Digital Design and Computer Organization

Adder, Subtractor, Overflow - 1

Reetinder Sidhu

Department of Computer Science and Engineering

Digital Design and Computer Organization

Adder, Subtractor, Overflow - 1

Reetinder Sidhu

Department of Computer Science and Engineering

Adder, Subtractor, Overflow - 1

Course Outline

- Digital Design
  - Combinational logic design
    - **Adder, Subtractor, Overflow - 1**
  - Sequential logic design
- Computer Organization
  - Architecture (microprocessor instruction set)
  - Microarchitecure (microprocessor operation)

Concepts covered

- Unsigned Binary Numbers
- Signed (Two's Complement) Binary Numbers
- Binary Addition

Binary Number Representation (Unsigned)

Binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Three bit binary numbers (unsigned)

- Unsigned number representation:
  $m = \Sigma_{i=0}^{n-1} x_i \times 2^i$

Binary Addition Algorithm Example

$$
\begin{array}{rcccc}
 & 1 & 0 & 1 & 1 \\
+ & 0 & 0 & 1 & 1 \\
\hline
\end{array}
$$

- Logic circuits handle only a fixed number of bits
  - So result may not fit leading to Overflow
- Logic circuits cannot directly represent minus sign
  - So need signed number representation

Binary Number Representation (Unsigned)

Binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Three bit binary numbers (unsigned)

- Unsigned number representation:
  $m = \Sigma_{i=0}^{n-1} x_i \times 2^i$

Binary Addition Algorithm Example

$$\begin{array}{r} 1\ 0\ 1\ 1 \\ +\ 0\ 0\ 1\ 1 \\ \hline \end{array} \qquad \begin{array}{r} 1\ 1 \\ +\ 0\ 3 \\ \hline 1\ 4 \end{array}$$

- Logic circuits handle only a fixed number of bits
  - So result may not fit leading to Overflow
- Logic circuits cannot directly represent minus sign
  - So need signed number representation

Adder, Subtractor, Overflow - 1

Binary Number Representation (Unsigned)

Binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Three bit binary numbers (unsigned)

- Unsigned number representation:
  $m = \Sigma_{i=0}^{n-1} x_i \times 2^i$

Binary Addition Algorithm Example

$$\begin{array}{cccc} & & 1 & \\ 1 & 0 & 1 & 1 \\ +\ 0 & 0 & 1 & 1 \\ \hline & & & 0 \end{array} \qquad \begin{array}{cc} 1 & 1 \\ +\ 0 & 3 \\ \hline 1 & 4 \end{array}$$

- Logic circuits handle only a fixed number of bits
  - So result may not fit leading to Overflow
- Logic circuits cannot directly represent minus sign
  - So need signed number representation

Adder, Subtractor, Overflow - 1

Binary Number Representation (Unsigned)

Binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Three bit binary numbers (unsigned)

- Unsigned number representation:
  $m = \Sigma_{i=0}^{n-1} x_i \times 2^i$

Binary Addition Algorithm Example

$$
\begin{array}{rrrrr}
 & & 1 & 1 & \\
 & 1 & 0 & 1 & 1 \\
+ & 0 & 0 & 1 & 1 \\
\hline
 & & & 1 & 0
\end{array}
\qquad
\begin{array}{rrr}
 & 1 & 1 \\
+ & 0 & 3 \\
\hline
 & 1 & 4
\end{array}
$$

- Logic circuits handle only a fixed number of bits
  - So result may not fit leading to Overflow
- Logic circuits cannot directly represent minus sign
  - So need signed number representation

Adder, Subtractor, Overflow - 1

Binary Number Representation (Unsigned)

Binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Three bit binary numbers (unsigned)

- Unsigned number representation: $m = \Sigma_{i=0}^{n-1} x_i \times 2^i$

Binary Addition Algorithm Example

$$\begin{array}{rcccc} & & 1 & 1 & \\ & 1 & 0 & 1 & 1 \\ + & 0 & 0 & 1 & 1 \\ \hline & & 1 & 1 & 0 \end{array} \qquad \begin{array}{rcc} & 1 & 1 \\ + & 0 & 3 \\ \hline & 1 & 4 \end{array}$$

- Logic circuits handle only a fixed number of bits
  - So result may not fit leading to Overflow
- Logic circuits cannot directly represent minus sign
  - So need signed number representation

Adder, Subtractor, Overflow - 1

Binary Number Representation (Unsigned)

Binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Three bit binary numbers (unsigned)

- Unsigned number representation:
  $m = \Sigma_{i=0}^{n-1} x_i \times 2^i$

Binary Addition Algorithm Example

$$
\begin{array}{rrrrr}
 & & 1 & 1 & \\
 & 1 & 0 & 1 & 1 \\
+ & 0 & 0 & 1 & 1 \\
\hline
 & 1 & 1 & 1 & 0
\end{array}
\qquad
\begin{array}{rrr}
 & 1 & 1 \\
+ & 0 & 3 \\
\hline
 & 1 & 4
\end{array}
$$

- Logic circuits handle only a fixed number of bits
  - So result may not fit leading to Overflow
- Logic circuits cannot directly represent minus sign
  - So need signed number representation

Binary Number Representation (Two's Complement)

- Intuition behind two's complement representation:
  - Incrementing the 3-bit number 110 yields 111 and incrementing again yields 000
  - So decrementing 000 yields 111 and decrementing again yields 110
  - Thus 111 can represent -1, 110 can represent -2 and so on . . .

