Acknowledgement

All praise is to ALLAH, All success is from ALLAH. I would like to thank ALLAH for bestowing upon me the chance, strength and ability to complete this work.

I would like to thank my parents for their support, help and prayers. Without their prayers, I could not achieve any progress in my life. Thanks a lot my father and my mother for all you did for me since my birth.

I would like to express my appreciation to my advisors Dr. Ahmed Hussien and Dr. Hossam Fahmy for their guidance during my research.

Special thanks to Dr. Hossam Fahmy for his endless support and advice. I think the best words to describe the relation between Dr. Hossam and me is as a person and his younger brother. Thanks a lot Dr. Hossam.

Special thanks to the deceased Dr. Ali Ezzat who taught me a lot. Dr. Ali supported and helped me in all directions.

Special thanks to Ahmed Farid my manager for his endless support and understanding. I would like also to thank my Mentor Graphics family. Special thanks to the Consulting team and my colleagues in my room who support me during the difficult days.

Finally, I would like to express my deepest appreciation to my family: my father, my mother and my sister for their continuous support and understanding.

Of course I say first and foremost “Thanks to ALLAH”.

Abstract

Two-operand addition is a primitive operation included in practically all arithmetic algorithms. The efficiency of arithmetic circuits depends mainly on the adder architecture and implementation.

In this thesis, we analyze the use of the signed digit (SD) number system and its advantages in eliminating the carry propagation chain and hence achieving simultaneously high speed and low EDP arithmetic circuit. This study is essential for future CAD synthesis tools where the tool is responsible for the choice of the best suited circuit depending on the application.

The hybrid signed digit (HSD) system is presented as an intermediate solution between the SD system and the conventional system. A HSD adder architecture is analyzed, a new HSD adder/subtractor and a modified version of this circuit are proposed. Furthermore a third HSD adder/subtractor from the literature is analyzed. A complete arithmetic unit based on the new circuit is also proposed and analyzed including the conversion to and from the conventional system.

The distance between neighboring signed digits (d) is the main parameter in our characterization of the previous architectures. This distance indicates the level of redundancy in the system. We analyze the effect of changing d on the delay, area, ADP, energy, EDP and EDAP of each architecture.

On one hand, as the number of signed digits increases in the mentioned architectures, it is possible to decrease the delay, ADP and EDP. The decrease percentage ranges from (66% to 95%) for the delay, (65% to 94%) for the ADP and (65% to 95%) for the EDP depending on the architecture.

On the other hand, as the number of signed digits decreases in the mentioned architectures, it is possible to decrease the area. The decrease percentage ranges from (3% to 49%) depending on the architecture.

The energy consumption depends mainly on the architecture. As the number of signed digits increases in the architecture, it is possible to decrease the EDAP in most cases. However, in some cases the area curve is dominant over the EDP curve.

We recommend using a higher redundancy for applications targeting a shorter delay, a smaller ADP, and a smaller EDP. For low area applications, conventional architectures are the best choice.

Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

ADP	Area - Delay Product
ASIC	Application Specific Integrated Circuit
CAD	Computer Aided Design
CLA	Carry Look-Ahead
CMOS	Complementary Metal Oxide Semiconductor
d	Distance between Signed Digits in HSD Architecture
DC	Direct Current
EDP	Energy - Delay Product
EDAP	Energy - Delay - Area Product
FA	Full Adder
FPGA	Field Programmable Gate Array
HA	Half Adder
HSD	Hybrid Signed Digit
IC	Integrated Circuits
log	Logarithmic Function
MOS	Metal Oxide Semiconductor
n	Number of Operand Bits
RTL	Register Transfer Language
SD	Signed Digit
TSMC	Taiwan Semiconductor Manufacturing Company

$\oplus$	Exclusive-or Logic Gate
+	Or Logic Gate
.	And Logic Gate
~	Not Logic Gate

Chapter 1

1- Introduction

1.1 The need of future circuits

Addition is the most primitive operation among all arithmetic operations. It is the basic building block in all arithmetic circuits such as multipliers, dividers, encryption algorithms and digital signal processing algorithms. So, the efficiency of arithmetic circuits depends mainly on the adder architecture and implementation.

With the rapid increase in battery operated portable applications such as mobile phones and laptops, there is a higher need to decrease the energy consumption of the circuit to increase the battery lifetime. The stringent requirements of future high speed low energy arithmetic challenge designers and CAD tools.

In 1961, signed digit (SD) number representations were proposed for fast arithmetic [22]. In a conventional decimal system each digit takes values from 0 to 9. In a purely signed digit decimal system, each digit is allowed to have negative values as well. For example, the digits may take any value from -9 to +9. Under these assumptions a number such as 3 may be represented with a 0 in the tens digits and a 3 in the unit digit as (0)(3) or with a 1 in the tens digit and a -7 in the units digit as (1)(-7) since (1)x10+(-7)x1=3. The possibility to represent the same number with multiple representations provides a redundancy in the system that permits the designer of an addition circuit to choose the representation that achieves the better speed by eliminating the carry propagation between the digits. In this thesis, we briefly discuss the theory behind the (SD) number system and the necessary conditions to achieve high speed arithmetic via carry free addition. We use the hybrid signed digit (HSD) number system [13] as a mixture of the pure SD number system and the conventional number system. In the HSD number system some of the digits are chosen as signed while the remaining digits are kept as unsigned digits.

We characterize the HSD adder circuit by changing its distance parameter d (distance between neighboring signed digits), and analyze its effect on the adder delay, area, ADP, energy, EDP and EDAP.

Also, we propose a HSD adder/subtractor and a modified version of this circuit. Furthermore, we analyze a third HSD adder/subtractor architecture. All three architectures are characterized by changing the distance parameter (d) and analyzing its effect on the adder/subtractor delay, area, ADP, energy, EDP and EDAP.

Finally, we propose an arithmetic unit: it consists of HSD adder/subtractor and a register file to store the results of the HSD adder/subtractor. Data stored in the register file support the HSD format. Also, we analyze the effect of the conversion from the conventional system to the HSD system and the reverse.

A characterization of the arithmetic unit is done by changing its distance parameter d and then analyzing its effect on the arithmetic unit delay, area, ADP, energy, EDP and EDAP.

Comparison with carry look-ahead (CLA) architectures is done for all previous architectures.

1.2 Thesis outline

The following chapters provide detailed information about SD number system, HSD architectures and their characterization with respect to the distance d parameter.

Chapter 2 discusses the basic adder architectures like ripple-carry adder, carry look-ahead adder and carry-select adder. Also it introduces the basic concepts and theory of the SD number system and the necessary conditions to achieve carry free addition.

Chapter 3 presents the HSD number system; also it analyzes the characterization of the HSD adder with respect to its area, delay and ADP.

Chapter 4 presents the newly proposed HSD adder/subtractor and a modified version of this circuit, Furthermore; it presents the analysis of a third HSD adder/subtractor. The three architectures are characterized with respect to their area, delay and ADP.

Chapter 5 presents the proposal of an arithmetic unit and its characterization with respect to its area, delay and ADP. Finally, it introduces the effect of the conversion from and to the conventional number system.

Chapter 6 presents the characterization of the HSD adder, the HSD adder/subtractor architectures and the arithmetic unit with respect to their energy, EDP and EDAP.

Chapter 7 sums up the conclusions and offers suggestions for future work.

Chapter 2

2- Basic adder architectures

2.1 Introduction

Two-operand addition is a primitive operation included in practically all arithmetic algorithms. A two-operand adder is used not only when performing additions and subtractions, but also employed when executing more complex operations like multiplication and division. As a consequence, the efficiency of an arithmetic circuit strongly depends on the way the adders are implemented.

A key point in two-operand adder implementation is the way the carries are computed. Normal ripple-carry adder delay is proportional to n (number of operand bits), while in architectures like the carry look-ahead adder the delay is proportional to log n [6].

In this chapter, the basic adder architectures like ripple-carry adder, carry look-ahead adder and carry-select adder are discussed. We finalize this chapter by introducing the signed digit number system and its advantages in eliminating the carry propagation chain and hence achieving high speed adder circuit. However, this increase in the speed does not come without a cost; the area of this circuit is larger than the area of the traditional architectures.

2.2 Half adder

A half adder (HA) is a digital logic circuit capable of adding two bits and producing the output sum and carry. Let us assume the two inputs bits are x_i and y_i, while the two outputs are the sum s_i and the output carry c_{out}. The truth table of the HA circuit is summarized in the following table.

Table 2.1: Truth table of HA circuit.

x_i	y_i	c_{out}	s_i
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The HA is a combinational digital circuit implementing the binary addition of two bits through the following Boolean equations [16]:

$$s_i = x_i \oplus y_i \tag{2.1}$$

$$c_{out} = x_i \cdot y_i \tag{2.2}$$

Where (x_i . y_i) is the logical AND operation, while ($x_i \oplus y_i$) is the logical Exclusive–or operation.

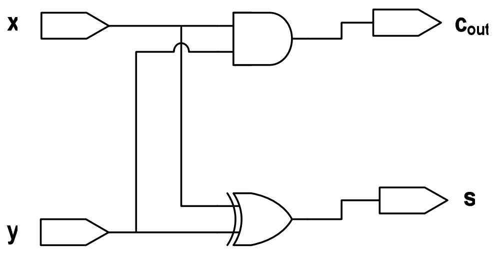

Figure 2.1: HA logic diagram.

2.3 Full adder

A full adder (FA) is a digital logic circuit capable of adding three bits and producing the output sum and carry. The three inputs are x_i, y_i and the input carry c_i , while the two outputs are the sum s_i and the output carry c_{out}. The truth table of the FA circuit is summarized in the following table.

Table 2.2: Truth table of FA circuit.

c_i	x_i	y_i	c_{out}	s_i
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

The FA is a combinational digital circuit implementing the binary addition of three bits through the following Boolean equations [20]:

$$s_i = x_i \oplus y_i \oplus c_i \tag{2.3}$$

$$c_{out} = x_i \,.\, y_i + c_i \,.\, x_i + c_i \,.\, y_i \tag{2.4}$$

Where (x_i . y_i) is the logical AND operation, (x_i $\oplus$ y_i) is the logical Exclusive–or operation and ($x_i + y_i$) is the logical OR operation.

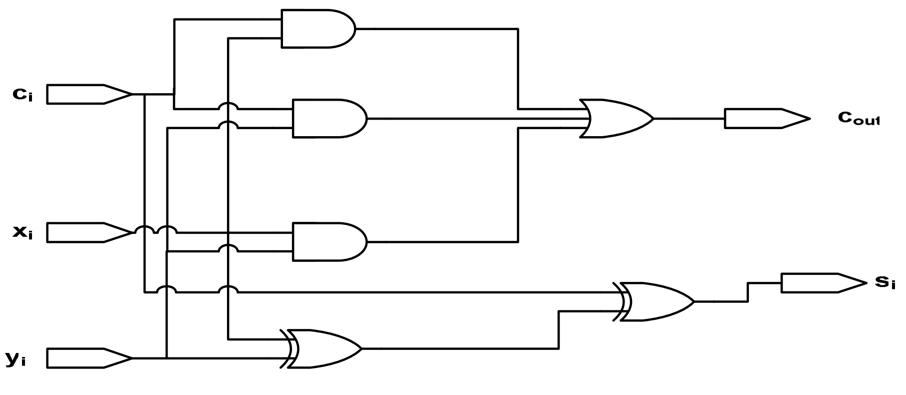

Figure 2.2: FA logic diagram.

2.4 Ripple-carry adder

Ripple-carry adder is the most straightforward implementation of a parallel adder, it consists of n cascaded FAs for adding two operands X and Y each n bits length.

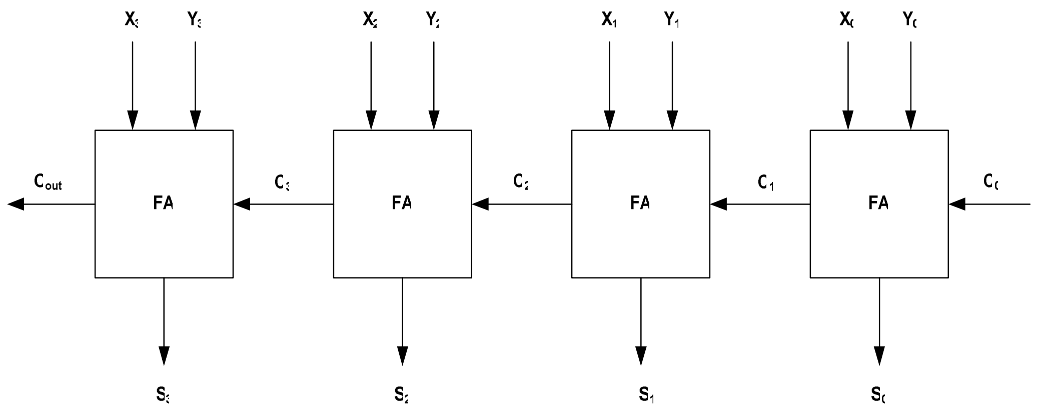

Figure 2.3: Four bit ripple-carry adder.

A ripple-carry adder for n=4 is depicted in figure 2.3. The two operands X and Y are available at the same time; however the carry signals between the FA cells have to propagate from the least significant cell to the most significant cell.

In other words, we must wait until the carry ripples through all FA cells to ensure that the final sum is correct and can be used in further calculations; hence the name of ripple-carry adder.

We discuss two examples to illustrate the delay in the ripple-carry adder, we assume that the delay through the FA cell is T_FA; in this analysis we assume that the delay of the sum and the carry is equal.

Table 2.3: First ripple-carry adder addition example.

Time	$x_3x_2x_1x_0$	$y_3y_2y_1y_0$	$c_{out}c_3c_2c_1c_0$	$s_3s_2s_1s_0$
T=0	1100	0111	xxxxx	xxxx
T=T_FA	1100	0111	01000	1011
T=2T_FA	1100	0111	11000	0011

In table 2.3, we have two operands X=1100 and Y=0111, at time T=0 both of the sum and the carry signals assume any value.

At T=T_FA, the first and second FAs compute the sum and do not generate any carries. Simultaneously, the third FA computes the sum and generates the carry to the fourth FA and the fourth FA computes the sum and does not generate any carry.

