A Review of Various Adders for Fast ALU

Sobina Gujral¹, Robina Gujral²

¹Assistant Professor Department of Electronics and Communication, Chandigarh University
²Assistant Professor Department of Electronics and Communication, MVJ College of Engineering

Abstract:

Adders are the key element of the arithmetic unit, especially fast parallel adder. This paper reviews a number of adders that have been presented over the last few decades. In this paper a comparison is made between conventional adders and nonconventional adders. There are several conventional adders but mainly here we are describing RCA and CLA adders and comparing it to other signed digit based adders such as Hybrid Signed Digit (HSD), Quaternary Signed Digit (QSD) and Redundant Binary Signed Digit (RBSD) to show that how these adders are better than CLA and simultaneously proving that RBSD is better than all.

QSD and RBSD are fast parallel adders as the time consumption in the addition process is very less as compared to other conventional and non-conventional adders.

Keywords: RCA, CLA, RBSD, Carry free addition, QSD, HSD, FastComputing.

1. Introduction

The carry propagation time is a limiting factor on speed with which two numbers are added in parallel. Although a parallel adder, or any combinational circuit will always have some value at its output terminal, the output will not be correct unless the signals are given enough time to propagate through the gates connected from the input to the outputs. Since all other arithmetic operations are implemented by successive additions, the time consumed during the addition process is very critical.

An obvious solution for reducing the carry propagation delay time is to employ faster gates with reduced delays and different technologies for the same, but physical circuit have a limit to their capability.

In this paper a comparative study is done among different adders such as Ripple carry adder (RCA), Carry look-ahead adder (CLA), Redundant Binary Signed Digit (RBSD), Hybrid Signed Digit (HSD) and Quaternary Signed Digit (QSD).

In RCA carry bit is calculated alongside the sum bit and each bit must wait until the previous carry is being calculated.

In CLA speed is improved by reducing the amount of time required to determine carry bits.

RBSD has a carry free addition and fast speed of addition.

The HSD representations are capable of bounding the maximum length of carry propagation chains during addition of any desired value between 1 and the entire word length.

Addition process is faster in QSD but the hardware circuitry will be complex due to more number of bits are required to encode the QSD number which ranges from -3 to +3.

In this paper we have done different sections as section 2 describes RCA, section 3 describes CLA, section 4 describes Signed Digit, section 4.1 describes RBSD, Section 4.2 describes HSD, section 4.3 describes QSD number system and in section 5 & 6 comparison and conclusion are presented.
2. Ripple Carry Adder

Ripple carry adder is a adder for which the carry bit is calculated alongside the sum bit, and each bit must wait until the previous carry has been calculated to begin calculating its own result and carry bits.

Example: Suppose two numbers \((6)_{10} = 0110\) and \((8)_{10} = 1000\) is being added.

Previous Carry Bit

<table>
<thead>
<tr>
<th>Carry</th>
<th>Operand A</th>
<th>Operand B</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 1 0</td>
<td>1 0 0 0</td>
<td>1 0 1 0</td>
</tr>
</tbody>
</table>

From the above example it is clear that the carry bits are propagated from LSB to MSB, results in more consumption of time due to carry propagation chain. If the number is large than the carry propagation chain will also be increased hence more time consumption.

3. Carry Look-ahead Adder

There are several techniques used for reducing the carry propagation time in a parallel adder. The most widely used technique employs the principle of look-ahead carry. A carry look-ahead adder improves speed by reducing the amount of time required to determine carry bits [4]

Considering the full adder in which \(A_i\) and \(B_i\) are inputs we can write

\[
P_i = A_i \oplus B_i \tag{1}
\]

\[
G_i = A_i \cdot B_i \tag{2}
\]

Where in equation (1) and (2) \(i = 0, 1, 2, 3\).

Then the output sum and carry can be expressed as:

\[
S_i = P_i \cdot C_i \tag{3}
\]

\[
C_{i+1} = G_i + P_i \cdot C_i \tag{4}
\]

Where in equation (3) \(P_i\) is carry propagate and in equation (4) \(G_i\) is called a Carry generator which produces an output carry when both \(A_i\) and \(B_i\) are one, regardless of input carry.

In Carry Lookahead Adder the carry propagation delay is reduced, means the processing time is reduced. But due to number of gates and levels the complexity is increased.

