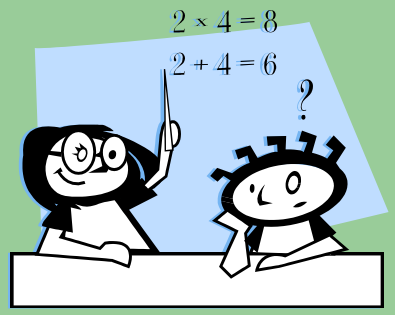
Lecture 04

Modeling, Analysis and Simulation in Logic Design

逻辑设计中的建模、分析与仿真



Dr. Yinong Chen

IoT & Robotics Education Laboratory

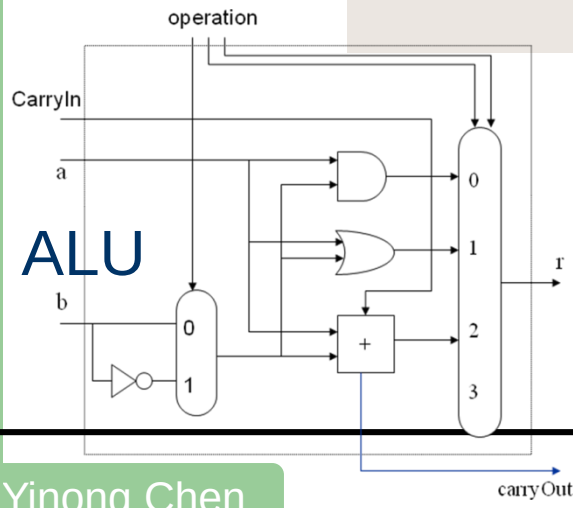
Engineering Design Process
工程设计过程

定义问题 研究 勾画可能的解答

Identify and define problem → research → sketch possible solutions

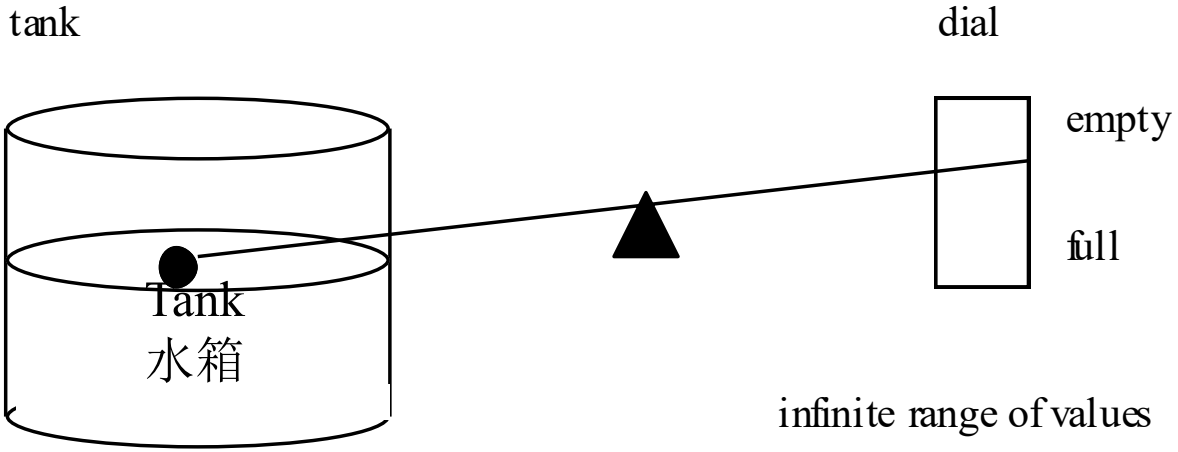
建模 Modeling → 分析 Analysis → 仿真 Simulation → 原型 Prototyping

最终选择 Final Selection → 实现和测试 Implementation and testing

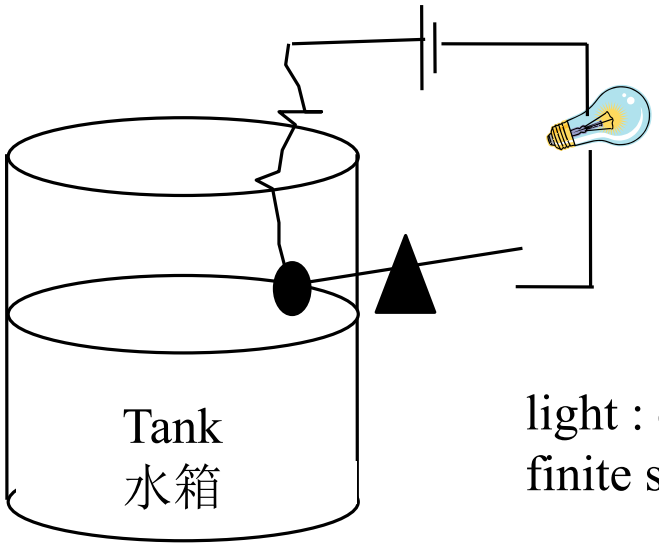


Contents 内容

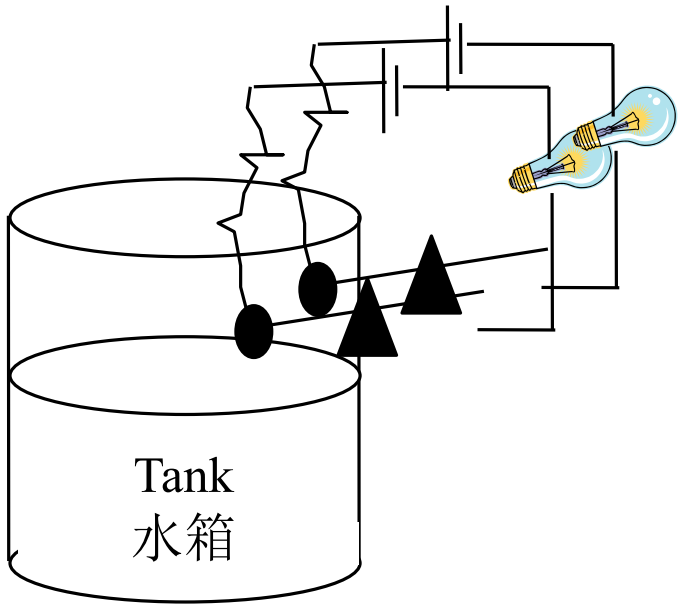
Logic Design: Analogue versus Digital
逻辑设计： 模拟与数字



infinite range of values



light : on / off
finite set of values



Propositional Logic and its Elements

命题逻辑及其构成要素

- Propositional logic is a language for modeling and specification 命题逻辑是一种建模和规范语言。
- Proposition: A statement that can either be true or false: 命题：一个语句，它的的值可以是真，或是假
  - One plus two is three
  - There are two Nobel prize winners at Arizona State University
  - The sky is blue
  - How old are you?
  - You must ride a bike to school!
- Logic connectives
  - AND ($\wedge$), OR ($\vee$), NOT ($\neg$), IMPLIES ($\rightarrow$)
  - light is on $\rightarrow$ tank is full
  - (Light is off) $\wedge$ (bulb is not broken) $\rightarrow$ tank is empty
- Truth and falsity values – Truth Table 真值表

真值表 是逻辑规范/电路模型
Truth Table is a Logic Specification/Model for Circuits

NOT

propositional variable	*statement*
P	¬ P
false	*true*
true	*false*

a	$\bar{a}$
0	1
1	0

AND

P	Q	P ∧ Q
false	*false*	*false*
false	*true*	*false*
true	*false*	*false*
true	*true*	*true*

a	b	a∧b
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table 真值表 (Contd.)