At T=2T_FA, the first, second and third FAs have the same sum and carries. Simultaneously, the fourth FA computes the sum and generates the output carry. No changes occur after this point in time. It is clear that we need two cycles of FA delay till the final sum is calculated correctly and hence can be used in further calculations.

Table 2.4: Second ripple-carry adder addition example.

Time	$x_3x_2x_1x_0$	$y_3y_2y_1y_0$	$c_{out}c_3c_2c_1c_0$	$s_3s_2s_1s_0$
T=0	1111	0001	xxxxx	xxxx
T=T_FA	1111	0001	00010	1110
T=2T_FA	1111	0001	00110	1100
T=3T_FA	1111	0001	01110	1000
T=4T_FA	1111	0001	11110	0000

In table 2.4, we have two operands X=1111 and Y=0001, at time T=0 both of the sum and the carry signals assume any value.

At T=T_FA, the first FA computes the sum and generates the carry to the second FA. Simultaneously, the second, third and fourth FAs compute the sum and do not generate any carries.

At T=2T_FA, only the second FA computes the sum and generates the carry to the third FA. Simultaneously, the first, third and fourth FAs have the same sum and carries.

At T=3T_FA, only the third FA computes the sum and generates the carry to the fourth FA. Simultaneously, the first, second and fourth FAs have the same sum and carries.

At T=4T_FA, only the fourth FA computes the sum and generates the output carry. Simultaneously, the first, second and third FAs have the same sum and carries. It is clear that, we need four cycles of FA delay till the final sum is calculated correctly and hence can be used in further calculations.

We can conclude that in a ripple-carry adder with n bits operands, the worst case delay is the FA delay multiplied by n. So the delay in ripple-carry adder is proportional to n. This long carry propagation chains must be dealt with in order to speed up the addition.

2.5 Carry look-ahead adder

Carry look-ahead adder is the most commonly used architecture for improving the carry propagation chain [9]. The main principle in carry look-ahead adder is to try to generate all the carries propagating between the FA cells in parallel instead of waiting for them to propagate from the least significant cell to the most significant cell.

All carry bits depend on the operands X and Y, which are available at time T=0, so we can use this information to generate all carries. However, this may lead to a large number of gates, hence we need to think in a different approach to accelerate the carry propagation chain and have a reasonable number of gates.

By analyzing how carries are generated and the condition to propagate them from stage to the next stage, we can decrease the number of gates required to generate the carry at each stage.

A carry is generated at stage i if both $x_i=y_i=1$, and in this case no dependency on the incoming carry from the previous stage. Also a carry is propagated from stage i to the next stage if x_iy_i=10, 01 or 11, stages in which $x_i=y_i=0$ can not propagate a carry.

We can define the following logic functions using the logical AND operation and the logical OR operation:

$$G_i = x_i \,.\, y_i \tag{2.5}$$

G_i represents the generated carry.

$$P_i = x_i + y_i \tag{2.6}$$

P_i represents the propagated carry.

At any stage i the output carry c_{i+1} is related to the input carry c_i and the input operands x_i and y_i by the following logic equation

$$c_{i+1} = x_i \,.\, y_i + c_i \,.\, x_i + c_i \,.\, y_i = x_i \,.\, y_i + c_i \,.\, (x_i + y_i) \tag{2.7}$$

Equation 2.7 can be written using the generation and propagation functions.

$$c_{i+1} = G_i + c_i \,.\, P_i \qquad (2.8)$$

c_i can be written as $c_i = G_{i-1} + c_{i-1} \,.\, P_{i-1}$

$$c_{i+1} = G_i + (G_{i-1} + c_{i-1} \,.\, P_{i-1}).\, P_i \qquad (2.9)$$

$$c_{i+1} = G_i + G_{i-1} \,.\, P_i + c_{i-1} \,.\, P_{i-1} \,.\, P_i \qquad (2.10)$$

Further substitutions result in

$$c_{i+1} = G_i + G_{i-1} \,.\, P_i + G_{i-2} \,.\, P_{i-1} \,.\, P_i + \ldots\ldots + c_0 \,.\, P_0 \,.\, P_1 \ldots\, P_i \qquad (2.11)$$

Equation 2.11 allow us to calculate all carries in parallel from the original operands $x_{n-1},x_{n-2},\ldots x_0$ and $y_{n-1},y_{n-2},\ldots y_0$ and the input carry c_0 .

For four bit carry look-ahead adder, the following equations relate the carries to the input operands and the input carry:

$$\begin{aligned}
c_1 &= G_0 + c_0.P_0, \\
c_2 &= G_1 + G_0.P_1 + c_0.P_0.P_1, \qquad (2.12) \\
c_3 &= G_2 + G_1.P_2 + G_0.P_1.P_2 + c_0.P_0.P_1.P_2, \\
c_4 &= G_3 + G_2.P_3 + G_1.P_2.P_3 + G_0.P_1.P_2.P_3 + c_0.P_0.P_1.P_2.P_3.
\end{aligned}$$

The following equations relate the sum to the input operands and the carries.

$$\begin{aligned}
s_0 &= x_0 \oplus y_0 \oplus c_0 \\
s_1 &= x_1 \oplus y_1 \oplus c_1 \qquad (2.13) \\
s_2 &= x_2 \oplus y_2 \oplus c_2 \\
s_3 &= x_3 \oplus y_3 \oplus c_3
\end{aligned}$$

To estimate the delay of the above circuit, let us assume ΔT is the delay of a single gate. We need ΔT to generate all P_i and G_i, also we need $2\Delta T$ to generate all c_i assuming a two level gate implementation, also we need another $2\Delta T$ to generate the sum digits assuming a two level gate implementation, so we need a total of $5\Delta T$ independent of n the number of bits in each operand.

For a large value of n, a large number of gates are required and also each gate has a very large fan-in, which may make this method impractical. Therefore the trend is to divide the n bits into groups and have a separate carry look-ahead adder in each group and it may be connected using the ripple-carry adder method. The common group size in today ICs is 4 [8].

As the number of bits increases, we can use more levels of carry-look-ahead generators to speed up the addition. The required number of levels for maximum speed up approaches log n, hence the overall addition time of a carry look-ahead adder is proportional to log n [4,10].

2.6 Carry-select adder

In the traditional ripple-carry adder, the carry is propagated from the least significant bit to the most significant bit which leads to a delay proportional to n. The carry-select adder is another architecture that provides a logarithmic speed-up [12].

The technique behind this architecture is to divide our n bits operand into smaller groups each L bits length [21], and then generate two sets of outputs for this small group: the first set assuming that the input carry is 0 and the other set assuming the input carry is 1. When the input carry is known, it is used to select the correct set of outputs.

If we apply this method for the whole n bits, no improvement occurs since we will wait till the carry propagates through the whole n bits and hence the delay is the same delay as the ripple-carry adder and the area is larger than that of the ripple-carry adder circuit.

We need to divide the n bits into smaller groups and apply this idea to each group separately. In this way, the serial carry propagation inside the separate groups can be done in parallel which results in increasing the overall speed of the circuit.

Each small group is divided into a smaller group and the same concept is applied. A logarithmic speed up is achieved using this concept.

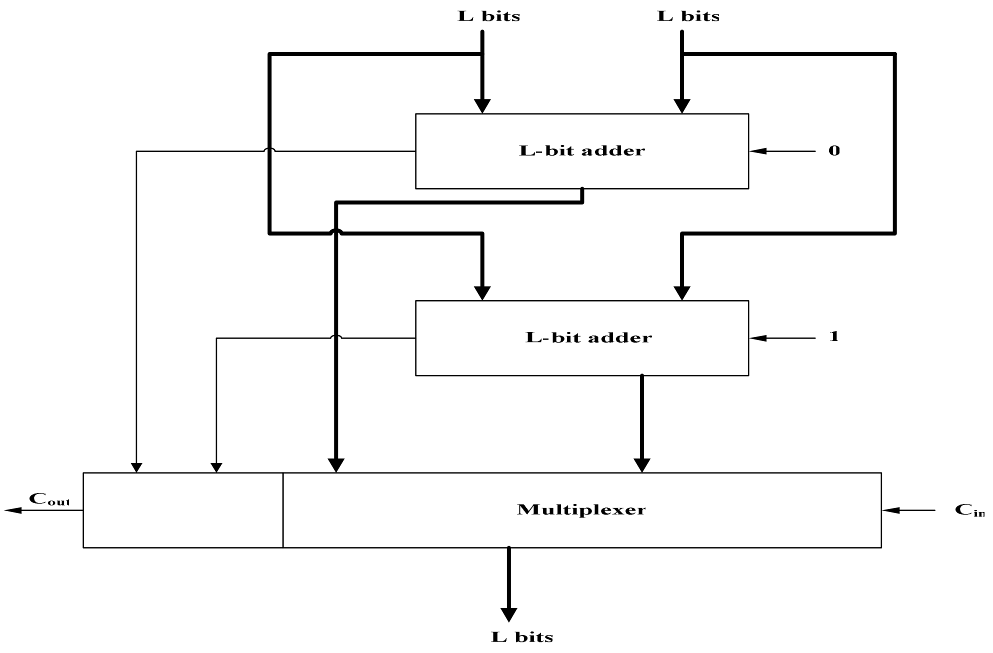

Figure 2.4: L bit carry-select adder.

2.7 Signed digit number system

2.7.1 Introduction

In conventional digital computers integers are represented as binary numbers of a fixed length. Each digit assumes a value between $\{0,.....,r-1\}$ where r is the radix of the system.

In conventional binary system $r=2$, each digit assumes either 0 or 1, the relation between an integer number and its binary representation is:

$$X= x_{n-1}\, 2^{n-1} + x_{n-2}\, 2^{n-2} + \ldots + x_1\, 2^1 + x_0\, 2^0 = \sum x_i\, 2^i. \qquad (2.14)$$

A conventional binary number system is a non-redundant system, which means each number has a unique representation.

The signed digit number system is a redundant number system that eliminates the carry propagation chain and hence achieves high speed circuit operation.

2.7.2 Theory of SD number system

SD number system allows its digits to have the following digit set [10]:

$$\{\bar{\alpha},\ldots\ldots\ldots\ldots,0,\ldots\ldots\ldots\ldots,\gamma\}$$

Where $\bar{\alpha} = -\alpha$, α is not necessarily equal to $-\gamma$ and γ is not necessarily equal to $(r-1)$.

However, we will concentrate our analysis on a symmetric SD number system where $\alpha = -\gamma$. Symmetric SD number system allows its digits to have the following digit set [8]:

$$\{\overline{r-1},\overline{r-2},\ldots\ldots\ldots,\bar{1},0,1\ldots,\ldots\ldots,(r-2),(r-1)\}$$

Where r is the radix of the system, $\overline{r-1} = -(r-1)$.

For r=8, the allowed digit set are $\{\bar{7},\bar{6},\ldots\ldots\ldots,\bar{2},\bar{1}$, 0, 1, 2,……, 6, 7$\}$
And, if n=2 the range is $(\bar{7})(\bar{7}) \leq$ X $\leq$ (7)(7), which include 155 numbers, each digit has 15 possibilities, so we have 15^2 = 225 representations.
We can conclude that, some numbers have more than one representation since the total available numbers are 155 while the whole range has 225 representations, so this number system is redundant. For example:
1 can be represented as (0)(1)=(1)($\bar{7}$)=1, also -3 can be represented as (0)($\bar{3}$)=($\bar{1}$)(5)= -3. The number of redundant representations is 225 -155 =70, and hence there is 45% redundancy.

For r=10, the allowed digit set are $\{\bar{9},\bar{8},\bar{7},\ldots\ldots\ldots,\bar{1},0,1,,\ldots\ldots\ldots,7,8,9\}$ and, if n=2 the range is $(\bar{9})(\bar{9}) \leq X \leq (9)(9)$, which include 199 numbers, each digit has 19 possibilities, so we have 19^2 = 361 representations.

We can conclude that, some numbers have more than one representation since the total available numbers are 361 while the whole range has 199 representations, so this number system is redundant. For example:

2 can be represented as (0)(2)=(1)($\bar{8}$)=2, also -5 can be represented as (0)($\bar{5}$)=($\bar{1}$)(5)= -5. The number of redundant representations is 361 -199 =162, and hence there is 81% redundancy.

The redundancy in the SD number system eliminates carry propagation chains and hence achieves a high speed circuit operation, however high level of redundancy is very costly since we need a large number of bits to represent each digit.

The amount of redundancy is reduced by restricting the allowed digit set to

$$X_i = \{\bar{a},\overline{a-1},\ldots,\bar{1}, 0, 1,\ldots,a-1,a\} \qquad \left\lceil \frac{r-1}{2} \right\rceil \leq a \leq r-1 \tag{2.15}$$

Where ceiling $\lceil x \rceil$ of a number x is the smallest integer that is larger than or equal to x. At least r different digits are needed to represent a number in a radix r system and for $\bar{a} \leq x_i \leq a$, we have $2a+1$ digits. Therefore the inequality $r \leq (2a+1)$ must be satisfied.

