Lecture 2

or “Something clever”

Review from last time

Let's convert the following to binary and to hex:

91_{10}

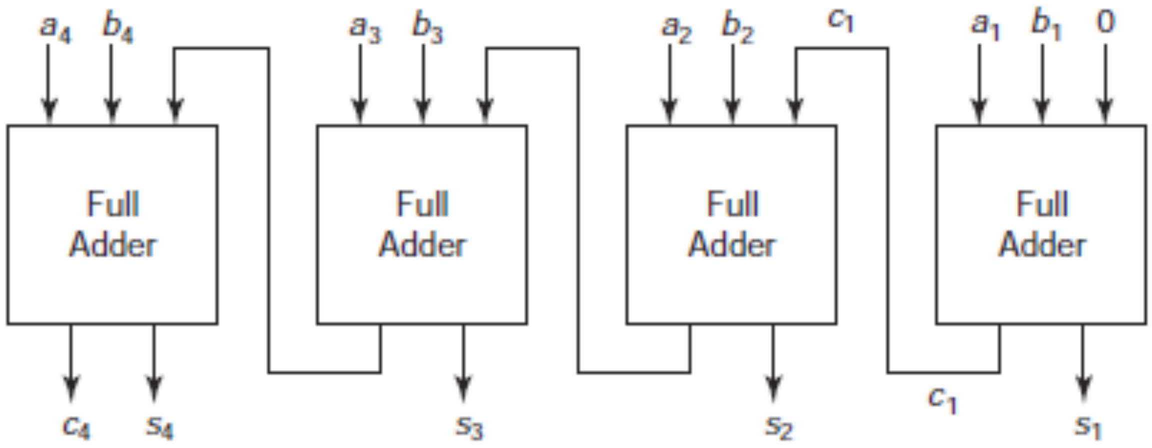
QUIZ!

Convert the following to binary and hex:

123_{10}

Finishing Adders

Figure 1.2 A 4-bit adder.



- Let’s say we have two 4 bit numbers and we are adding them together. We are storing the answer inside of another 4 bit number. Is this a problem?

Let’s try it

1111

0111 +

———

4 bits

Overflow

- If we add A and B together, and there is a carry on the last bit, where does it go?
- This is called overflow. It occurs when an arithmetic operation is out of range and indicates an error
- If 2 n-bit numbers are added together and they produce an (n+1) bit result, it is called overflow

Binary Coded Decimal (BCD)

- As we have said before, most computers operate on binary numbers
- People can’t (normally) read these numbers, so the computer has to:
  - On input: Convert from Decimal to Binary
  - On output: Convert from Binary to Decimal
- Decimal output still needs to be codes into binary, digit by digit

Table 1.7 Binary-coded decimal codes.

Decimal digit	8421 code	5421 code	2421 code	Excess 3 code	2 of 5 code
0	0000	0000	0000	0011	11000
1	0001	0001	0001	0100	10100
2	0010	0010	0010	0101	10010
3	0011	0011	0011	0110	10001
4	0100	0100	0100	0111	01100
5	0101	1000	1011	1000	01010
6	0110	1001	1100	1001	01001
7	0111	1010	1101	1010	00110
8	1000	1011	1110	1011	00101
9	1001	1100	1111	1100	00011
unused	1010	0101	0101	0000	any of
	1011	0110	0110	0001	the 22
	1100	0111	0111	0010	patterns
	1101	1101	1000	1101	with 0, 1,
	1110	1110	1001	1110	3, 4, or 5
	1111	1111	1010	1111	1's

Other codes

- ASCII: Alphanumeric information
- Gray code: Consecutive numbers differ by only 1 bit
  - Useful in coding the position of a continuous device and error detection

Table 1.8 ASCII code.

	$a_6a_5a_4$					
$a_3a_2a_1a_0$	010	011	100	101	110	111
0000	space	0	@	P	`	p
0001	!	1	A	Q	a	q
0010	“	2	B	R	b	r
0011	#	3	C	S	c	s
0100	$	4	D	T	d	t
0101	%	5	E	U	e	u
0110	&	6	F	V	f	v
0111	‘	7	G	W	g	w
1000	(	8	H	X	h	x
1001	)	9	I	Y	i	y
1010	*	:	J	Z	j	z
1011	+	;	K	[	k	{
1100	,	<	L	\	l	|
1101		=	M	]	m	}
1110	.	>	N	^	n	~
1111	/	?	O	_	o	delete

Let’s try it

Lets code the word “Logic” into ASCII

Table 1.9 Gray code.

Number	Gray code	Number	Gray code
0	0000	8	1100
1	0001	9	1101
2	0011	10	1111
3	0010	11	1110
4	0110	12	1010
5	0111	13	1011
6	0101	14	1001
7	0100	15	1000

Done with Ch. 1!

What did we learn?

- What digital systems are
- Truth tables for systems
- Number systems
  - Binary, decimal, hexadecimal
  - Conversion between these
- Binary addition and adders
- Overflow and it’s effects
- Binary Coded Decimal, ASCII, Gray code

What we didn’t cover

These topics are also in Ch. 1, but we didn’t cover them in lecture

- Signed numbers and two’s compliment
- Binary Subtraction

Some help

It will probably be useful to go through the solved problems at the end of the first chapter

If you still need help, stick a question in the box and I will email you some help

Chapter 2!

