2]), .B(B[2]), .Cin(carry[1]), .Cout(carry[2]), .Sum(Sum[2]));
adder U3(.A(A[3]), .B(B[3]), .Cin(carry[2]), .Cout(carry[3]), .Sum(Sum[3]));
adder U4(.A(A[4]), .B(B[4]), .Cin(carry[3]), .Cout(carry[4]), .Sum(Sum[4]));
adder U5(.A(A[5]), .B(B[5]), .Cin(carry[4]), .Cout(carry[5]), .Sum(Sum[5]));
adder U6(.A(A[6]), .B(B[6]), .Cin(carry[5]), .Cout(carry[6]), .Sum(Sum[6]));
adder U7(.A(A[7]), .B(B[7]), .Cin(carry[6]), .Cout(        ), .Sum(Sum[7]));
```

ripple_adder_test.v

```
3 + 223 = 226
3 + 224 = 227
3 + 225 = 228
3 + 226 = 229
3 + 227 = 230
3 + 228 = 231
3 + 229 = 232
3 + 230 = 233
```

What questions do you have about the implementation in Verilog?

Exam Question | Implementation in Verilog

How many test cases are required to exhaustively test a 4-bit ripple-carry adder?

A. 16

B. 32

C. 64

D. 256

Exam Question | Implementation in Verilog

How many test cases are required to exhaustively test a 4-bit ripple-carry adder?

A. ~~16~~

B. ~~32~~

C. ~~64~~

D. 256

Exam Question | Implementation in Verilog

How many test cases are required to exhaustively test a 4-bit ripple-carry adder?

A. ~~16~~

B. ~~32~~

C. ~~64~~

D. 256

```
// DATA INPUTS
input [3:0] A, B;
```

Two, 4-bit inputs -> 8 total inputs.

Exam Question | Implementation in Verilog

How many test cases are required to exhaustively test a 4-bit ripple-carry adder?

~~A. 16~~

~~B. 32~~

~~C. 64~~

D. 256

```
// DATA INPUTS
input [3:0] A, B;
```

Two, 4-bit inputs -> 8 total inputs.

$$Test\ Cases = 2^N = 2^8 = 256$$

Lecture Recap

- Binary Addition
- Designing a 1-bit Binary Adder
- The Ripple-Carry Adder
- Implementation in Verilog HDL

Binary Adders

ECE 2372 | Modern Digital System Design | Texas Tech University