2.7.3 SD number system addition algorithm

Consider the following operation:

$$(x_{n-1},\ldots\ldots, x_0) + (y_{n-1},\ldots\ldots, y_0) = (z_{n-1},\ldots\ldots z_0) \tag{2.16}$$

The main benefit of the SD number system is to break the carry propagation chain by having the sum digit z_i depending only on the four operand digits x_i,

y_i, x_{i-1} and y_{i-1}. If this is achieved, the addition time does not depend on the length of the operands which results in a high speed circuit operation. An addition algorithm that achieves this independence consists of the following steps:

Step 1: Compute an internal sum s_i and a carry digit c_i:

$$s_i = x_i + y_i - r\ c_i \tag{2.17}$$

Where

$$c_i = \begin{cases} 1 & \text{if } (x_i + y_i) \geq a \\ \bar{1} & \text{if } (x_i + y_i) \leq \bar{a} \\ 0 & \text{if } |x_i + y_i| < a \end{cases} \tag{2.18}$$

Step 2: Calculate the final sum $z_i = s_i + c_{i-1}$. (2.19)

Example:

If we select a =7 for r=10, then $x_i = \{\bar{7}, \bar{6}, \bar{5}, \ldots\ldots, \bar{1}, 0, 1, \ldots..5, 6, 7\}$. Step 1 in the previous addition algorithm becomes $s_i = x_i + y_i - 10\ c_i$ and

$$c_i = \begin{cases} 1 & \text{if } (x_i + y_i) \geq 7 \\ \bar{1} & \text{if } (x_i + y_i) \leq \bar{7} \\ 0 & \text{otherwise} \end{cases}$$

x_i		5	6	5	7
y_i	+	1	3	7	4
z_i		7	0	3	1

Thus, instead of performing the addition of two decimal numbers, such as 5657 and 1374, in the conventional decimal number system, in which the carry propagates from the least significant digit to the most significant one, we have no carry propagation chain in the following example.

$$\begin{array}{llccccc}
x_i & & & 5 & 6 & 5 & 7 \\
y_i & + & & 1 & 3 & 7 & 4 \\
c_i & & 0 & 1 & 1 & 1 & \\
s_i & & & 6 & \bar{1} & 2 & 1 \\
z_i & & & 7 & 0 & 3 & 1
\end{array}$$

Note that the carry bits were shifted to left to simplify the execution of the second step of the algorithm.

We use the previous addition algorithm as a method for converting the conventional decimal number system into the SD number system by considering each digit as the sum (x_i + y_i).

Example:

We use the previous addition algorithm to convert the decimal number 48957 into SD number system.

$$\begin{array}{lcccccc}
x_i{+}y_i & & 4 & 8 & 9 & 5 & 7 \\
c_i & 0 & 1 & 1 & 0 & 1 & \\
s_i & & 4 & \bar{2} & \bar{1} & 5 & \bar{3} \\
z_i & & 5 & \bar{1} & \bar{1} & 6 & \bar{3}
\end{array}$$

To convert a number from the SD representation to the conventional representation, one subtracts the digits with the negative weight from the digits with the positive weight. For $5\ \bar{1}\ \bar{1}\ 6\ \bar{3}$ we obtain:

$$\begin{array}{r} 5\;\;0\;\;0\;\;6\;\;0 \\ \_\;\;\; 0\;\;1\;\;1\;\;0\;\;3 \\ 4\;\;8\;\;9\;\;5\;\;7 \end{array}$$

To guarantee that no new carry is generated, the sum digit z_i calculated from $s_i + c_{i-1}$, must satisfy $|\, z_i \,| \le a$. Since $|\, c_{i-1} \,| \le 1$, the condition $|\, s_i \,| \le a-1$ has to be satisfied for all possible values of x_i and y_i.

For example, the largest value that $x_i + y_i$ can assume is $2a$, for which $c_i = 1$ and $s_i = 2a-r$. The inequality $s_i = 2a-r \le a-1$ is clearly satisfied, since $a \le r-1$. However if $x_i + y_i = a$, which is the smallest value for which c_i is still 1, then $s_i = a-r < 0$. Substituting $|\, s_i \,| = r-a$ into $|\, s_i \,| \le a-1$ yields the inequality $2a \ge r+1$, hence the selected digit set must satisfy

$$\left\lceil \frac{r+1}{2} \right\rceil \le a \le r-1 . \tag{2.20}$$

Example:

SD decimal number systems must satisfy $a \ge 6$ to guarantee that no new carries will be generated in the previous algorithm.

Although the previous restriction, for $a < 6$ we can guarantee that no new carries will be generated. This is done by examining the digits x_{i-1}, y_{i-1},x_i and y_i and make a proper selection of s_i and c_{i-1} to ensure that no new carry is generated in the operation $z_i = s_i + c_{i-1}$.

Kornerup in [14] and Parhami in [17] have discussed this concept in details. In the next section, we will discuss the binary SD numbers and we will apply this concept to ensure carry free addition and no new carry is generated in the second step of the addition algorithm.

2.7.4 Binary SD numbers

For r=2, the allowable digit set is $\{\bar{1}, 0, 1\}$. In other words, a must equal 1. The interim sum and the carry in the addition algorithm are

$$s_i = x_i + y_i - 2\, c_i \qquad (2.21)$$

$$c_i = \begin{cases} 1 & \text{if } (x_i + y_i) \geq 1 \\ \bar{1} & \text{if } (x_i + y_i) \leq \bar{1} \\ 0 & \text{otherwise} \end{cases} \qquad (2.22)$$

The rules are summarized in the following table.

Table 2.5: Binary SD number addition rules.

x_iy_i	c_i	s_i
(0)(0)	0	0
(0)(1)	1	$\bar{1}$
(0)($\bar{1}$)	$\bar{1}$	1
(1)(0)	1	$\bar{1}$
(1)(1)	1	0
(1)($\bar{1}$)	0	0
($\bar{1}$)(0)	$\bar{1}$	1
($\bar{1}$)(1)	0	0
($\bar{1}$)($\bar{1}$)	$\bar{1}$	0

We must note that the condition $\left\lceil \frac{r+1}{2} \right\rceil \leq a$ is not satisfied in the binary case, and consequently there is no guarantee that a new carry will not be generated in the second step of the algorithm.

In the next example, we add two binary numbers using the SD addition algorithm, it is clear that no carry will be generated, however in the conventional binary number system a carry will be propagated from the least significant bit to the most significant bit.

$$\begin{array}{lccccccc} x_i & & & 1 & 1 & 1\ldots\ldots 1 & 1 \\ y_i & + & & 0 & 0 & 0\ldots\ldots 0 & 1 \\ c_i & & 1 & 1 & 1 & 1\ldots\ldots 1 & \\ u_i & & & \bar{1} & \bar{1} & \bar{1}\ldots\ldots \bar{1} & 0 \\ S_i & & 1 & 0 & 0 & 0\ldots\ldots 0 & 0 \end{array}$$

Suppose we need to add two SD numbers with $\bar{1}$ digit, new carries may occur during this operation and this carries need to propagate from stage to another stage. For example, if x_{i-1} y_{i-1} = (0)(1), then $c_{i-1} = 1$; and if $x_iy_i = (0)(\bar{1})$, then $s_i = 1$, yielding $z_i = s_i + c_{i-1} = 1+1$. Thus a new carry is generated.

Examining the rules in table 2.5, one can verify that the combination $c_{i-1} = s_i = 1$ occurs when $x_iy_i = (0)(\bar{1})$ or $(\bar{1})(0)$ and x_{i-1} y_{i-1} equals either (1)(1),(0)(1) or (1)(0). We can avoid setting $s_i = 1$ in these cases by selecting $c_i = 0$ and therefore making s_i equal $(\bar{1})$. We should not, however, change the entry for $x_iy_i = (0)(\bar{1})$ or $(\bar{1})(0)$ in table 2.5.

Similarly, the combination $c_{i-1} = s_i = (\bar{1})$ occurs when x_iy_i = (0)(1) or (1)(0) and x_{i-1} y_{i-1} equals $(\bar{1})(\bar{1})$, $(0)(\bar{1})$ or $(\bar{1})(0)$. We can avoid setting $s_i = (\bar{1})$ by selecting in these cases (and in these case only) $c_i = 0$ and therefore $s_i = 1$.

In summary, we can ensure that no new carries will be generated by examining the two bits to the right x_{i-1} y_{i-1} when determining s_i and c_i, arriving at the rules in the following table [8].

Table 2.6: Modified binary SD number addition rules.

x_iy_i	x_{i-1} y_{i-1}	c_i	s_i
(0)(0)	-	0	0
(0)(1)	Neither is $\bar{1}$	1	$\bar{1}$
(0)(1)	At least one is $\bar{1}$	0	1
(0)($\bar{1}$)	Neither is $\bar{1}$	0	$\bar{1}$
(0)($\bar{1}$)	At least one is $\bar{1}$	$\bar{1}$	1
(1)(0)	Neither is $\bar{1}$	1	$\bar{1}$
(1)(0)	At least one is $\bar{1}$	0	1
(1)(1)	-	1	0
(1)($\bar{1}$)	-	0	0
($\bar{1}$)(0)	Neither is $\bar{1}$	0	$\bar{1}$
($\bar{1}$)(0)	At least one is $\bar{1}$	$\bar{1}$	1
($\bar{1}$)(1)	-	0	0
($\bar{1}$)($\bar{1}$)	-	$\bar{1}$	0

Binary SD numbers are very useful in breaking the carry propagation chain and hence achieving high speed arithmetic. This is done by making each digit z_i dependant only on the four operands x_i, y_i, x_{i-1} and y_{i-1} and hence the addition time is independent of the length of the operands.

Eliminating carry propagation chains when adding binary numbers speeds up operations like multiplications and divisions, whose execution usually includes a large number of add/subtract operations.

The encoding of digits 0, 1 and $\bar{1}$ is an important concern when dealing with SD arithmetic. Two common encodings are shown in the following table [8].

Table 2.7: Encoding of signed digit.

x	Encoding 1 $x^h\ x^l$	Encoding 2 $x^h\ x^l$
0	00	00
1	01	01
$\bar{1}$	10	11

Encoding 1 satisfies the simple relation:

$$X= x^l - x^h, \tag{2.23}$$

Both combination (0)(0) and (1)(1) are valid numerical value for 0.

Encoding 2 satisfies the following relation:

$$X= x^l - 2x^h \tag{2.24}$$

In this encoding (1)(0) is not a valid binary value.

Another concern arises, which is the need to convert the result in SD representation to its conventional two's complement representation. The main algorithm for conversion is to subtract the sequence (x^h_{n-1},……..x^h_0) from the sequence (x^l_{n-1}, ……………..x^l_0) using two's complement arithmetic. This algorithm is valid for encoding 1 conversion.

The main algorithm in the case of encoding 2 is to shift the sequence (x^h_{n-1},……..x^h_0) to the left one digit, then subtract this sequence from the sequence (x^l_{n-1}, ……………..x^l_0) using two's complement arithmetic.

2.8 Summary

In this chapter, the basic adder architectures like ripple-carry adder, carry look-ahead adder and carry-select adder have been discussed. Also a discussion of the basic concepts of the SD number system and its addition algorithm has been done. By using this redundant number system, carry free addition is achieved and as a result high speed circuit operation.

Chapter 3

3- Hybrid signed digit number system

3.1 Introduction

The SD number representation is used to speed up arithmetic operations. The redundancy in this system is used to limit the length of the carry propagation chain to only one digit position, so that the addition of two numbers is done in a fixed time independent of the word length. The SD adder cell is more complex than the adder cell of the unsigned digits; more area is the cost of the constant addition time.

The SD and the two's complement number representations are at the two extremes. In the SD number system more area for the adder cell, however carry propagation is limited to one digit position. In the two's complement number system less area, however the carry propagates across the entire word length.

In this chapter, we discuss the hybrid signed digit (HSD) number system that is a mixture of the SD number system and the conventional number system. A characterization of the HSD adder is done by changing its distance (d) parameter and analyzing its effect on the adder delay, area and ADP.

3.2 SD adder cell

Takagi et al in [18, 19] propose a circuit for adding two signed digits. The input digits and the output sum satisfy encoding 2 discussed in the previous chapter. However, the input and the output carry are expressed as the difference between two variables.

$$c_i = v_i - p_i. \quad (3.1)$$

The details of the logic equations and their derivations are found in [18, 19]. The following figure shows the logic diagram of the SD adder cell.

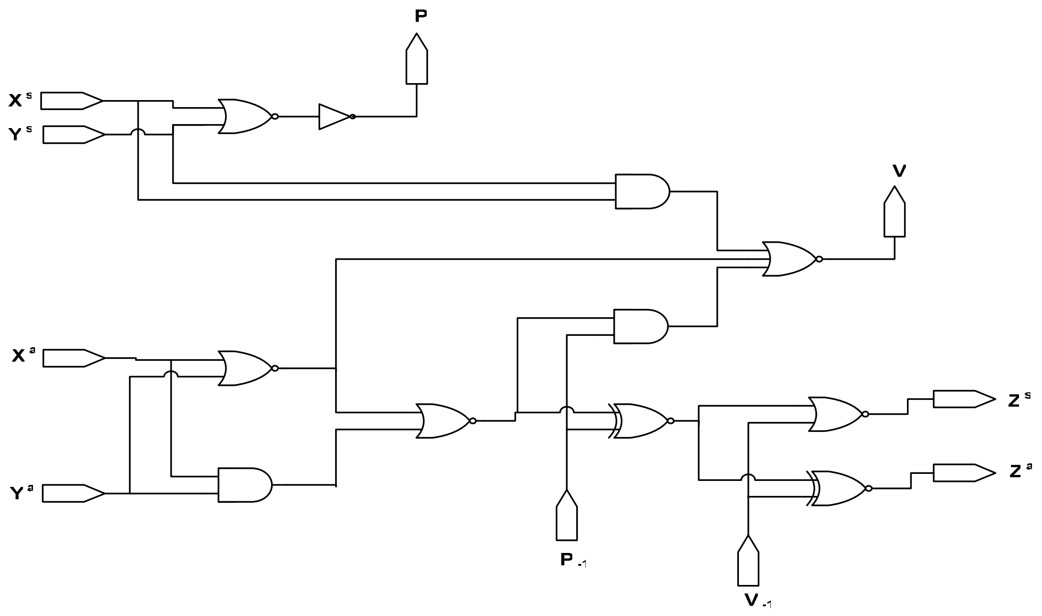

Figure 3.1: SD adder cell.

3.3 SD adder circuit design flow

In this section, we implement a SD adder circuit (32 digits) based on the SD adder cell discussed in the previous section. All architectures done in this chapter and the next chapters are 32 digits length.

The following design flow is done:

1. Develop a behavior model for the SD adder circuit.
2. Develop an RTL description for the SD adder circuit.
3. Develop a test bench to test the functionality of the circuit.
4. Apply 10 million random input vectors to test the functionality of the circuit (Functional verification).
5. Synthesize the circuit on ASIC technology (Artisan TSMC 0.13), the library used is "ss_1v08_125c".
6. Post-synthesis verification is done.

The ASIC technology used is Artisan (TSMC 0.13); the library "ss_1v08_125c" has the following conditions:

1. Slow Process.
2. Supply voltage is 1.08 V.
3. Temperature is $125°C$.

The result of the area can be reported in μ_m^2 or in gate count. We use the second approach, so that the result of the area is reported in gate count and our reference gate is the 2 input NAND gate (the minimum area NAND gate).