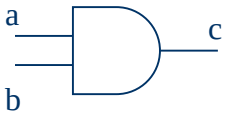
OR

P	Q	$P \vee Q$
false	*false*	*false*
false	*true*	*true*
true	*false*	*true*
true	*true*	*true*

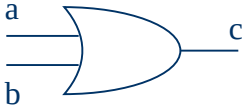
a	b	a∨b
0	0	0
0	1	1
1	0	1
1	1	1

Building Blocks of Digital Circuits
数字线路的基本构件

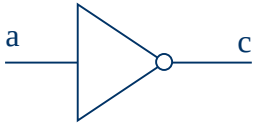
Building Blocks



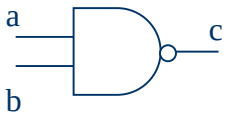
AND gate
$c = a \wedge b$



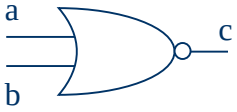
OR gate
$c = a \vee b$



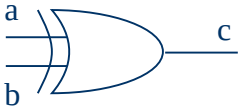
NOT
$c = \bar{a}$



NAND gate
$c = \overline{a \wedge b}$



NOR gate
$c = \overline{a \vee b}$



XOR gate
$c = \bar{a} \wedge b \vee a \wedge \bar{b}$

Truth Tables

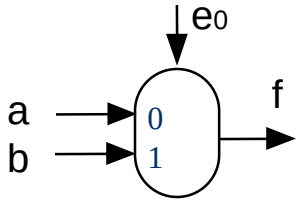
a	b	a∧b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a∨b
0	0	0
0	1	1
1	0	1
1	1	1

a	$\bar{a}$
0	1
1	0

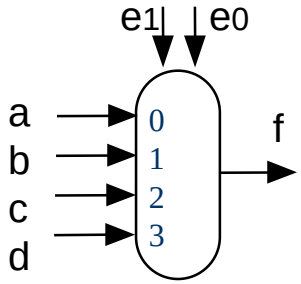
a	b	$\overline{a \wedge b}$	$\overline{a \vee b}$	$\bar{a} \wedge b \vee a \wedge \bar{b}$
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	0

2-1 Multiplexor



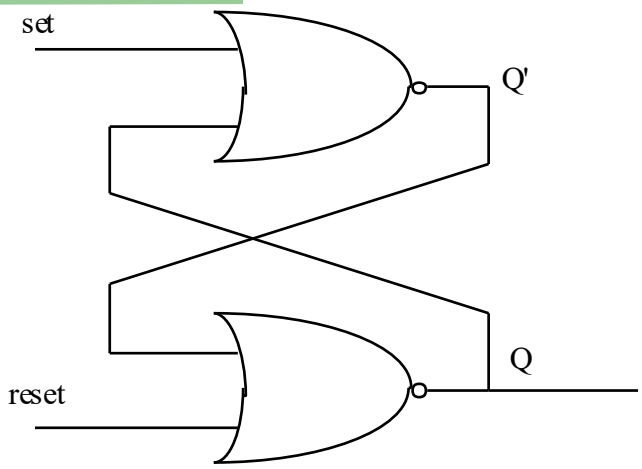
If (e_0=0) (f=a) else (f=b);

4-1 Multiplexor

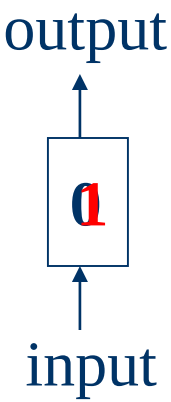


e_1	e_0	f
0	0	a
0	1	b
1	0	c
1	1	d

Memory Design 存储器设计: bit and Byte

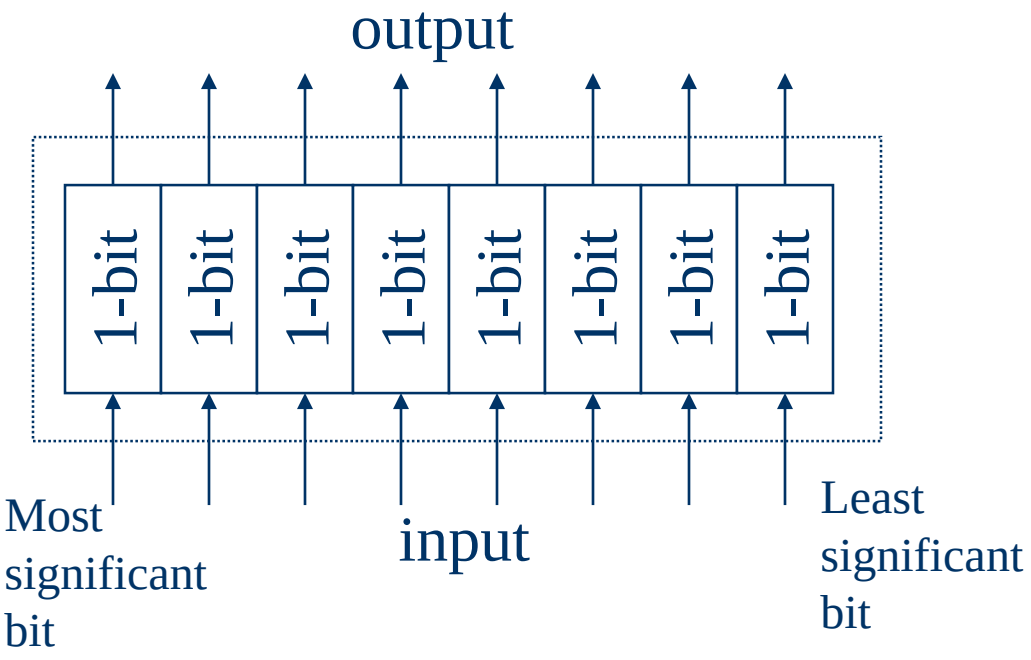


One-bit memory design



One bit can store a "1" or a "0"

8 bits = 1 Byte
Can store
a "character"
Or a short *integer*



Memory: Word, Long Word
存储器： 字，长字

In a 32-bit computer:

4 Bytes = 1 word and 8 bytes = 1 double word

Word 字

Can store:
int and *float*

double word 双字

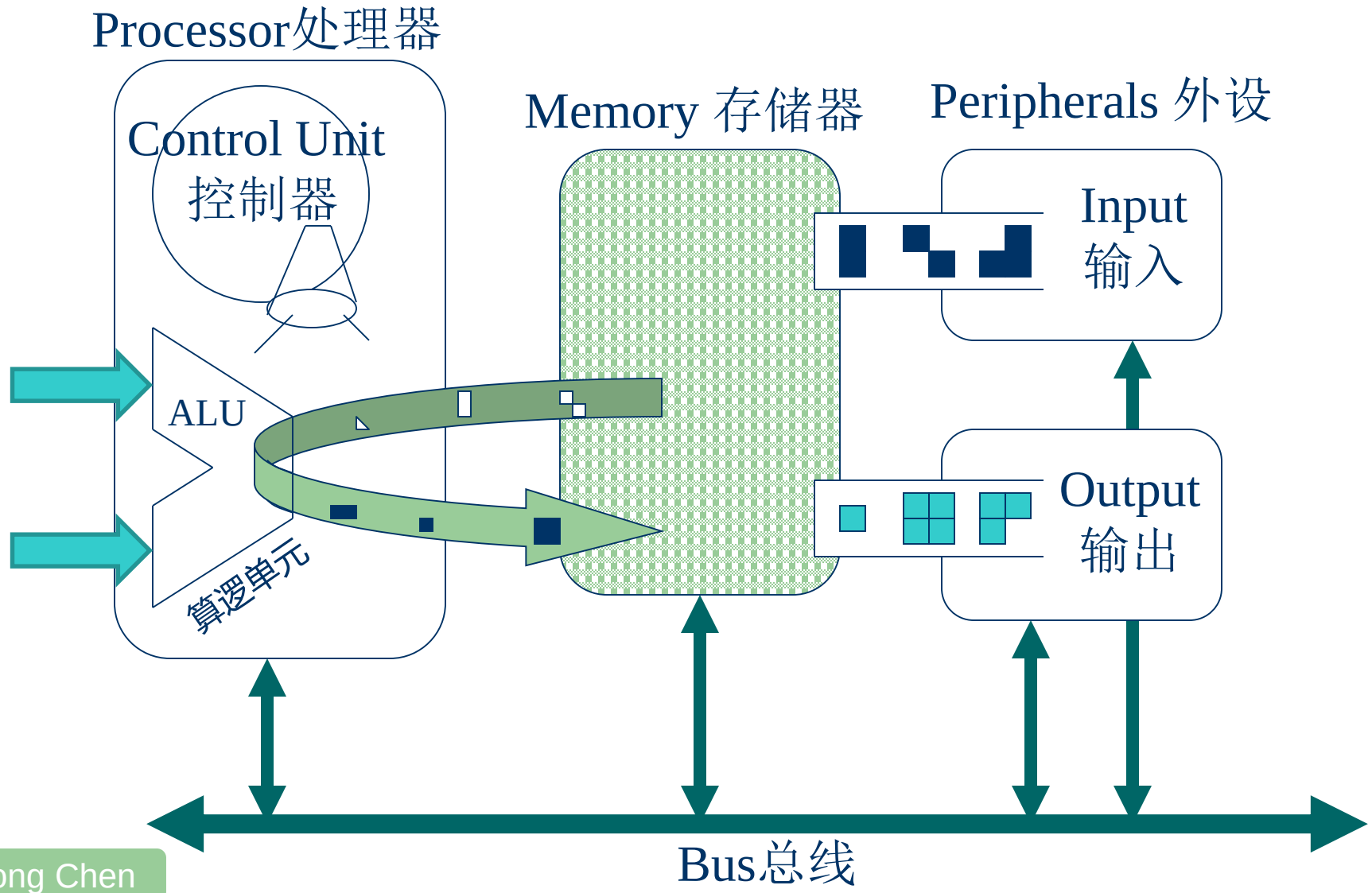
Can store: a *long int* and a *double float*

The Entire Memory, with 32-bit address space and byte-addressable

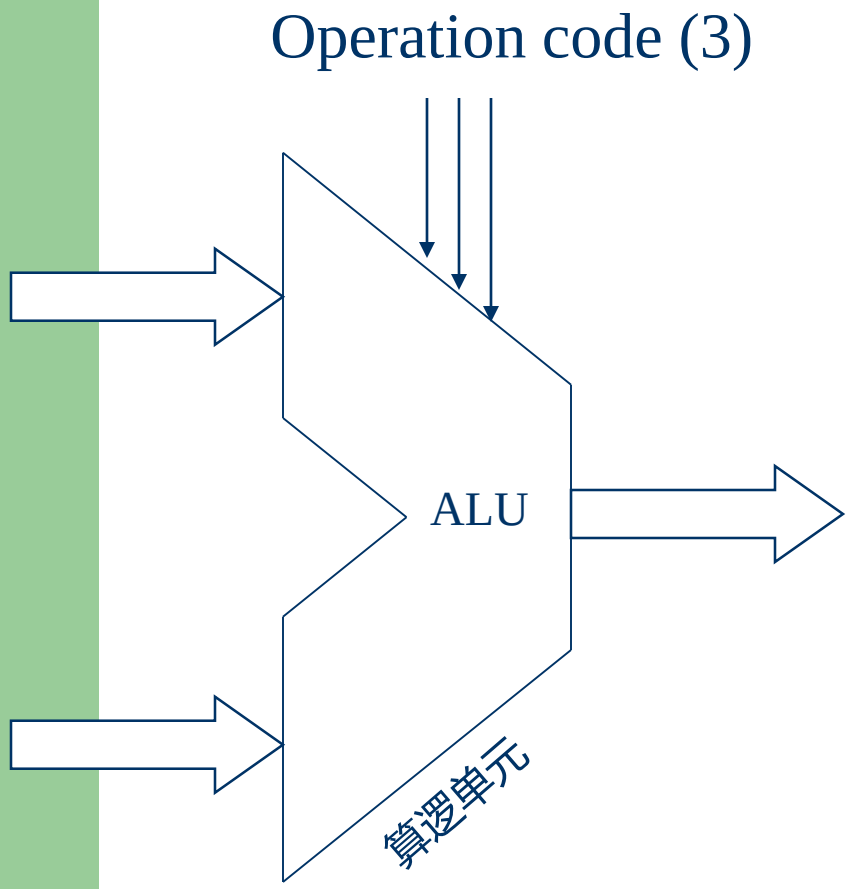
整个内存，32位地址空间和字节可寻址

Hex address	31 30 29			2 1 0
00000000h	1 byte	1 byte	1 byte	1 byte
00000004h	1 byte	1 byte	1 byte	1 byte
00000008h	1 byte	1 byte	1 byte	1 byte
0000000Ch	1 byte	1 byte	1 byte	1 byte
00000010h	1 byte	1 byte	1 byte	1 byte
00000014h	1 byte	1 byte	1 byte	1 byte
00000018h	1 byte	1 byte	1 byte	1 byte
0000001Ch	1 byte	1 byte	1 byte	1 byte
00000020h	1 byte	1 byte	1 byte	1 byte
00000024h	1 byte	1 byte	1 byte	1 byte
00000028h	1 byte	1 byte	1 byte	1 byte
0000002Ch	1 byte	1 byte	1 byte	1 byte
00000030h	1 byte	1 byte	1 byte	1 byte
. . .	. . .	. . .	. . .	. . .
FFFFFFF0h	1 byte	1 byte	1 byte	1 byte
FFFFFFF4h	1 byte	1 byte	1 byte	1 byte
FFFFFFF8h	1 byte	1 byte	1 byte	1 byte
FFFFFFFCh	1 byte	1 byte	1 byte	1 byte