The 4-bit carry look-ahead (CLA) adder consists of 3 levels of logic:

**First level:** Generates all the P & G signals. Four sets of P & G logic (each consists of an XOR gate and an AND gate). Output signals of this level (P’s & G’s) will be valid after 1\(\tau\).
Second level: The Carry Look-Ahead (CLA) logic block which consists of four 2-level implementation logic circuits. It generates the carry signals ($C_1$, $C_2$, $C_3$, and $C_4$) as defined by the above expressions. Output signals of this level ($C_1$, $C_2$, $C_3$, and $C_4$) will be valid after $3\tau$.

Third level: Four XOR gates which generate the sum signals ($S_i$) ($S_i = P_i \oplus C_i$). Output signals of this level ($S_0$, $S_1$, $S_2$, and $S_3$) will be valid after $4\tau$.

CLA adder can be preferred for 12 bit numbers as it has high speed as well as tolerable complexity.

4. Fast Addition Techniques

4.1 Signed Digit Number Representation

Signed –digit representation limit carry propagation to one position to the left during the operations of addition and subtraction in digital computers. Carry-propagation chains are eliminated by use of redundant representation for operands. In this method, each digit of a positional constant radix number representation with an integer radix $r$ is allowed to assume $q$ values [1]

$$r+2 \leq q \leq 2r-1$$  \hspace{1cm} (5)

Both positive and negative digit values are allowed. The purpose of signed digit representation is to allow addition and subtraction of two numbers in which no serial signal propagation is required along the adder; i.e., the time duration of the operation is independent of the length of the operands and is equal to the time required for addition or subtraction of two digits.

A signed digit number is represented by $n+m+1$ digits $z_i$ ($i=-n, \ldots, -1, 0, 1, \ldots, m$) and has the algebraic value

$$\sum_{i=-n}^{m} z_i \cdot r^{-i}$$  \hspace{1cm} (6)

Where the values of $r$ and $z_i$ in equation (6) are such that the following requirements are satisfied:

1) Radix $r$ is a positive integer.
2) The algebraic value $Z=0$ has a unique representation.
3) Totally parallel addition and subtraction is possible for all digits in corresponding positions of two representations.

Example: Signed Digit number representation

If the range is -2 to 2 the number $(57)_{10}$ can be represented as

$$21_{I\bar{I}} = 3^3 \cdot 2 + 3^2 \cdot 1 + 3^1 \cdot \bar{1} + 3^0 \cdot \bar{I} = 57$$

Similarly if the range is -1 to 1 the same number can be represented as

$$0100_{I\bar{I}} = 57$$
$$1000_{I\bar{I}} = 57$$
$$0111_{I\bar{I}} = 57$$

If radix 4 is considered then

Sobina Gujral¹, Robina Gujral²
\[
\begin{align*}
1 \equiv 1 &= 4^1 \cdot 1 + 4^0 \cdot 1 \\
4^2 \cdot 2 + 4^0 \cdot 1 &= 57 \\
2 \equiv 1 &= 4^3 \cdot 2 + 4^2 \cdot 3 + 4^1 \cdot 3 + 4^0 \cdot 1 = 57 \\
10 \equiv 1 &= 4^3 \cdot 1 + 4^2 \cdot 0 + 4^1 \cdot 3 + 4^0 \cdot 1 = 57
\end{align*}
\]

Redundancy:
From the above examples it is clear that by using the unconventional number representations i.e. signed digit representation a single number can be written in more than one way. This characteristics is called as redundancy and the different numbers are called redundant numbers.

4.2 Adder Design using RBSD
In this addition technique the numbers to be added must be in RBSD i.e. represented using the digit set \{1,0,\overline{1}\} unlike binary number system, which is represented, with digit set \{0,1\}.

The decimal value of RBSD number can be calculated by the following relation:
\[
D = \sum_{i=0}^{n-1} x_i \cdot 2^i \quad (7)
\]

Example:
Let the number 011000 represents 24 in binary number system. The same number in RBSD can be represented as follow
\[
(010\overline{1}000)_{\text{rbsd}} = \overline{0} \cdot 2^4 + \overline{1} \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = (24)_{10}
\]
Kal and Rajashekhar, 1990 has given following Boolean expression for RBSD addition. In addition process there was no carry propagation chain. A number of any bit length can be added in constant time period as it is parallel addition[5].

Addition of digits \(z_i\) and \(y_i\) is totally parallel if the following two conditions are satisfied.

1)The sum digit \(s_i\)(ith digit of the sum \(s=z+y\)) is a function only of augend digit \(z_i\), addend digit \(y_i\) and the transfer digit \(t_i\) from \(t_{i+1}\)th position on the right \(s_i=f(z_i, y_i, t_i)\).