The maximum frequency of the SD adder circuit is 1408.5 MHZ, while its area is 1398 gates. However, the maximum frequency of the conventional ripple-carry adder is 72 MHZ, while its area is 650 gates.

We can conclude that, the SD adder circuit trades off area for a higher speed. So in the following section we discuss the HSD adder, which is an intermediate solution between the SD adder circuit and the ripple-carry adder, in order to make a suitable trade off between the speed and the area of the circuit.

3.4 HSD adder

Phatak and Koren propose a HSD number representation [13], They propose an architecture that is a mixture of the SD number system and the conventional number system. They choose some of the digits to be signed digits while leave the remaining digits as unsigned digits.

This representation allows the maximum carry propagation chains to have any desired values between one and the full word length. They prove that the maximum length of a carry propagation chain equals (d+1), where d is the longest distance between neighboring signed digit [13].

This representation allows the designer to make a trade off between the area, the speed and the power consumption of the circuit by changing the distance (d) to the value that satisfies the required constraints.

This architecture consists of signed digits and unsigned digits. The signed digit allow the following digit set $\{\bar{1}, 0, 1\}$, while the unsigned digit allow the following set $\{0, 1\}$. The carry signals between all digit positions (signed or unsigned) assume the following set $\{\bar{1}, 0, 1\}$.

In the HSD representation, it is preferred to have the most significant digit as a signed digit, in order to incorporate enough negative numbers.

The range covered by an n digit HSD representation depends on the numbers of the signed digits and their positions. We assume that our system is a binary HSD number system hence $r = 2$. The range covered for n bit HSD with d_0 (d_0 means distance (d) is equal to 0) is [-2^n-1, 2^n-1], d_0 means all digits are signed digits and hence it is the SD system.

For any other d, the largest positive number has the same value as the value for d_0; however the smallest negative number depends on d and the exact positions of the signed digits. Its value is obtained by setting all unsigned digits to 0 and all signed digits to $\bar{1}$.

For d_{n-1} the range is [-2^{n-1}, 2^n-1], thus the range of an HSD representation with $d > 0$ includes fewer negative numbers than the positive ones.

In the following sections, we do a characterization of the HSD adder and analyze the effect of changing d on the area, delay and ADP of the circuit.

3.5 HSD adder implementation

Addition of bits at the unsigned digit position:

Inputs to the cell are the bits a_{i-1}, b_{i-1} and the incoming carry $c_{i-2.}$ The outputs are the carry out c_{i-1} and the final sum e_{i-1}. The output digit e_{i-1} is unsigned and can take one of the two values 0 or 1. The carries c_{i-2} and c_{i-1} can take values in $\{\bar{1}, 0, 1\}$ and need two bits for encoding. We encode each of these carries using two binary variables, each requiring a single bit ($c_i = v_i - w_i$). The details of the logic equations and their derivations are found in [13]. The implementation of the unsigned digit adder cell of the HSD adder is shown in the following logic diagram.

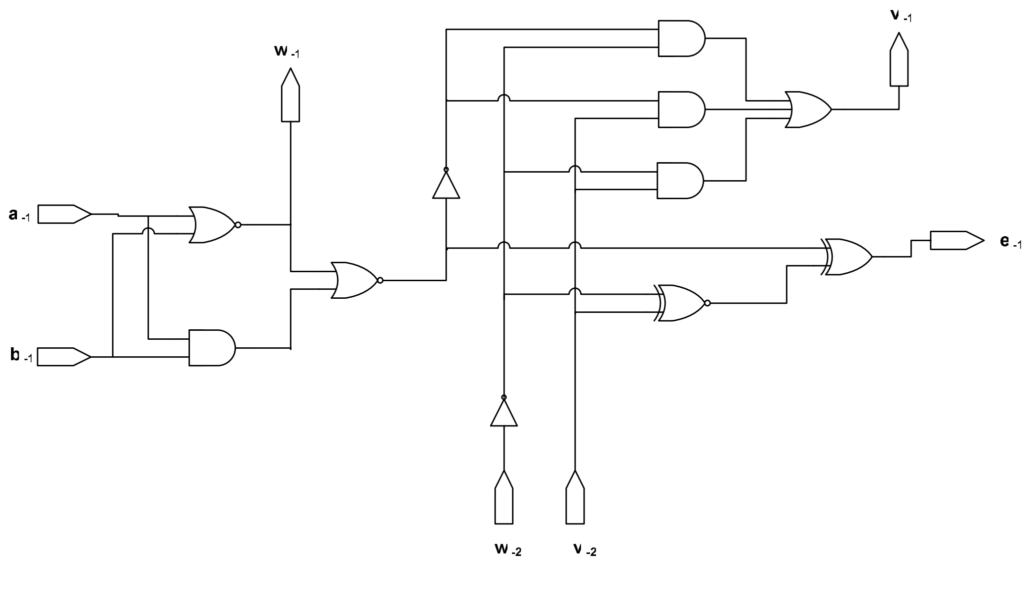

Figure 3.2: Unsigned digit adder cell of a HSD adder.

Addition of bits at the signed digit position:

The inputs to this cell are the signed digit x_i, y_i and the carry signals w_{i-1}, v_{i-1}. A signed digit x is encoded into two bits $x^s\, x^a$, which satisfy encoding 2 discussed in the previous chapter. The outputs of the cell are the carry signals w_i, v_i and the bits that represent the output signed digit $z^s\, z^a$. The details of the logic equations and their derivations are found in [13]. The implementation of the signed digit adder cell of the HSD adder is shown in the following logic diagram.

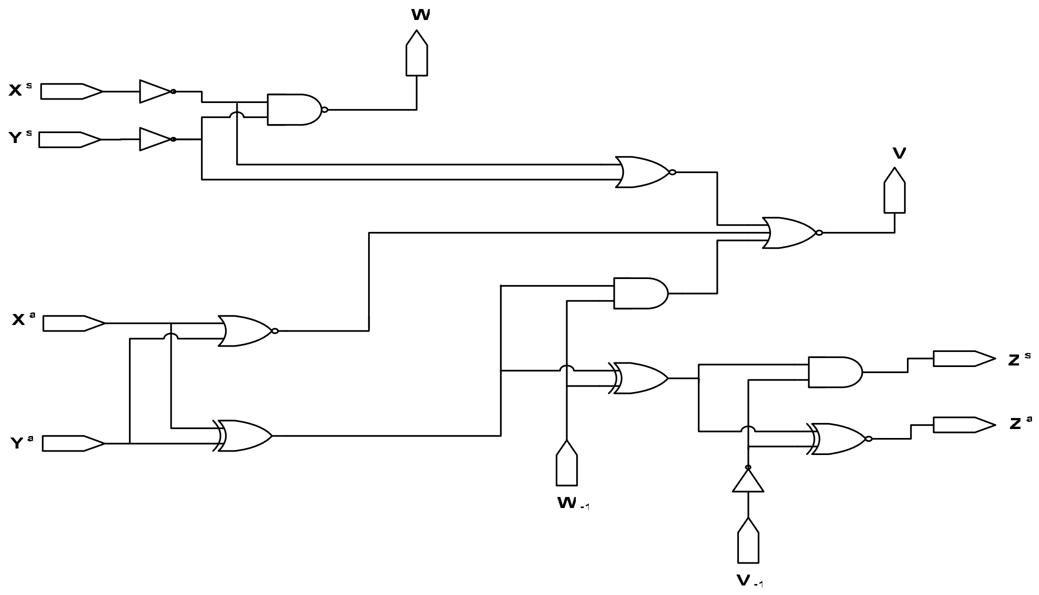

Figure 3.3: Signed digit adder cell of a HSD adder.

3.6 HSD adder characterization

In this section, a characterization of the HSD adder is done by changing its distance parameter from 0 to 32 and analyzing its effect on the area, delay and ADP of the circuit.

The design flow discussed in section 3.3 is done, however the synthesis is done on three libraries "ss_1v08_125c", "tt_1v20_25c" and "ff_1v26_m40c".

1. library "ss_1v08_125c" has the following conditions:
   - Slow Process.
   - Supply voltage is 1.08 V.
   - Temperature is $125°C$.
2. library "tt_1v20_25c" has the following conditions :
   - Typical Process.
   - Supply voltage is 1.2 V.
   - Temperature is $25°C$.
3. library "ff_1v26_m40c" has the following conditions :
   - Fast Process.
   - Supply voltage is 1.26 V.
   - Temperature is $-40°C$.

In the following sections, we refer to the fast library as "FF", refer to the typical library as "TT" and refer to the slow library as "SS".

In the following figure, different architectures of the HSD adder are illustrated. We refer to U as the unsigned digit adder cell and to S as the signed digit adder cell.

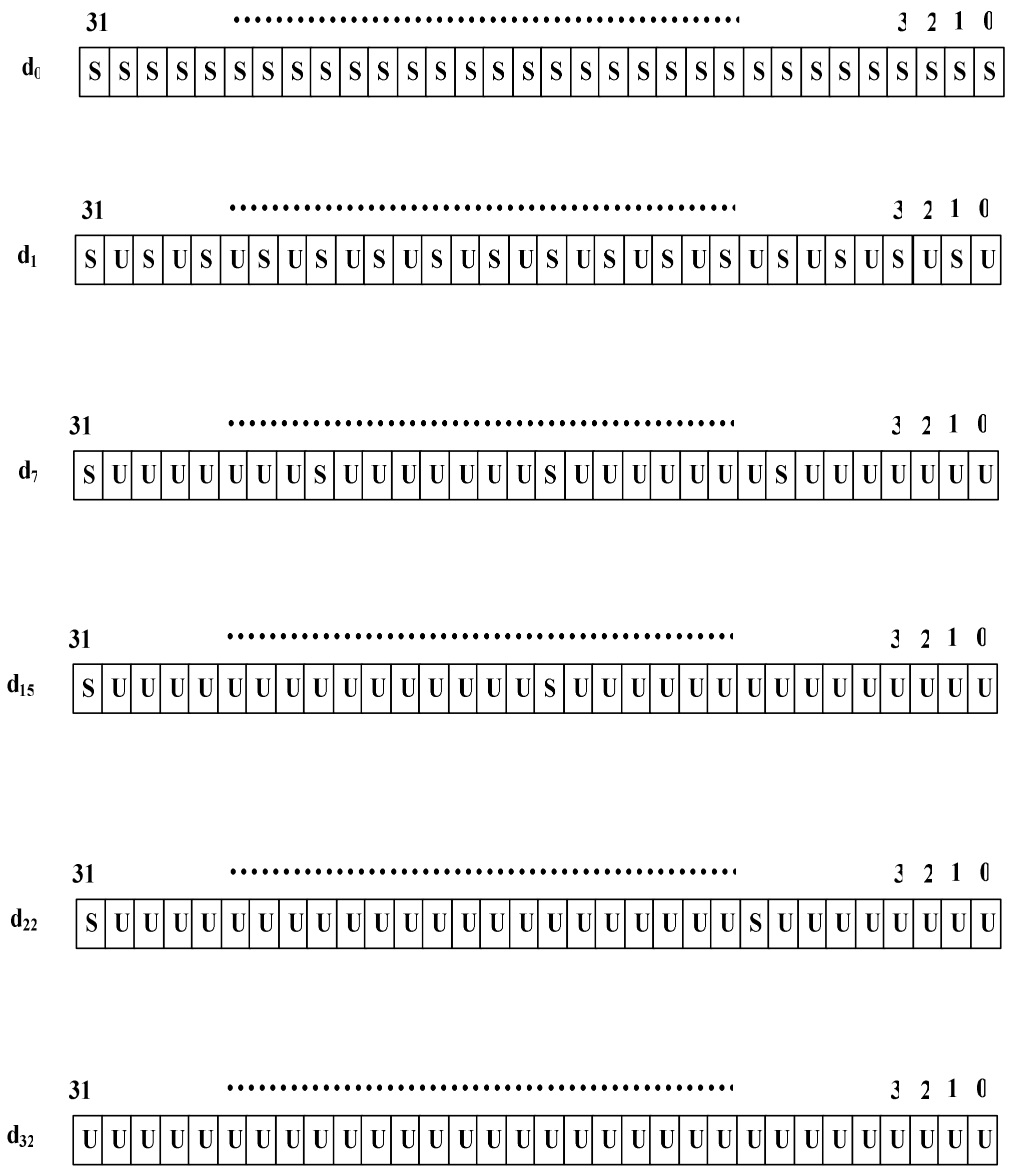
31
3 2 1 0
d0
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S
d1
S U S U S U S U S U S U S U S U S U S U S U S U S U S U S U S U
d7
S U U U U U U U S U U U U U U U S U U U U U U U S U U U U U U U
d15
S U U U U U U U U U U U U U U U S U U U U U U U U U U U U U U U
d22
S U U U U U U U U U U U U U U U U U U U U U U S U U U U U U U U
d32
U U U U U U U U U U U U U U U U U U U U U U U U U U U U U U U U

Figure 3.4: Examples of distances in a HSD adder circuit.

In the following section, the HSD adder characterization results are illustrated. The unit of the area is gate count, the unit of the delay is nano second (ns) and the unit of the ADP is (gate count * ns).

3.7 HSD adder characterization results

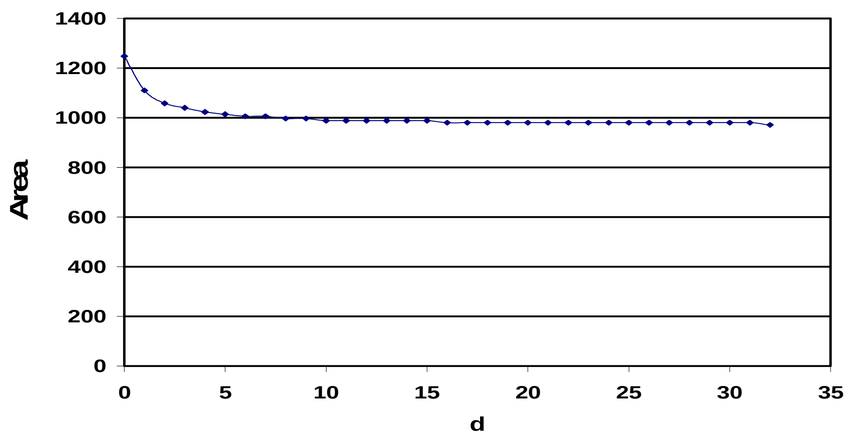

Figure 3.5: Area of the HSD adder (FF).