Five Component Model of a Computer -- A Conceptual Model
计算机的5-部件概念模型



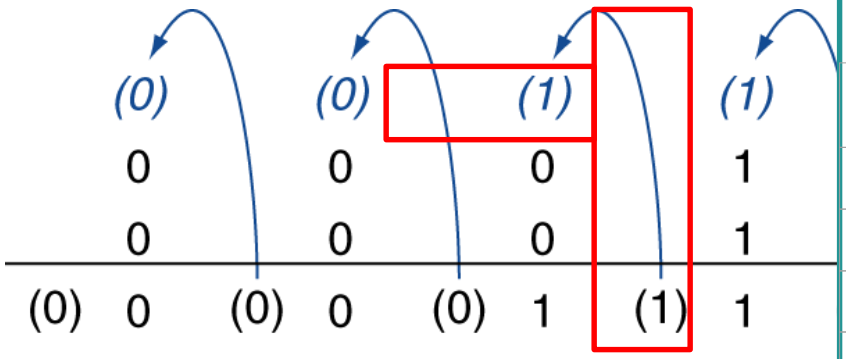
Arithmetic/Logic Unit (ALU)
算术/逻辑单元



操作码 Operation code	功能 function
0 0 0	AND
0 0 1	OR
0 1 0	ADD
1 1 0	SUB

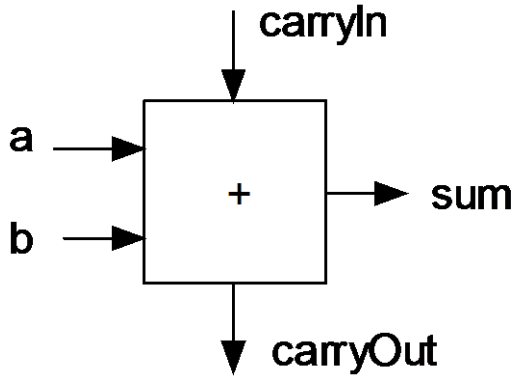
Adder

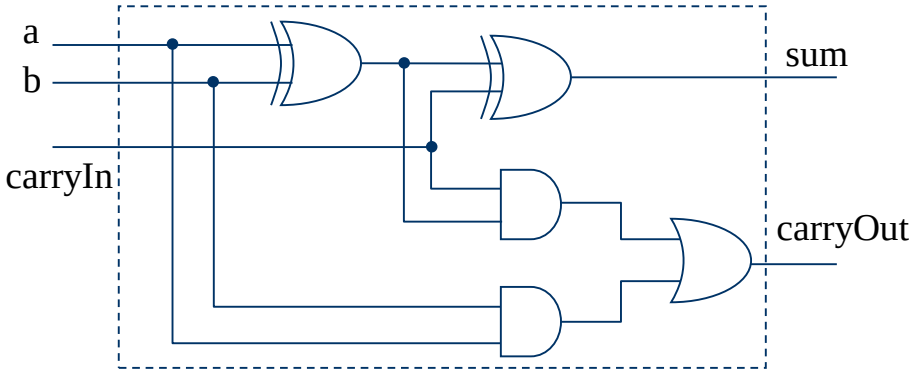
Truth Table and Design of a One-Bit Adder
真值表和1-位加法器的设计



Truth Table

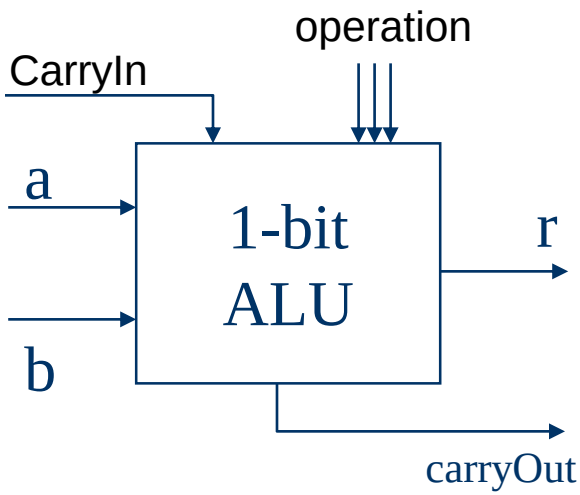
input			output	
a	b	carryIn	carryOut	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