2)The transfer digit \(t_{i-1}\) to the \((i-1)\)th position on the left is a function only of augend digit \(z_i\) and the addend digit \(y_i\): \(t_{i-1} = f(z_i, y_i)\).

The digit addition rules may be modified to allow the propagations to the left. Two transfer addition process is executed in following three successive steps

1) \(z_i + y_i = r_{i-1} + w_i\) \quad (8)
2) \(w_i + t_i = r_{i-1} + w_i\) \quad (9)
3) \(s_i = w_i + t_i\) \quad (10)

The potential advantage of two transfer addition is decreased redundancy requirement.

A number in signed representation requires more allowed values and more storage capacity per digit than the same number in conventional representation and complexity is also increased in its addition.

![Redundant Binary Signed Digit Adder](image-url)

Fig 4. Redundant Binary Signed Digit Adder
In simple signed digit addition a borrow digit is used to ensure that all intermediate sum digits are members of the set \{0, 1\} and all intermediate carry digits are members of the set \{0, 1\}.

Digital Design:

In digital logic -1 is not considered hence we refer some representations in binary codes as \{1,0,\overline{1}\} are represented as (0,1), (0,0) and (1,0) respectively.

For digital designing the equations from (11) –(15) are used [6]

\[ d_i = m_i \oplus \overline{x_i^+ x_i^-} \oplus \overline{x_i^+ y_i^-} \]  

(11)

\[ m_{i+1} = x_i^+ y_i^+ \]  

(12)

\[ b_{i+1} = m_i + x_i^+ y_i^+ + \overline{x_i^+ y_i^-} + \overline{y_i^+} \]  

(13)

\[ s_i^* = \overline{d_i} \]  

(14)

\[ s_{i+1} = d_i b_i \]  

(15)

In above equations \( m_i, b_i, \) and \( d_i \) are binary variables with digit set \{0,1\} and \( x_i^+, y_i^+ \) are RBSD input numbers and \( s_i^* \) output sum in RBSD with digit set \{-1,0,1\} represented by binary encoded form i.e. (0,1), (0,0), (1,0) respectively.

The binary variable representation corresponding to \( x_i^*, y_i^* \) and \( s_i^* \) are \( \overline{x_i}, \overline{y_i}, \overline{y_i} \) and \( \overline{s_i} \) respectively.

Fig 5. Logic Diagram of Basic RBSD Adder

It has a carry free and fast speed of addition. Comparing it to the other adders, it is clear that it has high speed as well as due to parallel execution the time required for the processing is not affected by the changes in the number of bits hence it is better than other adders.

4.3 Adder Design using Hybrid Number System

In binary signed digit number system each digit can assume any one of three values \{-1,0,1\}. As a result, redundancy is introduced in the number system i.e., number can be represented in more than one way.

Example:

I can be represented by 01 or 1\overline{1}, where \( \overline{1} = -1 \).

This redundancy can be exploited to limit the length of carry propagation chains to only one digit position, making it possible to add two numbers in fixed time, irrespective of the word length [7]. In this representation instead of insisting that every digit be signed digit, we let

Sobina Gujral¹, Robina Gujral²
some of the digits to be signed and leave the others unsigned.

For example, every alternate or every third or fourth digit can be signed; all the remaining ones are unsigned. This representation can limit the maximum length of carry propagation chains to any desired value. Maximum length of a carry propagation chain equals \((d+1)\), where \(d\) is the longest distance between neighbor signed digits.

For instance, let \(x_i\) and \(y_i\) be radix -2 signed digits to be added at \(i\)th digit position and \(c_{i-1}\) be the carry into the \(i\)th digit position. Each of these variables can assume any of the three values \{-1, 0, 1\}. Hence \(-3 \leq x_i + y_i + c_{i-1} \leq +3\). This sum can be represented in terms of a signed output \(z_i\) and a signed carry \(c_i\) as follows:

\[
X_i + y_i + c_{i-1} = 2c_i + z_i \quad (16)
\]

Where \(c_i, z_i \in \{-1, 0, 1\}\). In practice, the signed digit output \(z_i\) is not produced directly. Instead the carry \(c_i\) and an intermediate sum \(s_i\) are produced in first step, and the summation \(z_i = s_i + c_{i-1}\) is carried out in the second. The operations in an unsigned digit position are as follows.