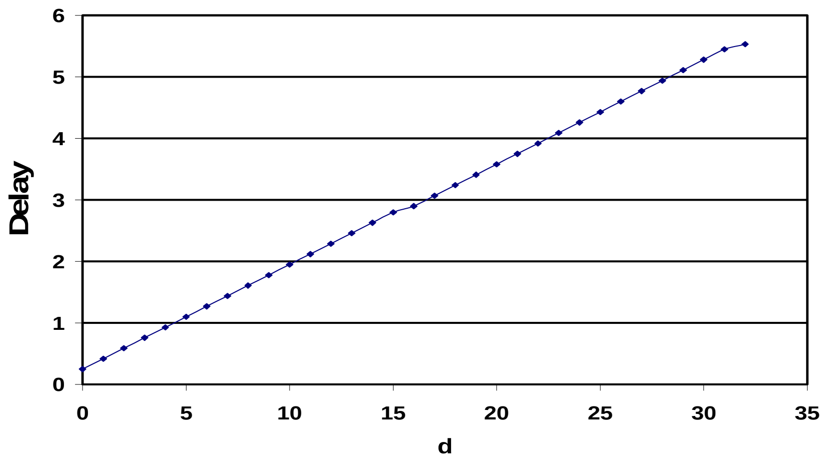

Figure 3.6: Delay of the HSD adder (FF).

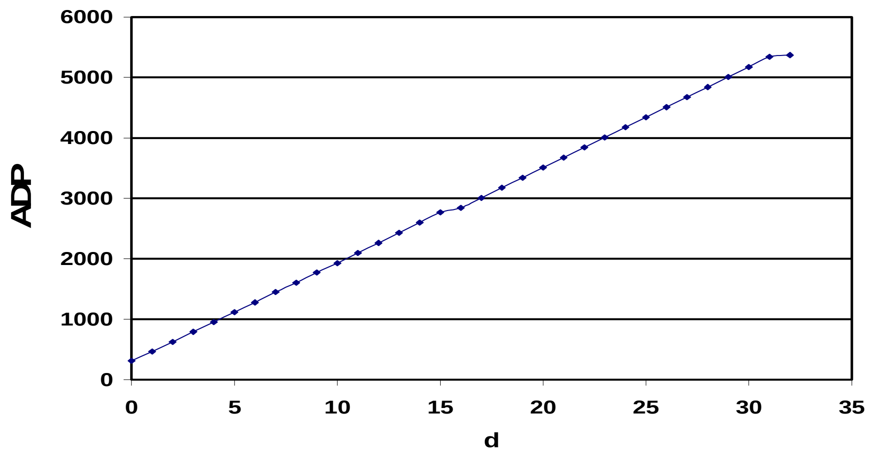

Figure 3.7: ADP of the HSD adder (FF).

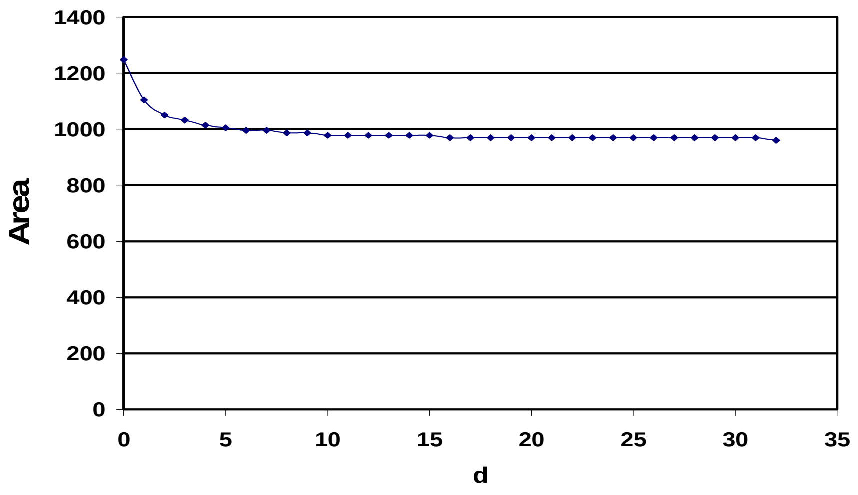

Figure 3.8: Area of the HSD adder (TT).

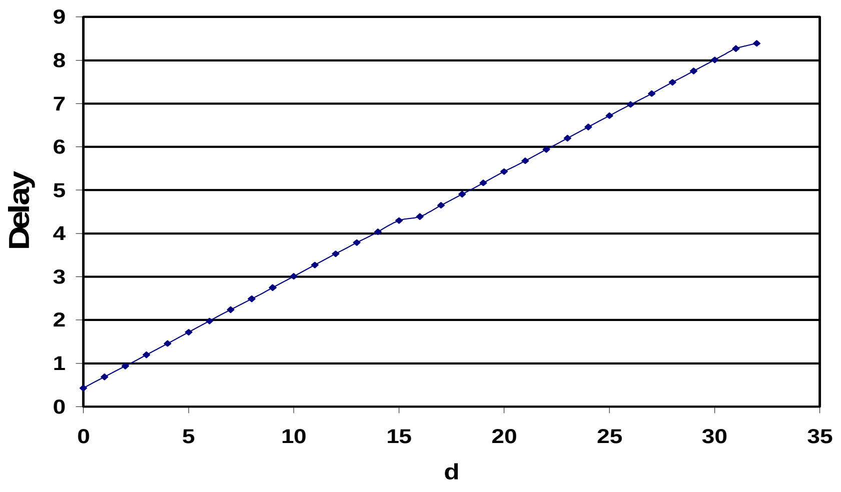

Figure 3.9: Delay of the HSD adder (TT).

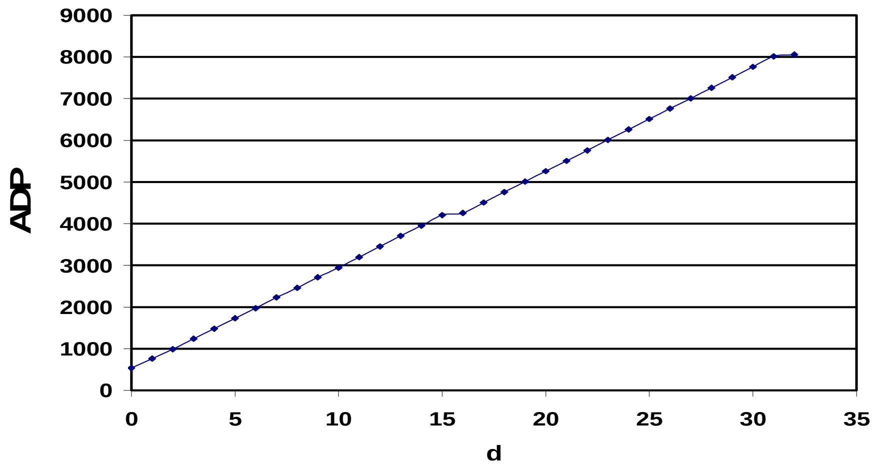

Figure 3.10: ADP of the HSD adder (TT).

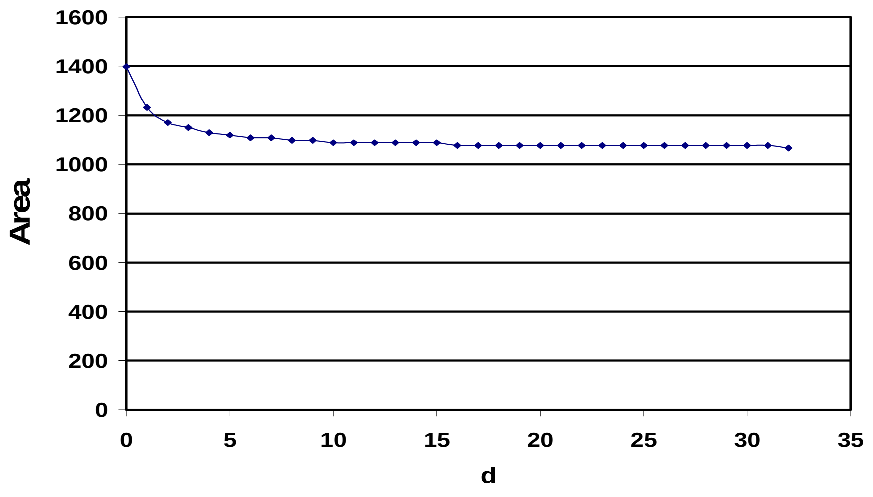

Figure 3.11: Area of the HSD adder (SS).

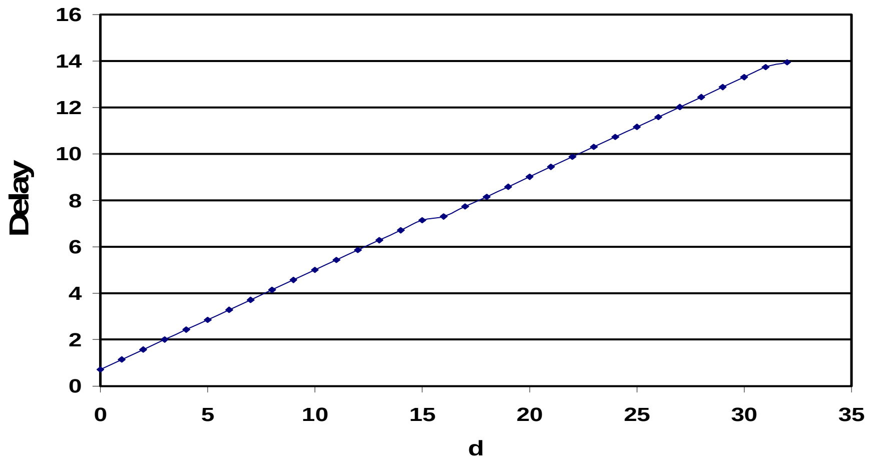

Figure 3.12: Delay of the HSD adder (SS).

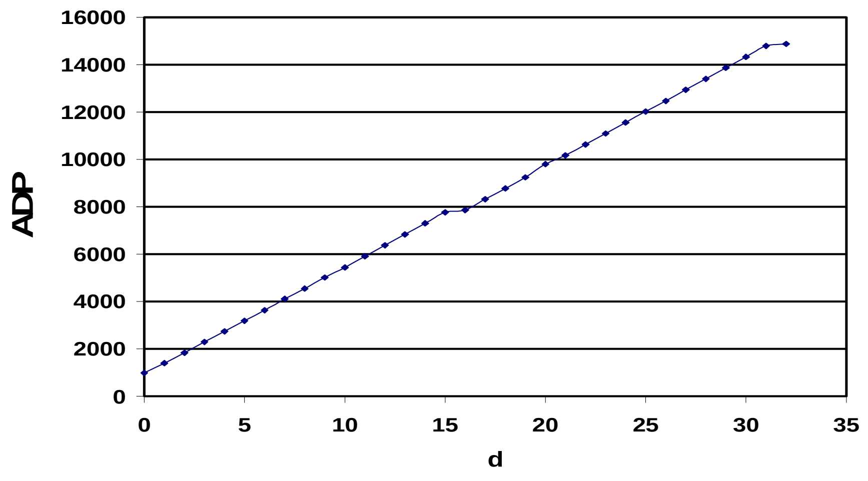

Figure 3.13: ADP of the HSD adder (SS).

3.8 Analysis of the HSD adder characterization results

In our analysis, we use the formula $\frac{value_{\max} - value_{\min}}{value_{\max}}$ to calculate the percentage of decrease in each metric.

Table 3.1: Analysis of the HSD adder characterization results.

Metric	Percentage of decrease formula	Percentage of decrease (FF)	Percentage of decrease (TT)	Percentage of decrease (SS)
Area	$\frac{d_0 - d_{32}}{d_0}$	22.19%	23%	23.6%
Delay	$\frac{d_{32} - d_0}{d_{32}}$	95.4%	94.8%	94.9%
ADP	$\frac{d_{32} - d_0}{d_{32}}$	94.1%	93.3%	93.3%

We conclude the following from the former table and the former figures:

1- Area characterization.

- As the number of signed digits decreases in the architecture, it is possible to decrease the area.

2- Delay characterization.

- As the number of signed digits increases in the architecture, it is possible to decrease the delay.
- We notice that at d_{16} the delay curve has a change; this is due to the change of the number of signed digits as the distance changed. The number of signed digits is settled to be 2 at d_{16}.

3- ADP characterization.

- As the number of signed digits increases in the architecture, it is possible to decrease the ADP.
- The ADP curve is similar to the delay curve which means that the decrease in the delay is dominant over the increase in the area.

Table 3.2: HSD adder (d_0) comparison with CLA adder.

Metric	Comparison formula	(FF)	(TT)	(SS)
Area	$\frac{d_0}{CLA}$	1.73	1.77	1.82
Delay	$\frac{CLA}{d_0}$	8.64	8.6	9
ADP	$\frac{CLA}{d_0}$	4.9	4.8	4.9

We can conclude that, the delay of the CLA adder is larger than the delay of the HSD (d_0) as well as the ADP. However, the area of the HSD (d_0) is larger than the area of the CLA adder.

3.9 Summary

In this chapter, the HSD number system has been analyzed. A characterization of the HSD adder has been done by changing its distance (d) parameter. It was found that, the most redundant adder is the optimum with respect to the delay and the ADP. However, the non-redundant adder is the optimum with respect to the area of the circuit.

Chapter 4

4- Hybrid signed digit adder/subtractor

4.1 Introduction

The architecture of the HSD adder is discussed in the previous chapter; also its area, delay and ADP are characterized by changing the distance (d) parameter. The subtraction operation is almost the addition with the negative of the number. We need to review the subtraction algorithm of the normal two's complement system, then deduce the subtraction algorithm of the SD system and finally deduce the subtraction algorithm of the HSD system.