One-bit adder

One-Bit ALU Design 1位-ALU的设计



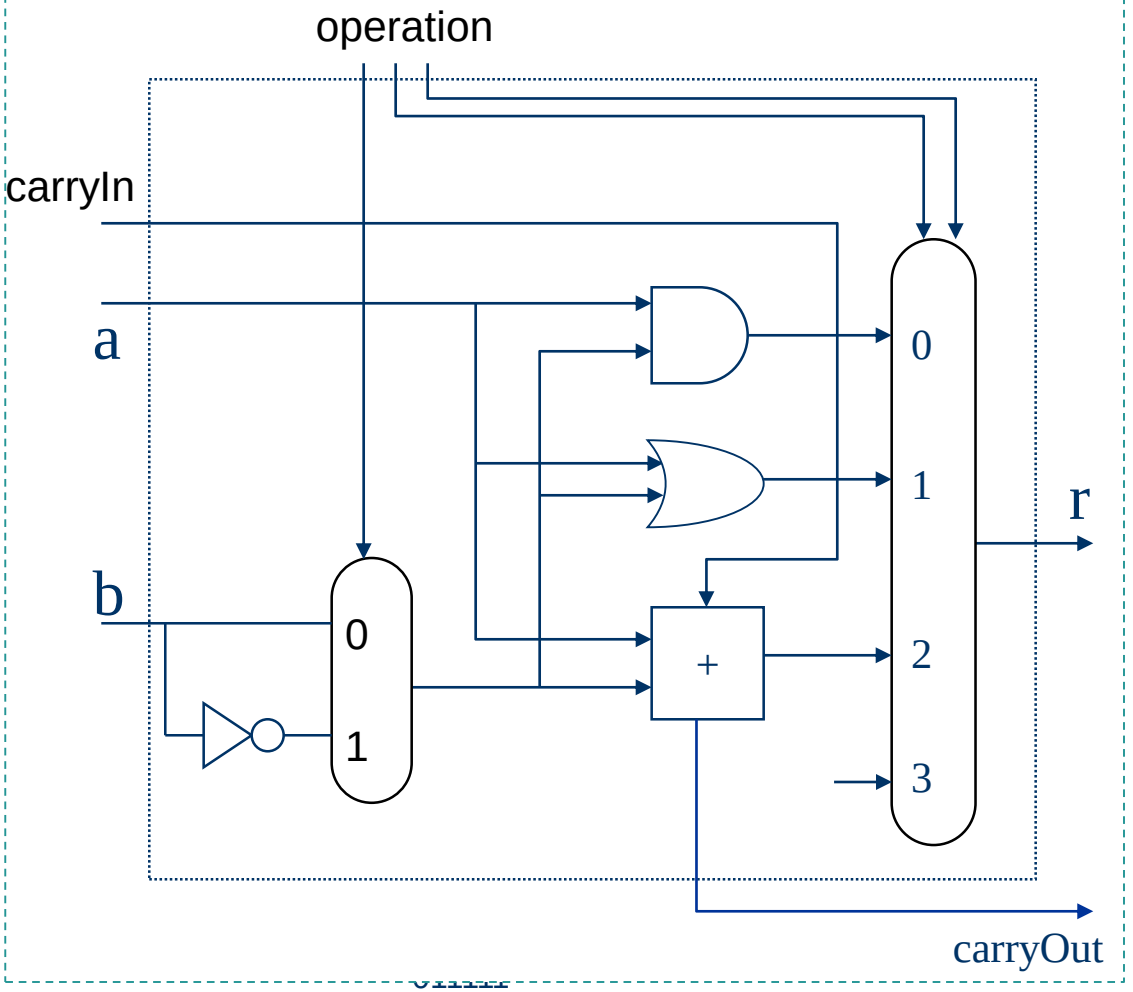
Six inputs and two outputs

Partial Truth Table

operation	function
0 0 0	AND
0 0 1	OR
0 1 0	ADD
1 1 0	SUB

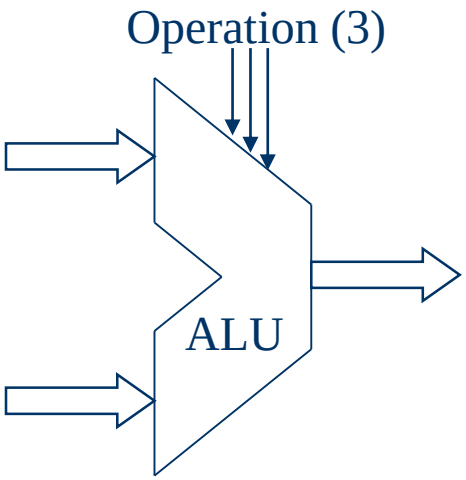
Inputs	carryOut	r
000001	d	0
000010	d	0
100000		

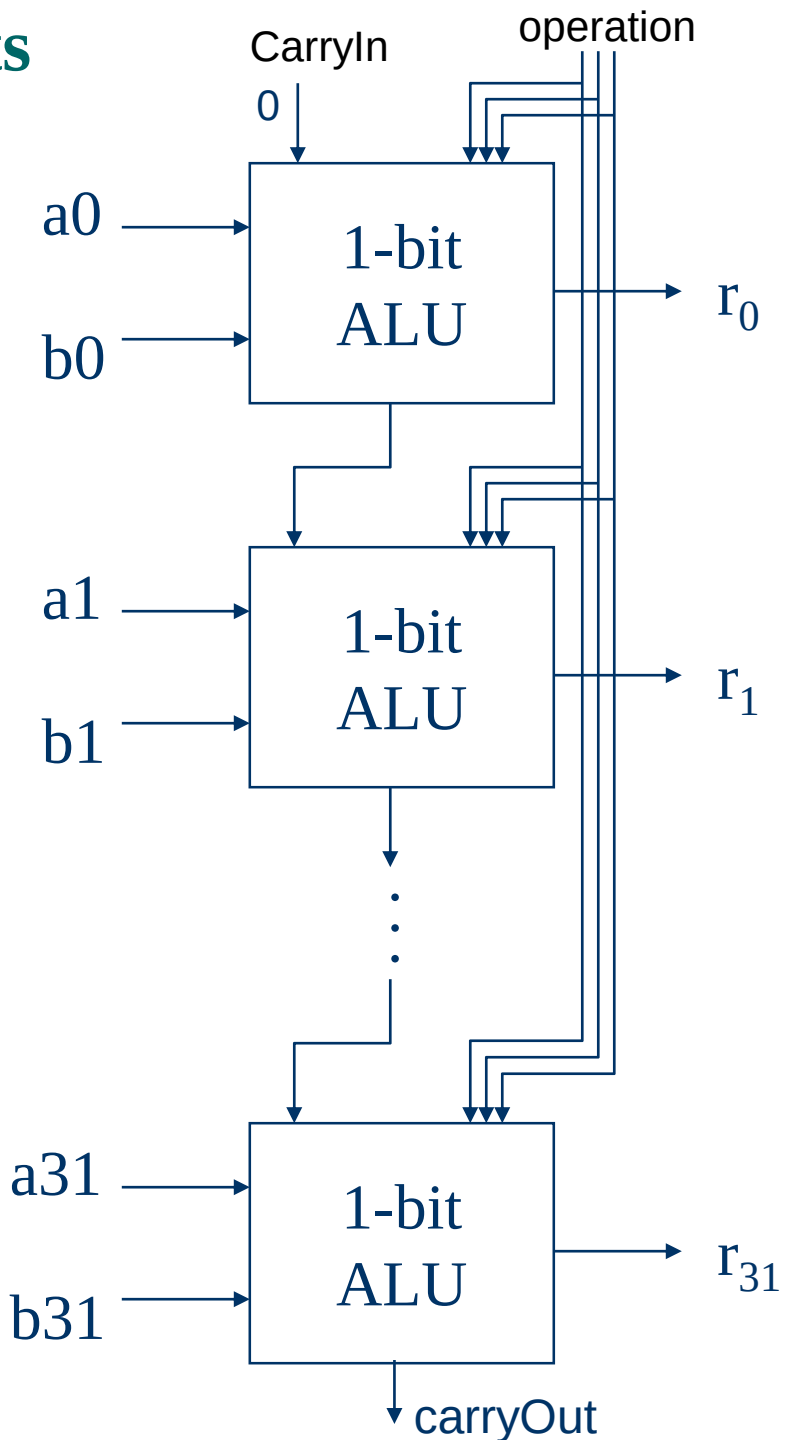
Component-based design



32-Bit ALU with 96 Inputs
32-ALU, 有96条输入线

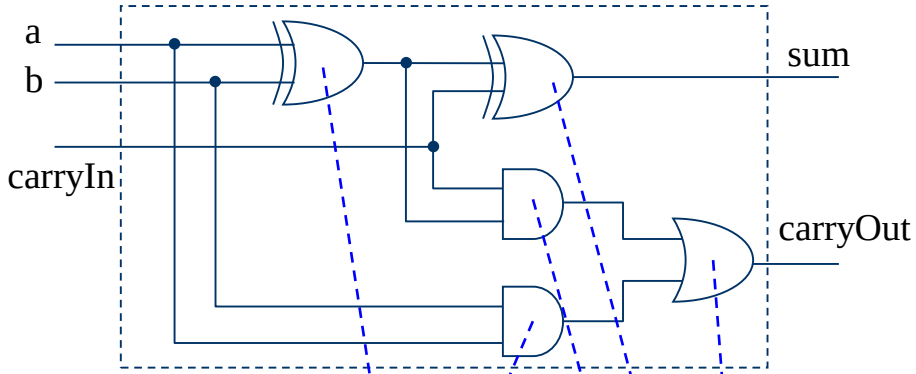
operation	function
0 0 0	AND
0 0 1	OR
0 1 0	ADD
1 1 0	SUB