This chapter is about combinational systems

Our goals are to:

- Develop the tools to specify combinational systems
- Develop an algebraic approach for the description, simplification, and implementation of combinational systems

Continuing Examples

- A system with 4 inputs, A, B, C, and D, and one output, Z, such that Z=1 if three of the inputs are 1
- A system to do 1 bit of binary addition. It has 3 inputs (the 2 bits to be added plus the carry from the next lower order bit) and produces two outputs, a sum bit and a carry to the next higher order position

More examples

A system with 9 inputs, representing two 4-bit binary numbers and a carry input, and one 5 bit output, representing the sum

Design process for Combinational Systems

Step 1: Represent each of the inputs and output in binary

Step 1.5: If necessary, break the problem into smaller subproblems

Step 2: Formalize the design specification either in the form of a truth table or of an algebraic expression

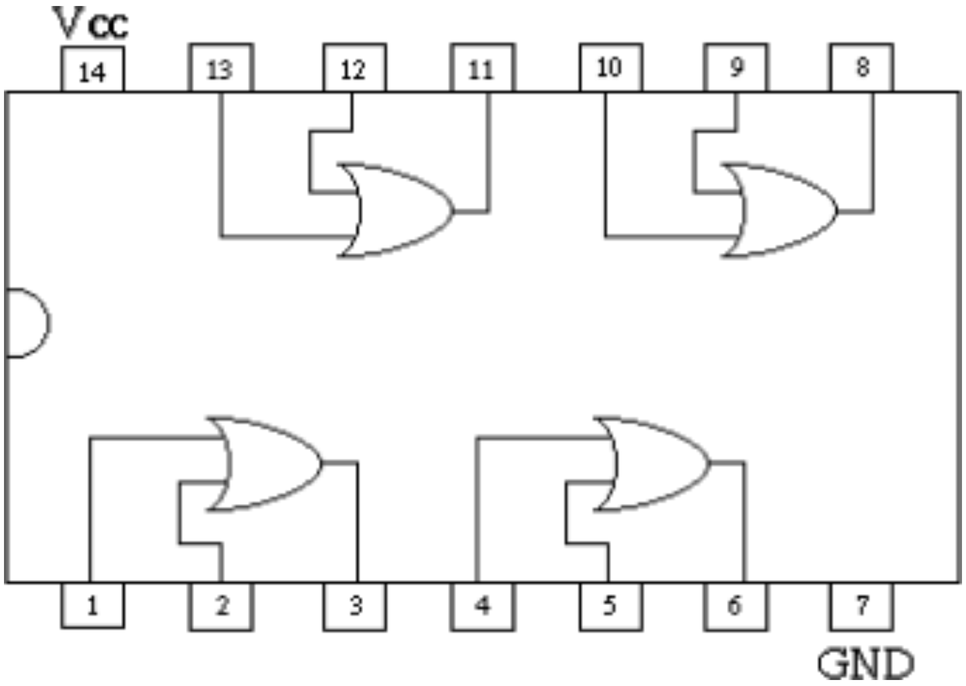
Step 3: Simplify the description

Step 4: Implement the system with the available components subject to the design objectives and constraints

Gates

- A gate is a network with one output
- Gate is the basic component for implementation
- For example, an OR gate is shown below:

Looking at a chip



Delays

Going from A, B to Y is not instantaneous

We will get into this more later, but this is something you should know

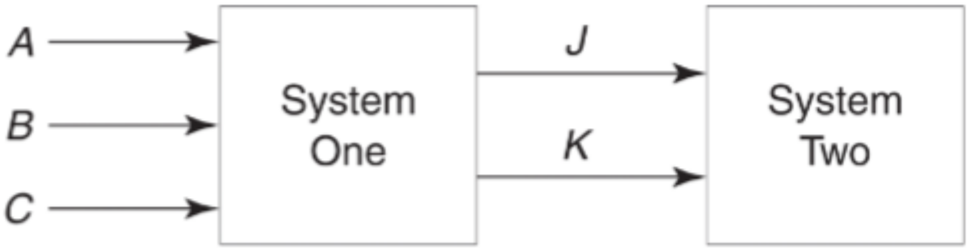
This is why you simplify systems

Don’t Care Conditions

- For some input combinations, it doesn’t matter what the output is
- Represented as X
- Examples of don’t cares:
  - Some input combinations that never occur
  - When one system is designed to drive a second system, some input combination of the first system will make the second system behave the same way

Example of Don’t Cares

- If for some combination of A, B, C, System Two behaves the same way no matter if J is 0 or 1, then J is a don’t care in this case



Truth Tables

Time to go back to the Continuing Examples!

Here is a new one

- A single light (that can be on or off) that can be controlled by any one of 3 switches. One switch is the master on/off switch. If it is off, the light is off. When the master is on, a change in the position of one of the other switches will cause the light to change state

One more new one...

- A system that has as its input the code for a decimal digit, and produces as its output the signals to drive a seven-segment display, such as those on most digital watches and numeric displays

Seven segment display