4.2 Two's complement subtraction

In the conventional two's complement system, the operation X-Y is done in another way as X+ (-Y), where -Y is the two's complement of Y. As a result of this algorithm, the addition and the subtraction are implemented with the same circuit, but with small changes in the adder circuit to generate the two's complement needed to make the subtraction operation.

Two's complement for integer only number (not including fractional number) is done as follows:

1- Generate the one's complement of the number by changing each 1 to 0 and each 0 to 1. This is done using the inverter logic gate.
2- Add 1 to the one's complement of the number to generate the two's complement of the number.

In the following figure, a four bit adder/subtractor is illustrated [15]. If M=0, the output of all xor cells is Y and the input carry to the circuit is 0, so we do the addition operation. If M=1, the output of all xor cells is the inversion of Y which is the one's complement of Y and the input carry to the circuit is 1, so this 1 + one's complement of Y generate the two's complement of Y. Hence, we do the addition of X and the two's complement of Y which is the subtraction operation X-Y.

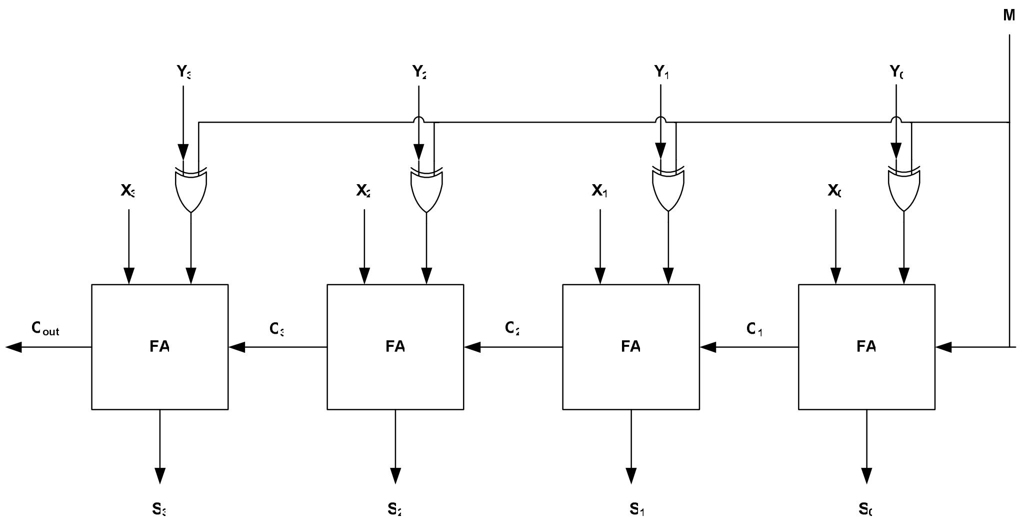

Figure 4.1: Four bit adder/subtractor.

4.3 SD subtraction

As discussed before, in the SD number system each digit allows the following digit set {$\bar{1}$, 0, 1}. Suppose we have the following SD number (1)(0)(0)(1) which is decimal 9, we can derive that -9 is ($\bar{1}$)(0)(0)($\bar{1}$). Also, suppose the following number ($\bar{1}$)(1)(0)($\bar{1}$) which is -5, we can derive that 5 is (1)($\bar{1}$)(0)(1). Although 5 can be represented as (0)(1)(0)(1), however we are interested in the first vector to reach the definition of the complement of the SD number.

We can conclude from the previous discussion that the complement of the SD number is done by converting each 1 digit to $\bar{1}$ digit, each $\bar{1}$ digit to 1 digit and keep the 0 digit as it is, so we can reach the complement of this number and hence the subtraction operation as X-Y is the addition of X with the complement of Y.

If our system supports encoding 2 discussed in chapter 2, we can use table 4.1 to derive the following logic equations.

Table 4.1: Encoding conversion of the SD subtractor circuit.

A_1A_0	B_1B_0
00	00
01	11
11	01

Let A is the input and B is the complement of A.

$$B_0 = A_0. \tag{4.1}$$

$$B1 = A_0. (\sim A_1). \tag{4.2}$$

The above equations convert each signed digit into its complement form, if we have signed digit number and by applying the above equations we can generate the complement of this signed digit number and hence perform the subtraction operation as the addition with the complement of this number.

4.4 HSD subtraction

The HSD number system consists of a sequence of unsigned digits and signed digits. This forms a new number system since it is different from the two's complement number system and different from the SD number system.

To deduce the complement rules of the HSD number, we start with a small number consists of two unsigned digits and one signed digit as follows, and then generalize these rules to a larger number.

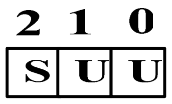

Figure 4.2: Three digit HSD number.

We refer to U as the unsigned digit and S as the signed digit. Each U digit allows 1 or 0, while each digit of S allows the following digit set $\{\bar{1}, 0, 1\}$.
The truth table of the valid values of this hybrid signed digits and their complement are illustrated in the following table.

Table 4.2: Three digit HSD number and its complement.

X	Complement of X
$(\bar{1})(0)(0)$	(0) (1)(0)(0)
$(\bar{1})(0)(1)$	(0) (0)(1)(1)
$(\bar{1})(1)(0)$	(0) (0)(1)(0)
$(\bar{1})(1)(1)$	(0) (0)(0)(1)
(0)(0)(0)	(0) (0)(0)(0)
(0)(0)(1)	(0) $(\bar{1})(1)(1)$
(0)(1)(0)	(0) $(\bar{1})(1)(0)$
(0)(1)(1)	(0) $(\bar{1})(0)(1)$
(1)(0)(0)	$(\bar{1})$ (1)(0)(0)
(1)(0)(1)	$(\bar{1})$ (0)(1)(1)
(1)(1)(0)	$(\bar{1})$ (0)(1)(0)
(1)(1)(1)	$(\bar{1})$ (0)(0)(1)

From the above table we can deduce the following:

1- The complement of the HSD number is divided into two groups, the first group is the unsigned digits and the second group is the signed digit.

2- The complement of the unsigned digits is the same as the two's complement of this number. This is done by inverting each digit and then adding 1 to the whole sequence.
3- The complement of the signed digit is deduced from the above table, this complement is an encoding conversion and it depends on the input signed digit and another input comes from the output of the two's complement logic of the unsigned digits.
4- As discussed in the previous chapter, the range covered by an n digits HSD representation depends on the numbers of the signed digits and their positions. For (d_{n-1}), the range is [-2^{n-1}, 2^n-1], for n=3 the range is [-4,7].
5- Suppose we have the number (1)(0)(1) decimal 5, the complement of this number is decimal -5 which can not be represented in three digits. However, we need four digits to make the correct complementation. Decimal -5 is ($\bar{1}$)(0)(1)(1) which is -8+3. So in this case we need to adjust the carry out of the signed digit to make the correct complementation. This carry out is either the carry in to another unsigned digit or the final carry out of this subtractor circuit. Also we need to make a sum adjustment in order to ensure correct results out of the subtractor circuit.

In the following section, we discuss the logic equations of the main building blocks of the HSD adder/subtractor circuit, and then the final block diagram of this design is illustrated.

4.4.1 SD encoding conversion

Table 4.3: Encoding conversion of the signed digit.

A_1A_0	W	B_1B_0
11	0	00
11	1	01
00	0	11
00	1	00
01	0	00
01	1	01

Let us assume that A is the input signed digit and B is the complement version of this signed digit. W is an input used to control the correctness of the signed digit conversion algorithm. We can deduce the following logic equations from table 4.3.

$$B_1 = (\sim W).(\sim A_0).(\sim A_1) \tag{4.3}$$

$$B_0 = B_1 + A_0W \tag{4.4}$$

4.4.2 SD carry out adjustment

Table 4.4: SD carry out adjustment.

co	co_modified
00	10
11	10
01	00
10	10

co_modified [0] = ‘0’. (4.5)

co_modified [1] = ~ (co [0]. (~co [1])). (4.6)

The final subtractor circuit of this three digit HSD number is illustrated in the following figure.

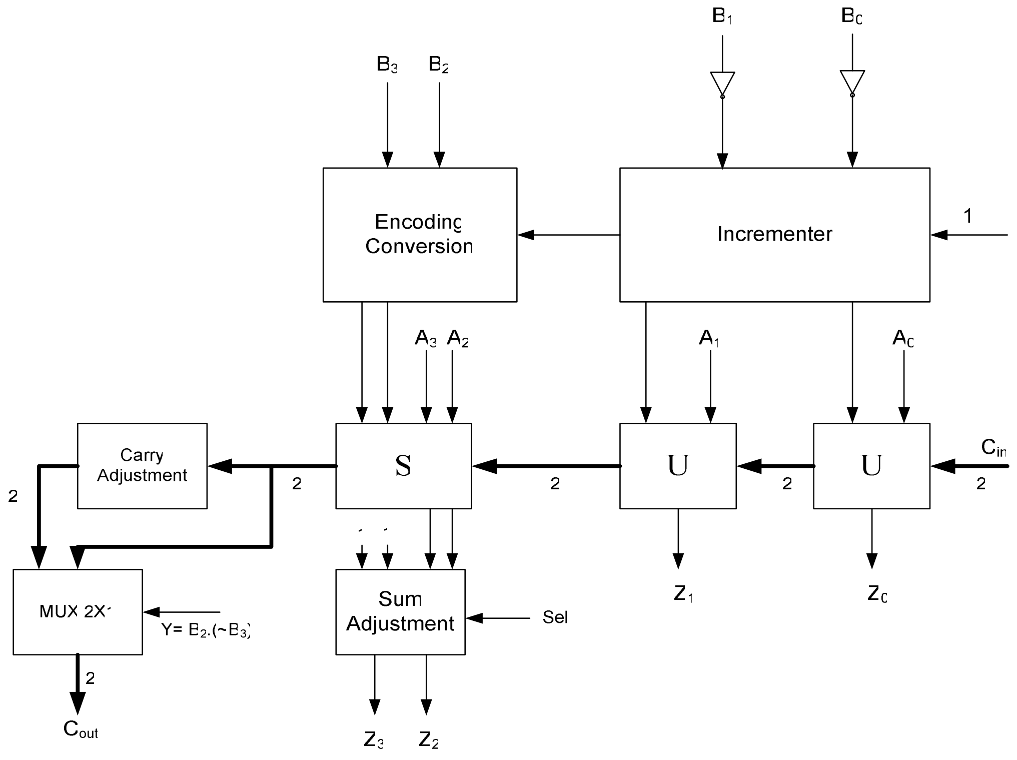

Figure 4.3: Three digit HSD subtractor circuit.

From the previous figure, we deduce that the HSD subtractor is implemented using the HSD adder circuit and additional modules to generate the complement of the HSD number and also ensure correct results in the carry and the sum of the signed digit.

HSD adder/subtractor circuit is the same as the above figure and additional multiplexer to choose between the normal input and the complemented one.

All our discussions are related to a three digit HSD number, however if we have another number such as thirty-two digit HSD number, the same rules applies and we need only to divide the whole HSD number into groups like the above HSD number we discussed and then apply the same rules.

In the following figure, n digit HSD subtractor is designed using 1 digit HSD subtractor circuit.

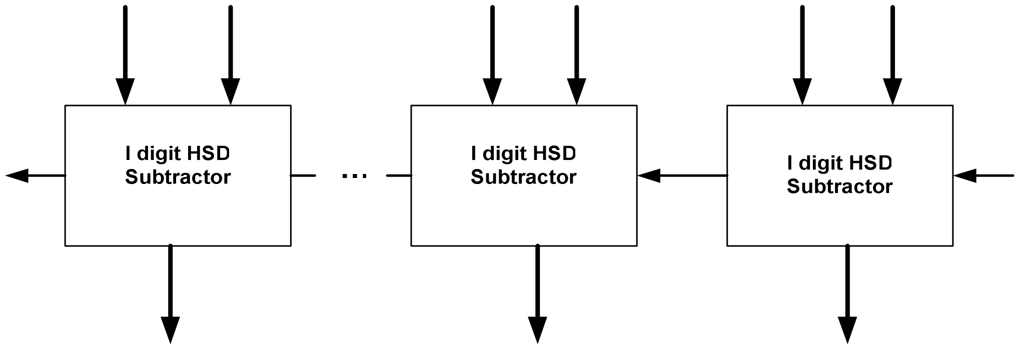

Figure 4.4: n digit HSD subtractor circuit.

4.5 HSD adder/subtractor (architecture 1) characterization

We use the same design flow used in section 3.6, a characterization of the HSD adder/subtractor is done by changing the distance parameter from 0 to 32 and analyzing its effect on the area, delay and ADP of the circuit.

In the following sections, we refer to the fast library as “FF”, refer to the typical library as “TT” and refer to the slow library as “SS”.

In the following section, the HSD adder/subtractor (architecture 1) characterization results are illustrated. The unit of the area is gate count, the unit of the delay is nano second (ns) and the unit of the ADP is (gate count * ns).

4.6 HSD adder/subtractor (architecture 1) characterization results

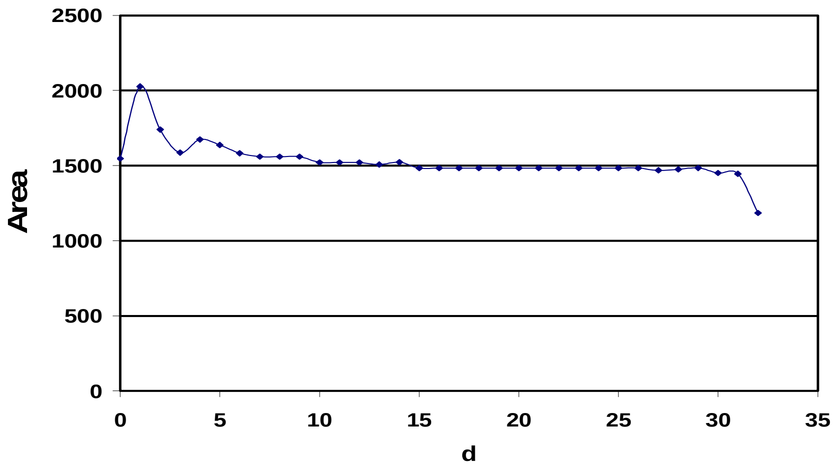

Figure 4.5: Area of architecture 1 (FF).