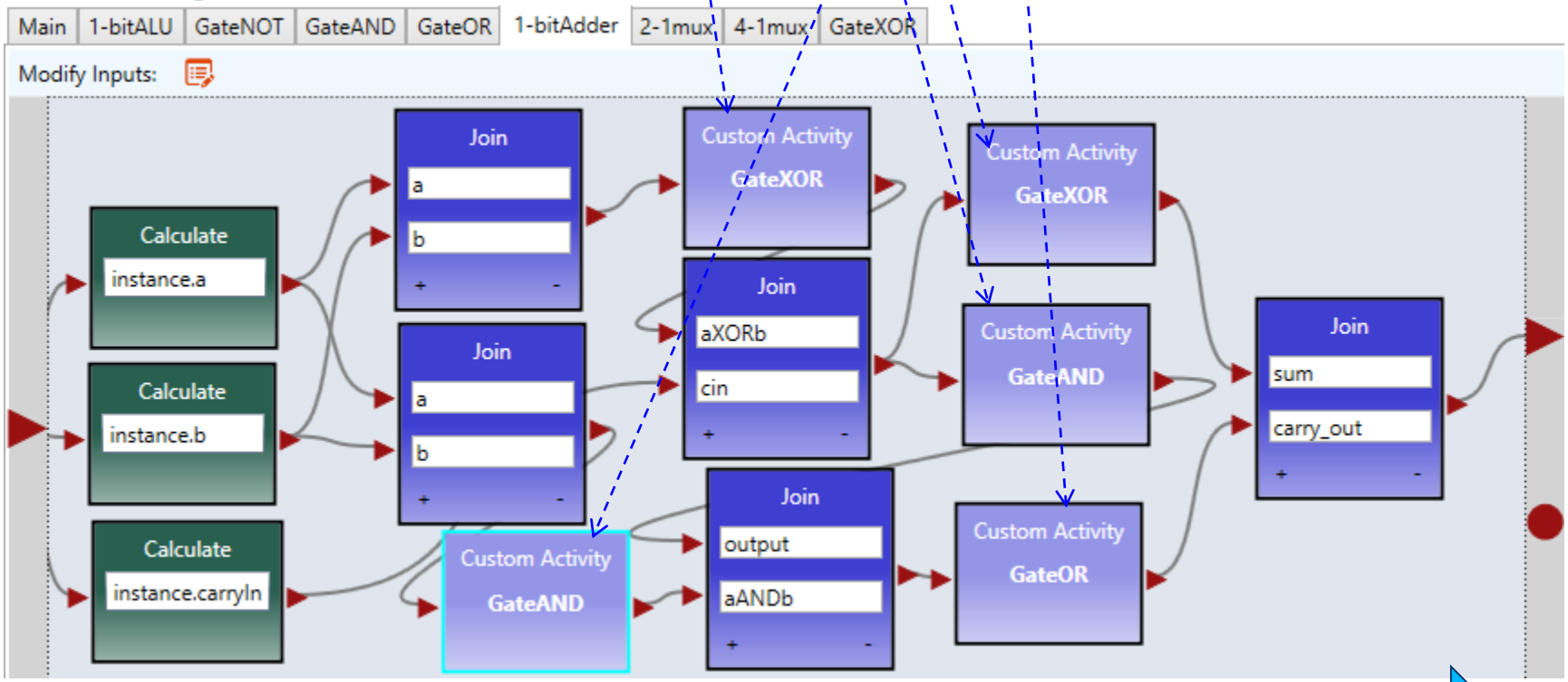


Simulation of 1-bit Adder 1-位计算器仿真

One-bit adder
logic design

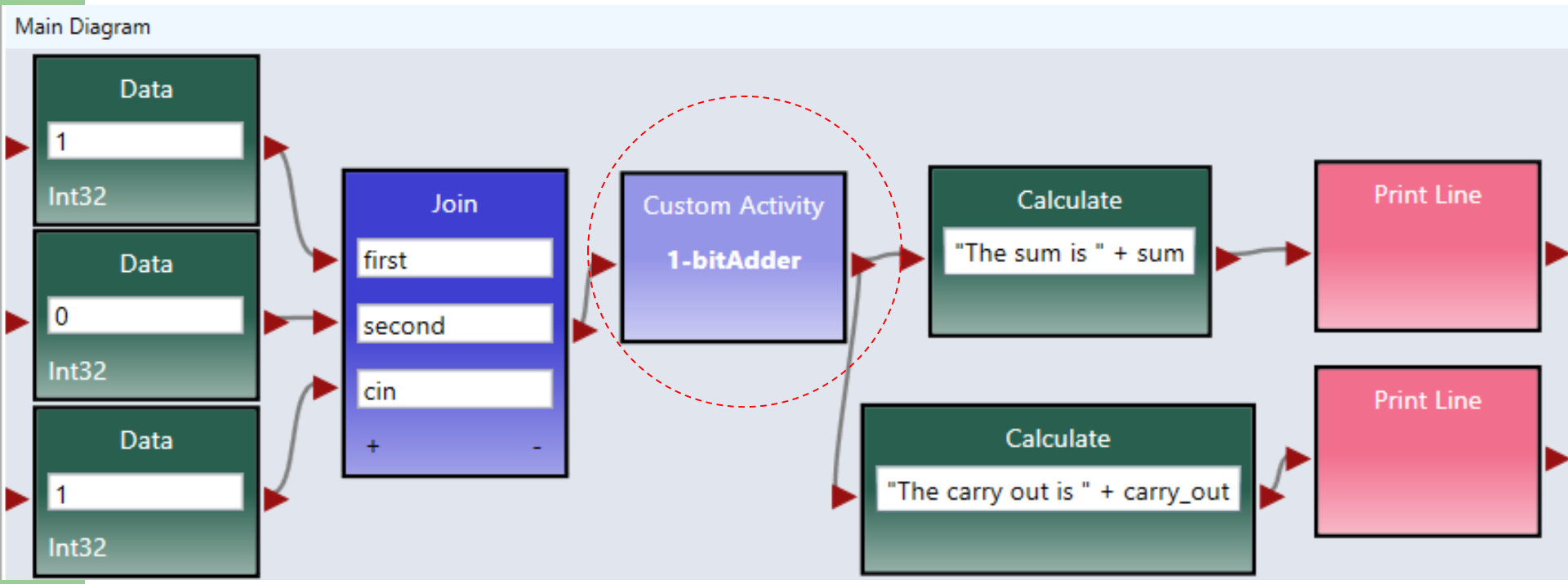


VIPLE Implementation



Define Before Using: Define the input whenever you see a warning sign

Testing the One-Bit Adder 测试1-位加法器



Converting Decimal to Binary Pattern for Automated Test Case Generation
为自动测试输入生成，把二进制转换成十进制