Binary	Decimal
000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

Three bit binary numbers (two's complement)

- Twos complement number representation: $m = -x_{n-1}2^{n-1} + \Sigma_{i=0}^{n-2} x_i \times 2^i$
- For 3-bit numbers, range shift from 0–7 to -4–3

Adder, Subtractor, Overflow - 1

Two's Complement of a Number

Two's Complement Procedure

Taking the two's complement of a number reverses the sign of the number:

1. Invert each bit of the number[a]
2. Add 1 to the number obtained

[a] Step 1 called one's complement.

Two's Complement Example

Consider the 4-bit number 0101 (5 in decimal):

1. Inverting all bits yields 1010
2. Adding 1 yields 1011, which represents -5

- Taking two's complement of 1011 reverses the sign again yielding 0101

Two's Complement Addition

PES UNIVERSITY ONLINE

Two's Complement Addition Algorithm Example

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

- Same addition algorithm
  - If inputs are interpreted as unsigned numbers, the output can be interpreted as their sum represented as an unsigned number
  - If inputs are interpreted as two's complement numbers, the output can be interpreted as their sum represented as a two's complement number
  - A key property of the two's complement representation

Adder, Subtractor, Overflow - 1

Two's Complement Addition

Two's Complement Addition Algorithm Example

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

$$\begin{array}{r} -5 \\ +\ \ 3 \\ \hline -2 \end{array}$$

- Same addition algorithm
  - If inputs are interpreted as unsigned numbers, the output can be interpreted as their sum represented as an unsigned number
  - If inputs are interpreted as two's complement numbers, the output can be interpreted as their sum represented as a two's complement number
  - A key property of the two's complement representation

Adder, Subtractor, Overflow - 1

Two's Complement Addition

Two's Complement Addition Algorithm Example

```
        1
    1 0 1 1
  + 0 0 1 1
  ---------
          0
```

```
   -5
 +  3
 ----
   -2
```

- Same addition algorithm
  - If inputs are interpreted as unsigned numbers, the output can be interpreted as their sum represented as an unsigned number
  - If inputs are interpreted as two's complement numbers, the output can be interpreted as their sum represented as a two's complement number
  - A key property of the two's complement representation

Two's Complement Addition

PES UNIVERSITY ONLINE

Two's Complement Addition Algorithm Example

		1	1	
	1	0	1	1
+	0	0	1	1
			1	0

	-5
+	3
	-2

- Same addition algorithm
  - If inputs are interpreted as unsigned numbers, the output can be interpreted as their sum represented as an unsigned number
  - If inputs are interpreted as two's complement numbers, the output can be interpreted as their sum represented as a two's complement number
  - A key property of the two's complement representation

Two's Complement Addition

Two's Complement Addition Algorithm Example

```
    1 1
  1 0 1 1
+ 0 0 1 1
---------
    1 1 0
```

```
   -5
+   3
-----
   -2
```

- Same addition algorithm
  - If inputs are interpreted as unsigned numbers, the output can be interpreted as their sum represented as an unsigned number
  - If inputs are interpreted as two's complement numbers, the output can be interpreted as their sum represented as a two's complement number
  - A key property of the two's complement representation

Adder, Subtractor, Overflow - 1

Two's Complement Addition

Two's Complement Addition Algorithm Example

$$
\begin{array}{r}
1\ 1 \\
1\ 0\ 1\ 1 \\
+\ 0\ 0\ 1\ 1 \\
\hline
1\ 1\ 1\ 0
\end{array}
\qquad
\begin{array}{r}
-5 \\
+\ \ 3 \\
\hline
-2
\end{array}
$$

- Same addition algorithm
  - If inputs are interpreted as unsigned numbers, the output can be interpreted as their sum represented as an unsigned number
  - If inputs are interpreted as two's complement numbers, the output can be interpreted as their sum represented as a two's complement number
  - A key property of the two's complement representation

Adder, Subtractor, Overflow - 1

Two's Complement Addition

Two's Complement Addition Algorithm Example

		1	1	
	1	0	1	1
+	0	0	1	1
	1	1	1	0

	-5			1	1
+	3		+	0	3
	-2			1	4

- Same addition algorithm
  - If inputs are interpreted as unsigned numbers, the output can be interpreted as their sum represented as an unsigned number
  - If inputs are interpreted as two's complement numbers, the output can be interpreted as their sum represented as a two's complement number
  - A key property of the two's complement representation

Basic Definitions

- Sign extension
- Hexadecimal number representation
- Most Sgnificant Bit (msb)
- Least Sgnificant Bit (lsb)
- Most Sgnificant Byte (MSB)
- Least Sgnificant Byte (LSB)
- Bit, nibble, byte
- Kilobit (Kb), Megabit (Mb) and Gigabit (Gb)
- Kilobyte (KB), Megabyte (MB) and Gigabyte (GB)

Think About It

Another Binary Addition

- Add the binary numbers 0111 and 1011. Does the output make sense if inputs are interpreted as:
  - Unsigned binary numbers
  - Signed, two's complement numbers
- Hint: may need to remove overflow bit

Two's Complement Exceptions

- Consider 4-bit binary numbers
- There are two numbers whose two's complement does not reverse their signs
- What are those numbers?
- Does above apply to 3-bit or 5-bit numbers?