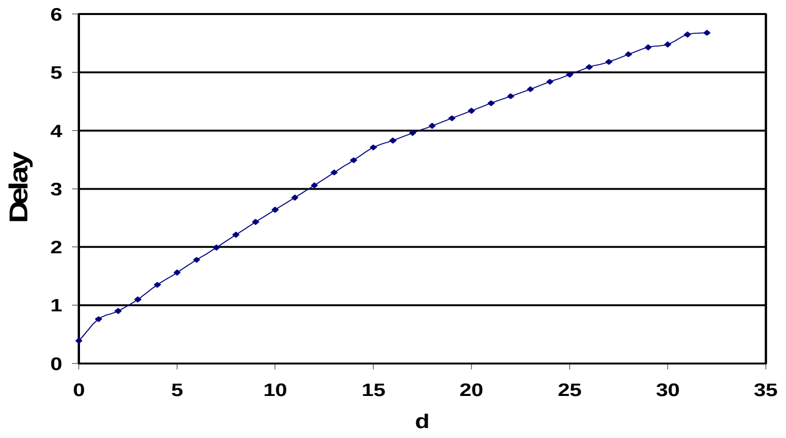

Figure 4.6: Delay of architecture 1 (FF).

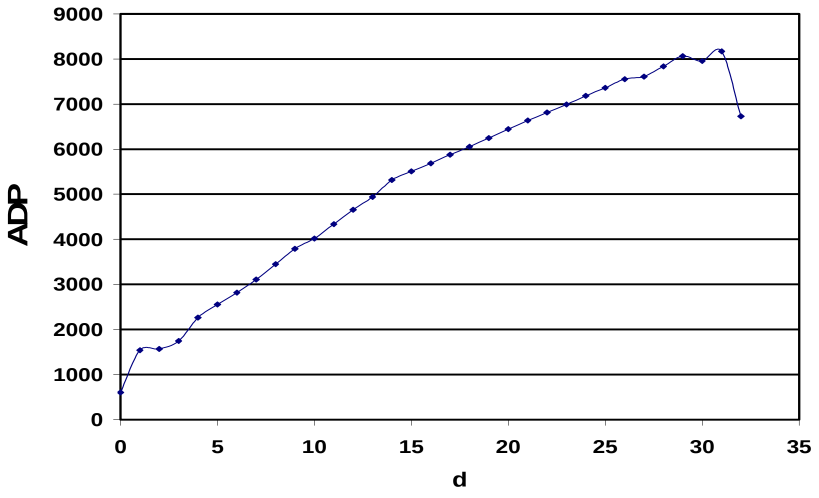

Figure 4.7: ADP of architecture 1 (FF).

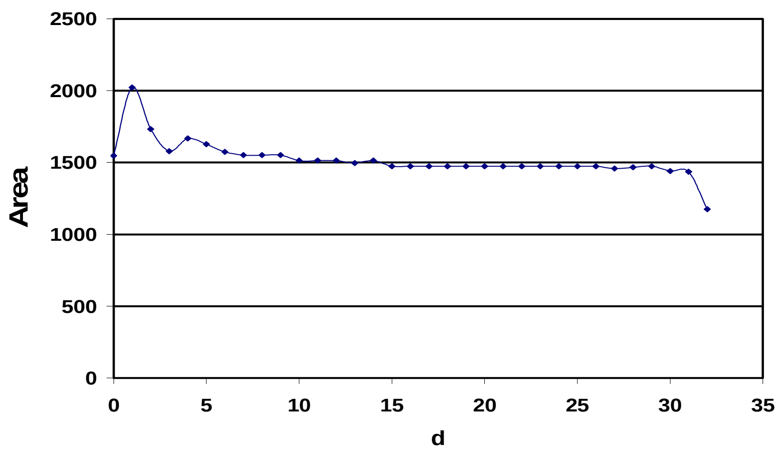

Figure 4.8: Area of architecture 1 (TT).

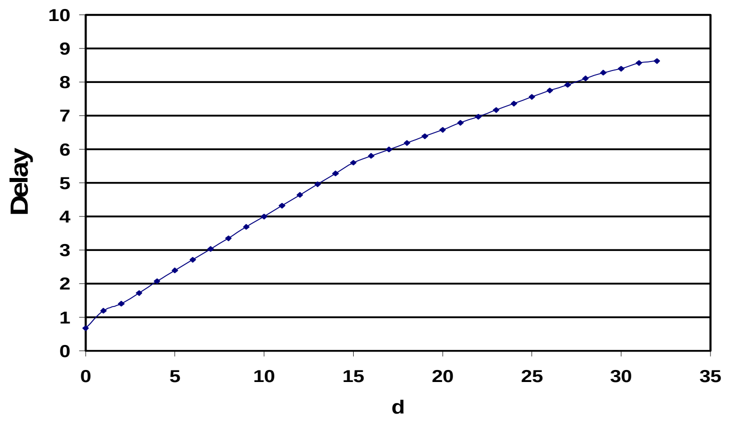

Figure 4.9: Delay of architecture 1 (TT).

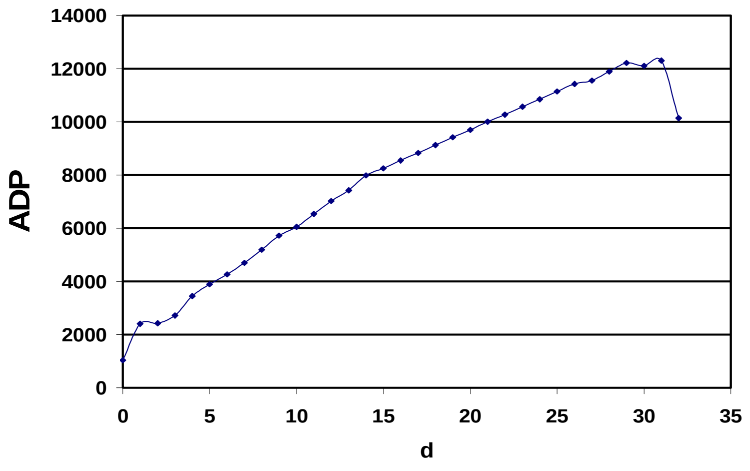

Figure 4.10: ADP of architecture 1 (TT).

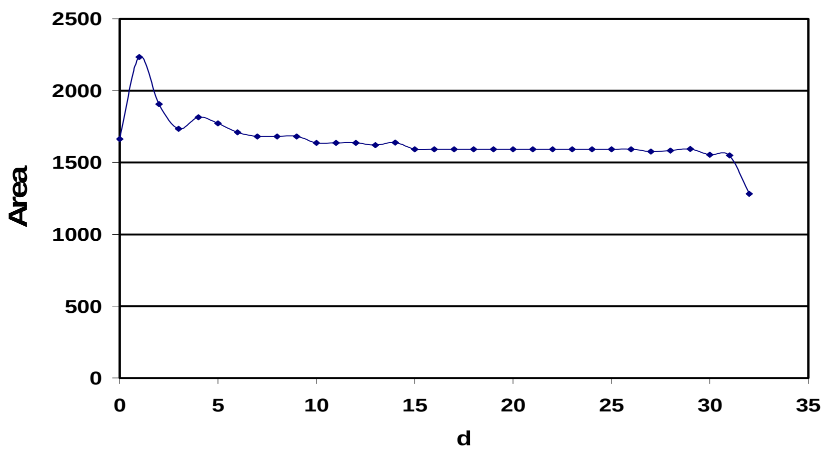

Figure 4.11: Area of architecture 1 (SS).

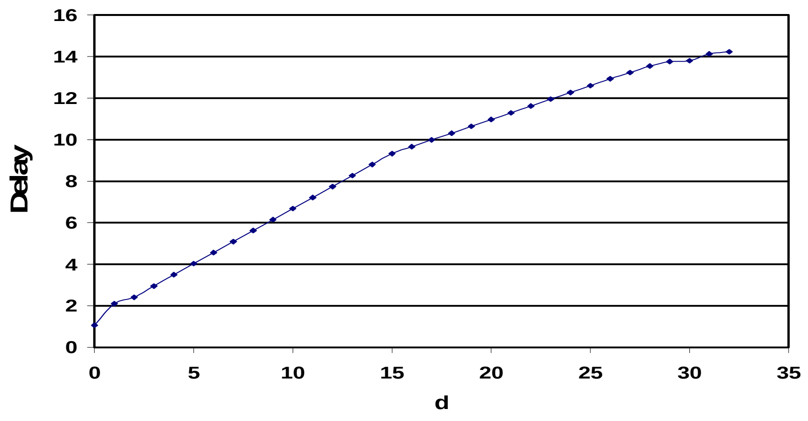

Figure 4.12: Delay of architecture 1 (SS).

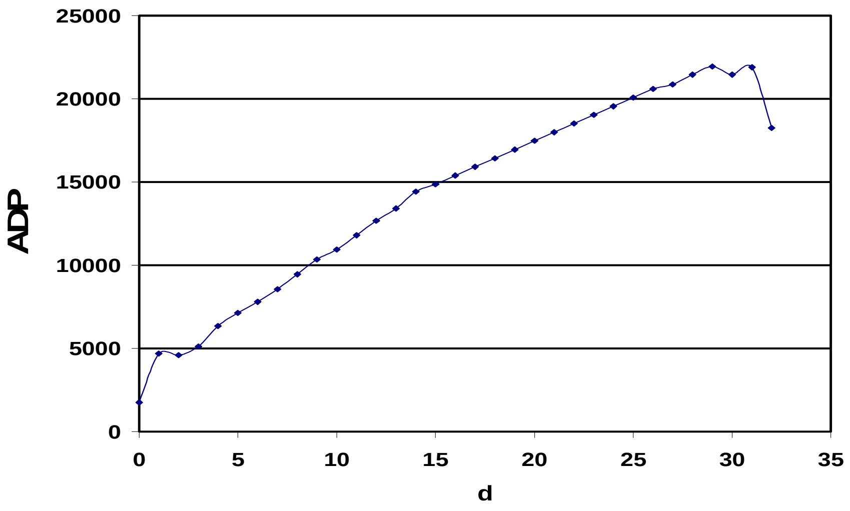

Figure 4.13: ADP of architecture 1 (SS).

4.7 Analysis of architecture 1 characterization results

In our analysis, we use the formula $\frac{value_{\max} - value_{\min}}{value_{\max}}$ to calculate the percentage of decrease in each metric.

Table 4.5: Analysis of architecture 1 characterization results.

Metric	Percentage of decrease formula	Percentage of decrease (FF)	Percentage of decrease (TT)	Percentage of decrease (SS)
Area	$\frac{d_1 - d_{32}}{d_1}$	41.5%	41.8%	42.6%
Delay	$\frac{d_{32} - d_0}{d_{32}}$	93.1%	92.2%	92.5%
ADP	$\frac{d_{31} - d_0}{d_{31}}$	92.6%	91.5%	91.9%

We conclude the following from the former table and the former figures:

1. Area characterization
    - As the number of signed digits decreases in the architecture, it is possible to decrease the area in most cases.
    - We notice that the area of d_1 is larger than the area of d_0, although the number of signed digits is 64 at d_0 and 48 at d_1. d_0 is the fully signed digit system and the subtractor circuit is simply an encoding conversion circuit which discussed in section 4.3, however at d_1 the subtractor circuit is the one based on figure 4.3, so this subtractor circuit is larger than the subtractor circuit used for d_0. Hence the area of d_1 is larger than the area of d_0.
2. Delay characterization.
    - As the number of signed digits increases in the architecture, it is possible to decrease the delay.

- We notice that at d_{16} the delay curve has a change; this is due to the change of the number of signed digits as the distance changed. The number of signed digits is settled to be 2 at $d_{16.}$

3. ADP characterization
   - As the number of signed digits increases in the architecture, it is possible to decrease the ADP, as a special case is the range from (d_{29} to d_{30}) and from (d_{31} to d_{32}), which has a dominant area over the delay.
   - The ADP curve is almost similar to the delay curve which means that the decrease in the delay is dominant over the increase in the area.(area is dominant only from (d_{29} to d_{30}) and from (d_{31} to d_{32})).

Table 4.6: Architecture 1 (d_0) comparison with CLA adder/subtractor.

Metric	Comparison formula	(FF)	(TT)	(SS)
Area	$\frac{d_0}{CLA}$	1.63	1.66	1.67
Delay	$\frac{CLA}{d_0}$	5.8	5.7	6.3
ADP	$\frac{CLA}{d_0}$	3.5	3.4	3.7

We can conclude that, the delay of the CLA adder/subtractor is larger than the delay of architecture 1 (d_0) as well as the ADP. However, the area of architecture 1 (d_0) is larger than the area of the CLA adder/subtractor.

4.8 HSD adder/subtractor (architecture 2)

HSD subtractor architecture 2 is the same as architecture 1 except that we implement the circuit which generates the two's complement of operand B using the CLA adder. The architecture of three digit HSD subtractor circuit is shown in figure 4.14.

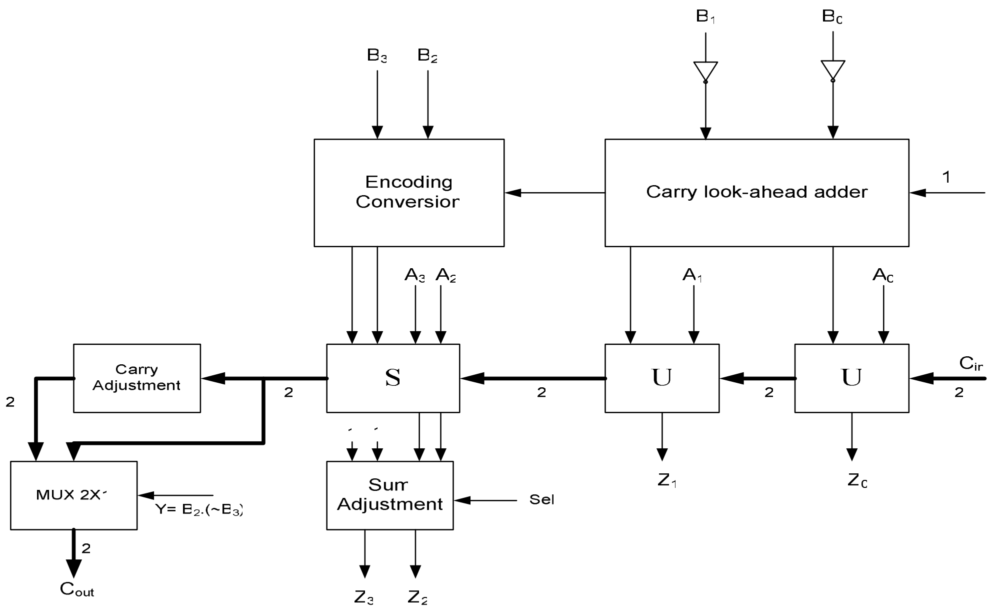

Figure 4.14: Three digit HSD subtractor circuit (architecture 2).