Let \(a_{i-1}\) and \(b_{i-1}\) be the bits to be added at the \((i-1)\)th digit position; \(a_{i-1}, b_{i-1} \in \{0, 1\}\). The carry into the \((i-1)\)th position is signed and can be \(-1, 0, \text{or} 1\). The output digit \(e_{i-1}\) is restricted to be unsigned, i.e., \(e_{i-1} \in \{0, 1\}\). Hence, the carry out of the \((i-1)\)th position must be allowed to assume the value \(-1\) as well. In particular, if \(a_{i-1} = b_{i-1} = 0 \& c_{i-2} = -1\) then \(c_{i-1} = -1\) and \(e_{i-1} = 1\);

else

\[a_{i-1} + b_{i-1} + c_{i-2} = 2c_{i-1} + e_{i-1}\]

where \(c_{i-1}, e_{i-2} \geq 0\)

Table 1. Rules for selecting the carry \(c_i\) and intermediate sum \(s_i\) based on \(X_i, Y_i, A_{i-1}\) and \(B_{i-1}\), where \(X_i\) and \(Y_i\) are signed digits and \(A_{i-1}, B_{i-1}\) are unsigned digits.

<table>
<thead>
<tr>
<th>(x_i, y_i)</th>
<th>(a_{i-1}, b_{i-1})</th>
<th>(c_i)</th>
<th>(s_i)</th>
<th>(c_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>11</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>00</td>
<td>a_{i-1} = b_{i-1} = 0</td>
<td>{-1, 0}</td>
<td>+</td>
</tr>
<tr>
<td>-1</td>
<td>01</td>
<td>At least one of (a_{i-1}, b_{i-1}) is 1</td>
<td>(+1, 0)</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>+1</td>
<td>01</td>
<td>a_{i-1} = b_{i-1} = 0</td>
<td>{-1, 0}</td>
<td>+</td>
</tr>
<tr>
<td>+1</td>
<td>10</td>
<td>At least one of (a_{i-1}, b_{i-1}) is 1</td>
<td>(+1, 0)</td>
<td>-1</td>
</tr>
<tr>
<td>+2</td>
<td>11</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

From the table, it is clear that the carry \(c_i\) out of the signed digit is independent of the carry into the previous unsigned \((i - 1)\)th digit position, viz., \(c_{i-2}\).

Hence, the carries out of, and the intermediate sums at all the signed digits positions can be calculated in parallel in the first step.

Furthermore, from the table it is seen that whenever the carry \(c_{i-1}\) to be generated at the \((i - 1)\)th position is expected to be nonnegative, \(s_i\) is selected to be non-positive and vice versa. In other words, \(s_i\) and \(c_{i-1}\) are guaranteed to have opposite polarity.

Sobina Gujral¹, Robina Gujral²
Consequently, the addition \( z_i = s_i + c_{i-1} \) can never generate a new carry. Thus, the carry propagation stops at the signed digit(s). The most important point is that it is possible to predict when \( c_{i-1} \) will be non-positive and when it will be non-negative, just by looking at the operand digits \( a_{i-1} \) and \( b_{i-1} \).

It is not necessary to wait until the actual value of \( c_{i-1} \) becomes available; which makes it possible to break the carry propagation chain. All the other digits can be unsigned.

**Example:**

If the word length is 32 digits, then, the 32nd (ie., the most significant) digit is a signed digit. The remaining digits are at the designer’s disposal. If regularity is not necessary, one can make the 1st, 2nd, 4th, 8th and 16th (and 32nd) digits signed and let all the remaining digits be unsigned digits (bits).

The addition time for such a representation is determined by the longest possible carry-propagation chain between consecutive signed digit positions (16 digit positions; from the 16th to the 32nd digit in this example).

The hybrid number representations are capable of bounding the maximum length of carry propagation chains during addition of any desired value between 1 and the entire word length. The HSD representation is not restricted to a particular HSD format.

The representation can be modified without any additional delay. But when two HSD numbers are added, an extra digit position is required to store the result, this leads to non-uniformity.

### 4.4 Adder Design using QSD

Any signed digit quaternary integer may be represented by [3]

\[
D = \sum_{i=1}^{n} 4^{i-1} z_i
\]

(17)

Where the variable \( z_i \) can take any value from the set \{ -3, -2, -1, 0, 1, 2, 3 \}.

The design using QSD demonstrates that the circuit can be implemented with fewer gates as compared to binary signed digit adder (BSD).