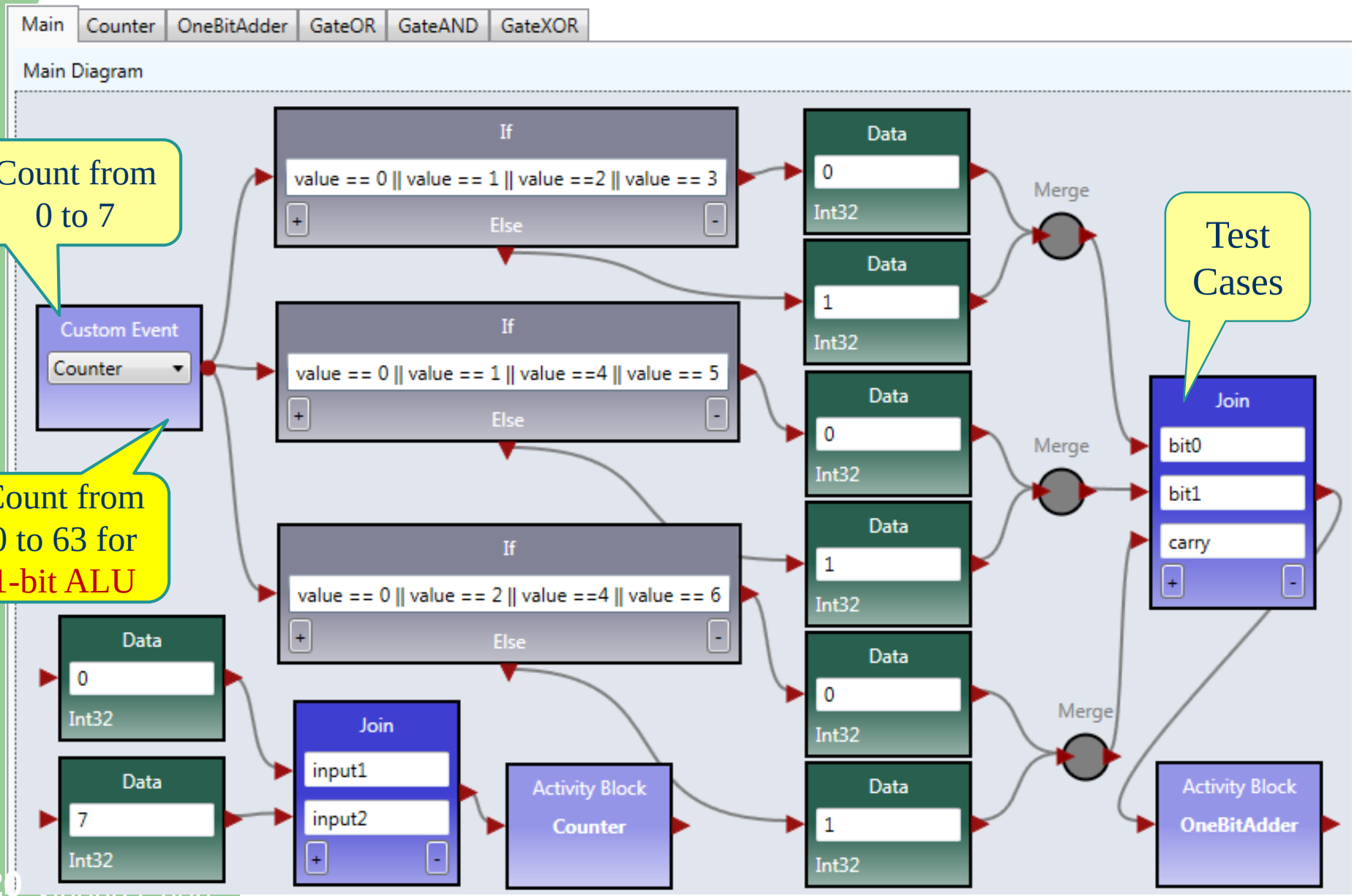
CountTo7	a	b	carryIn
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

if CountTo7 = 0, 1, 2, 3, then a = 0, else a = 1;

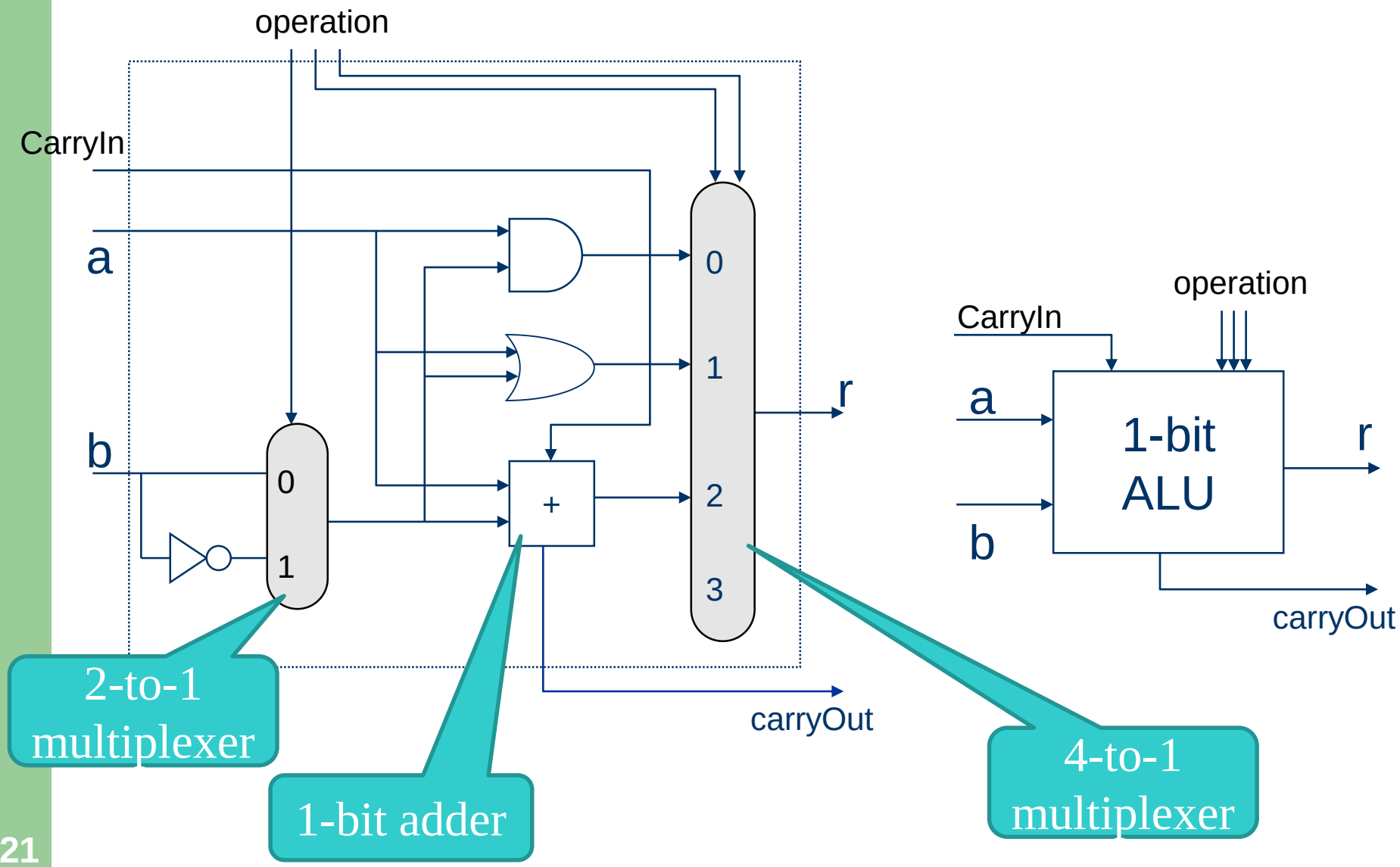
if CountTo7 = 0, 1, 4, 5, then b = 0, else b = 1;

if CountTo7 = 0, 2, 4, 6, then carryIn = 0, else carryIn = 1;

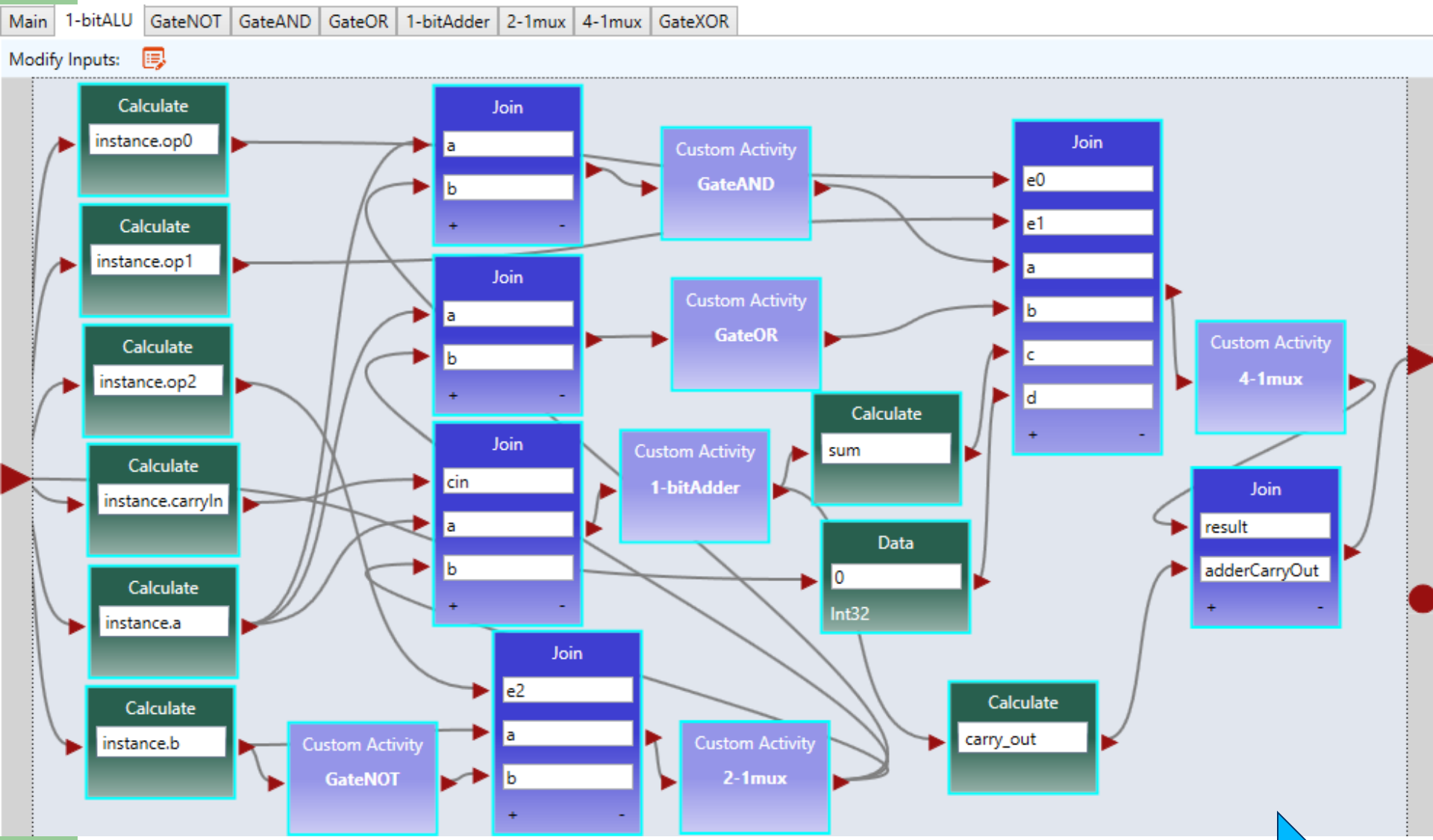
Automated Test Case Generation
自动测试输入生成



One-Bit ALU 1-位算逻部件的设计

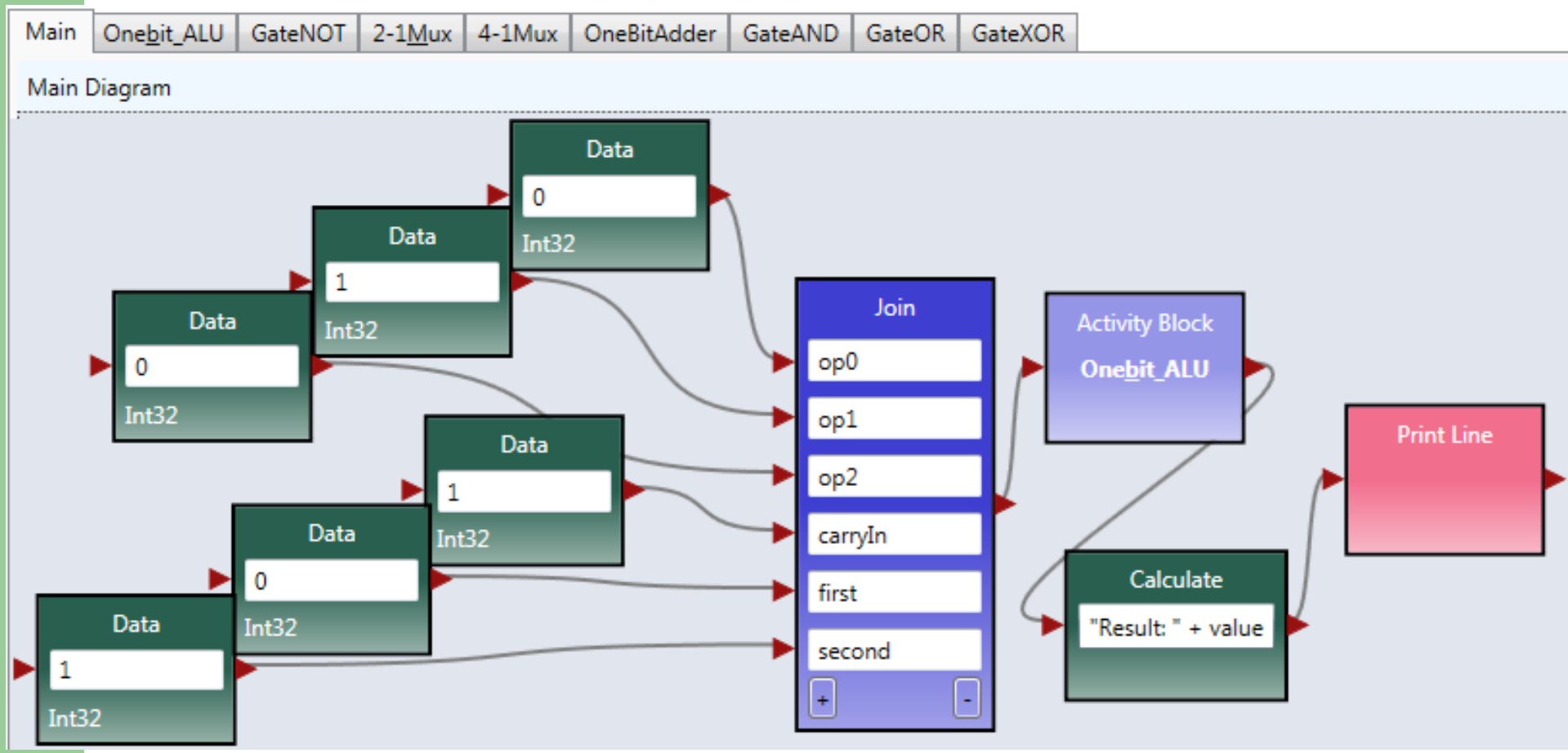


One-Bit ALU VIPLE Implementation
1-位算逻部件的实现



Define Before Using: Define the input whenever you see a warning sign

Testing the One-Bit ALU 测试 1-位算逻部件的



Automated Test Case Generation
自动测试输入生成