Example: Suppose the number in QSD is represented as \( 2102 \).

\[
4^3 \times 2 + 4^2 \times 1 + 4^1 \times 0 + 4^0 \times (-2) = 142
\]

Each digit is represented by a 3 bit 2’s complement binary notation. The addition between two binary QSD number \( a_n, a_{n-1}, \ldots, a_1 \) and \( b_n, b_{n-1}, \ldots, b_1 \) can be represented by the following sequence

\[
S_i = q_i \times \overline{q_{i-1}} + s_i \times (ti + 4xi) - q_{i-1} - (-q_{i-1}x_{i-1})
\]

(18)

Where

\[
to = 0
\]

\[
t_i = s_i + b_i
\]

\[
x_i = 0 \quad \text{if} \ -4 < t_i < 4
\]

\[
= \overline{1} \quad \text{if} \ t_i \geq 4
\]

\[
= 1 \quad \text{if} \ t_i \leq -4
\]

\[
q_i = 1 \quad \text{if} \ t_i = \overline{3}
\]

\[
= \overline{1} \quad \text{if} \ t_i = 3
\]

\[
= 0 \quad \text{otherwise}
\]

\[
\overline{q}_i = 0 \quad \text{if} \ q_i \neq 0
\]

\[
= 1 \quad \text{if} \ q_i = 0
\]

**Example:**
Suppose the two numbers represented in QSD are 210 and 321.

There sum will be

\[ 210\overline{5} + 321\overline{5} = 531\overline{5} \text{(QSD Representation)} \]

\[
\begin{align*}
2 & \phantom{10} 1 & 0 & \overline{5} \\
+ & 3 & 2 & 1 & \overline{5} \\
\hline
& 5 & 3 & 1 & \overline{5}
\end{align*}
\]

In above example it is clear that some representations of the digits must be taken for further solutions.

The new representation can be taken from theTable 2:-

<table>
<thead>
<tr>
<th>Digits</th>
<th>QSD code</th>
</tr>
</thead>
<tbody>
<tr>
<td>-6</td>
<td>\overline{12}</td>
</tr>
<tr>
<td>-5</td>
<td>\overline{11}</td>
</tr>
<tr>
<td>-4</td>
<td>\overline{10}</td>
</tr>
<tr>
<td>-3</td>
<td>\overline{10}</td>
</tr>
<tr>
<td>-2</td>
<td>0 \overline{2}</td>
</tr>
<tr>
<td>-1</td>
<td>0 \overline{1}</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
</tr>
<tr>
<td>3</td>
<td>1\overline{1}</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
</tr>
</tbody>
</table>

**Table 2. QSD Code Representation**

Hence we can further proceed as:

\[
\begin{align*}
5 & \phantom{10} 3 & 1 & \overline{5} \\
1 & \phantom{10} 1 & 0 & \overline{1} \\
\hline
& 1 & \overline{1} & 1 & \overline{1}
\end{align*}
\]

In decimal form

\[
210\overline{5} = 4^3 \times 2 + 4^2 \times 1 + 4^1 \times 0 + 4^0 \times (-2) = (142)_{10}
\]

\[
321\overline{5} = 4^3 \times 3 + 4^2 \times 2 + 4^1 \times 1 + 4^0 \times 3 = (225)_{10}
\]

Result:- 142+225=367

QSD addition save 25% storage compared to BSD but it has a bit higher gate delay.

The concept of addition using QSD is shown in figure 5. It is clear from the diagram that the sum digit of QSD addition can be achieved only in one stage as compared to RBSD addition which takes two steps.

**Fig 6. Block Diagram Of QSD Addition**

Therefore the addition process is faster in QSD but the hardware circuitry will be complex due to more number of bits are required to encode the QSD number which ranges from -3 to +3.

**5. Discussion:**

Different types adders are described in this paper. In Carry Lookahead Adder the carry propagation delay is reduced, but due to number of gates and levels the complexity is increased. Signed digit representation offer the means for a
completely parallel execution of arithmetical operations in digital computer, but a number in signed representation requires more allowed values and more storage capacity per digit than the same number in conventional representation and complexity is also increased in its addition. QSD addition save 25% storage compared to BSD but it has a bit higher gate delay. QSD addition can be achieved only in one stage as compared to RBSD addition which takes two steps. Therefore the addition process is faster in QSD but the hardware circuitry will be complex due to more number of bits are required to encode the QSD number which ranges from -3 to +3. HSD representation can limit the maximum length of carry propagation chains to any desired, but when two HSD numbers are added an extra digit position is required to store the result, this leads to nonuniformity. At last RBSD has a carry-free addition and fast speed of processing.

6. Conclusion
Through the above discussions it is clear that each and every adder has some of the merits and demerits. So the best choice depends on the requirement of specific application and technology.

References