From the above figure, we deduce that the HSD Subtractor is implemented using the HSD adder circuit and additional modules to generate the complement of the HSD number and also ensure correct results in the carry and the sum of the signed digit.

The HSD adder/subtractor circuit is the same as the above figure and additional multiplexer to choose between the normal input and the complemented one.

Figure 4.14 shows the architecture of a three digit HSD number, however if we have another number as thirty-two digit HSD number, the same rules applies and we need only to divide the whole HSD number into groups like the above HSD number we discussed and then apply the same rules.

4.9 HSD adder/subtractor (architecture 2) characterization

We use the same design flow used in section 3.6; however we do the synthesis using library “ff_1v26_m40c” only. We do the characterization for the following distances d_0, d_4, d_9, d_{14}, d_{20}, d_{26} and d_{32} only.

In the following section, we refer to the fast library as “FF”, also in this section the HSD adder/subtractor (architecture 2) characterization results are illustrated. The unit of the area is gate count, the unit of the delay is nano second (ns) and the unit of the ADP is (gate count * ns).

4.10 HSD adder/subtractor (architecture 2) characterization results

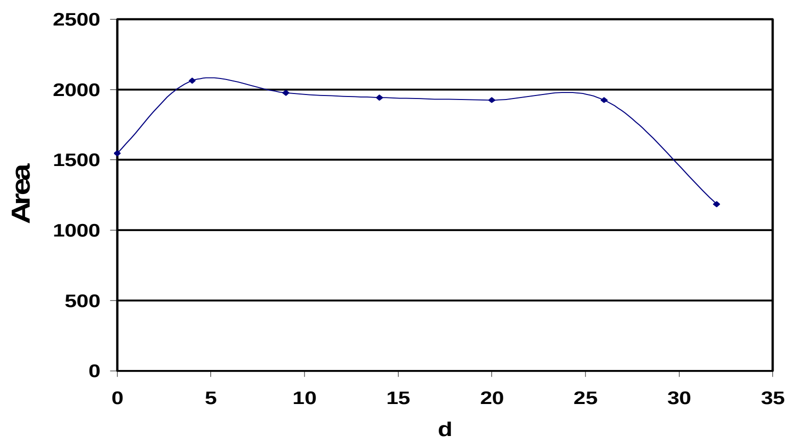

Figure 4.15: Area of architecture 2 (FF).

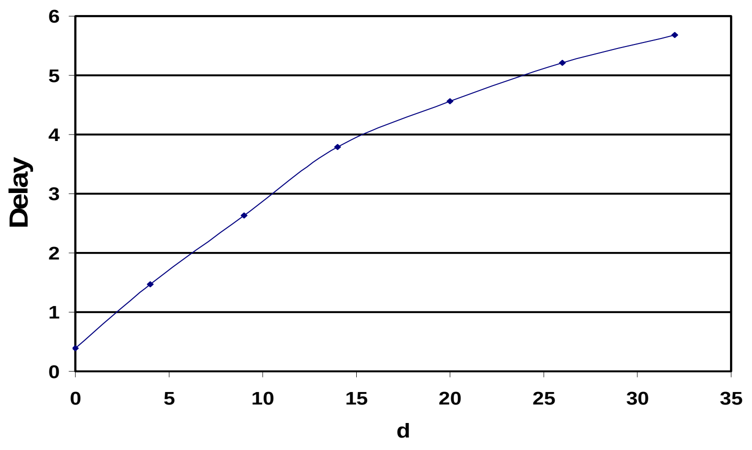

Figure 4.16: Delay of architecture 2 (FF).

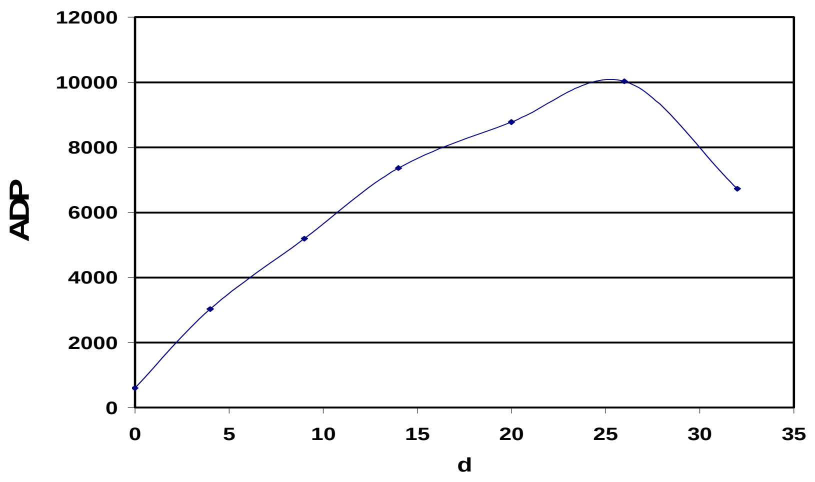

Figure 4.17: ADP of architecture 2 (FF).

4.11 Analysis of architecture 2 characterization results

In our analysis, we use the formula $\frac{value_{\max} - value_{\min}}{value_{\max}}$ to calculate the percentage of decrease in each metric.

Table 4.7: Analysis of architecture 2 characterization results.

Metric	Percentage of decrease formula	Percentage of decrease (FF)
Area	$\frac{d_4 - d_{32}}{d_4}$	42.5%
Delay	$\frac{d_{32} - d_0}{d_{32}}$	93.1%
ADP	$\frac{d_{26} - d_0}{d_{26}}$	93.9%

We conclude the following from the former table and the former figures:

1. Area characterization
    - As the number of signed digits decreases in the architecture, it is possible to decrease the area in most cases.
    - We notice that the area of d_4 is larger than the area of d_0, as the subtractor circuit used for d_4 is larger than the subtractor circuit used for d_0.
2. Delay characterization.
    - As the number of signed digits increases in the architecture, it is possible to decrease the delay.
3. ADP characterization
    - As the number of signed digits increases in the architecture, it is possible to decrease the ADP; the range from (d_{26} to d_{32}) is a special case which has a dominant area over the delay.

- The ADP curve is almost similar to the delay curve which means that the decrease in the delay is dominant over the increase in the area.(area is only dominant from (d_{26} to d_{32})).

4. Comparison with the CLA adder/subtractor.
    - The CLA adder/subtractor delay is 5.8 times the delay of architecture 2 (d_0), the CLA adder/subtractor ADP is 3.5 times the ADP of architecture 2 (d_0), while the area of architecture 2 (d_0) is 1.63 times the area of the CLA adder/subtractor.
5. Comparison with architecture 1.
    - Comparing these results with the results obtained from architecture 1. The delay, area and ADP results of architecture 2 are larger than the results obtained from architecture 1.

4.12 HSD adder/subtractor (architecture 3)

All discussed architectures till now whether the HSD adder or the HSD adder/subtractor apply the following concepts: they consist of a mixture of redundant (signed) digits and non-redundant (unsigned) digits, the non-redundant digits allow the following digit set {0, 1}, however the redundant digits allow the following digit set $\{\bar{1}, 0, 1\}$ where $\bar{1} = -1$.

Jaberipur et al [1,7] use another concept in implementing the hybrid redundant system. They define posibits as digits allow {0, 1} and negabits as digits allow {-1, 0}. Each redundant digit is expressed as a mixture of a posibit and a negabit. Also in their proposal they allow the non-redundant positions to have a negabit as well as have a posibit. They use it in an inverted encoding format $\{\bar{1}, 0\} \rightarrow \{0, 1\}$.

This allow them to use the conventional FA and HA in the circuit implementing the addition algorithm. Also, they propose a modified FA circuit based on multiplexers and inverters for improving the speed of the circuit [1, 3].

They implement hybrid signed digit adder circuit capable of adding two numbers which use the previous concept. Also, their system is symmetric which allows minor modification to perform the subtraction. They negate the subtrahend by bitwise inversion of each digit and then perform the addition [1]. Full details of their suggestions and implementation are found in [1, 3].

The following figure shows their implementation of the full adder circuit. It adds three bits x, y and c_{in} and produces the output sum s and output carry $c_{out.}$

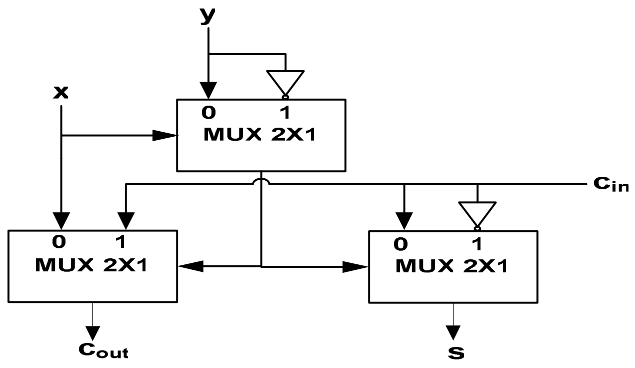

Figure 4.18: Full adder cell built of three muxes.

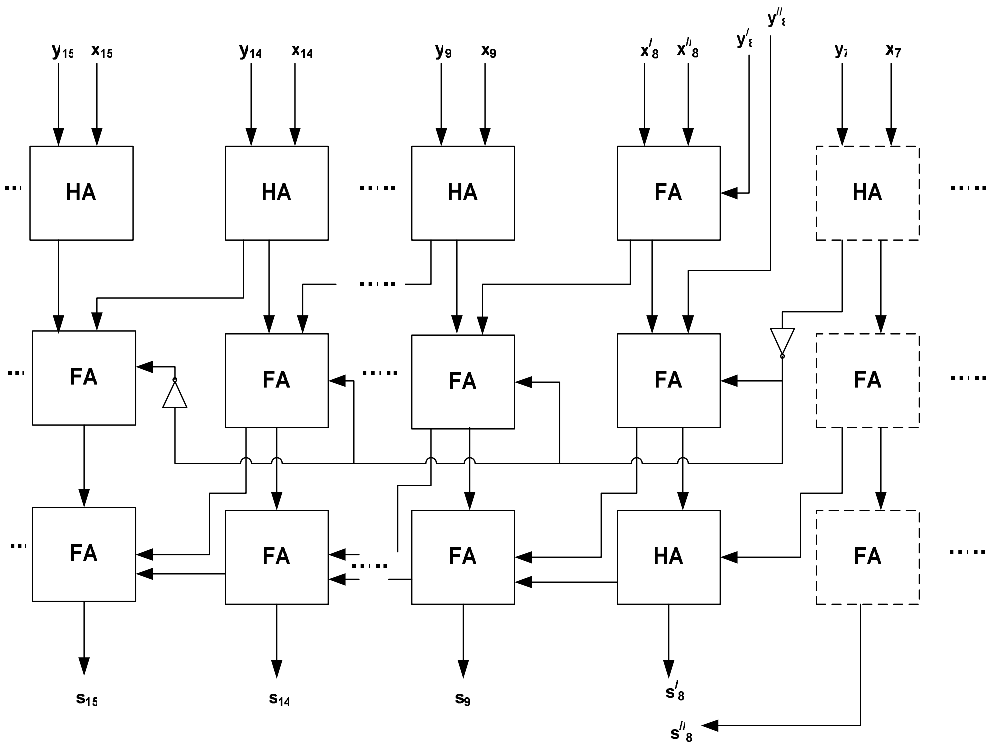

Figure 4.19: HSD adder (architecture 3) circuit d_8.

In the above figure, example of their hybrid signed digit adder with d_8 is illustrated. The signed digit has two bits for each digit; prime variable denotes the posibit, while double prime variable denotes the negabit.

The HSD adder/subtractor circuit is the same as the above figure and additional modules to choose between the normal input and the complemented one and also to do the inversion.

4.13 HSD adder/subtractor (architecture 3) characterization

We use the same design flow used in section 3.6; however we do the synthesis using library "ff_1v26_m40c" only. We do the characterization for the following distances d_3, d_7, d_{14}, d_{20} and d_{26} only.

In the following section, we refer to the fast library as "FF", also in this section the HSD adder/subtractor (architecture 3) characterization results are illustrated. The unit of the area is gate count, the unit of the delay is nano second (ns) and the unit of the ADP is (gate count * ns).

4.14 HSD adder/subtractor (architecture 3) characterization results

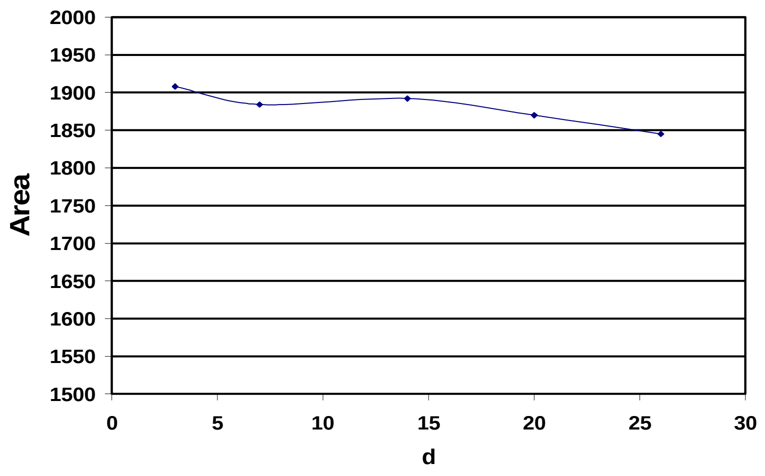

Figure 4.20: Area of architecture 3 (FF).

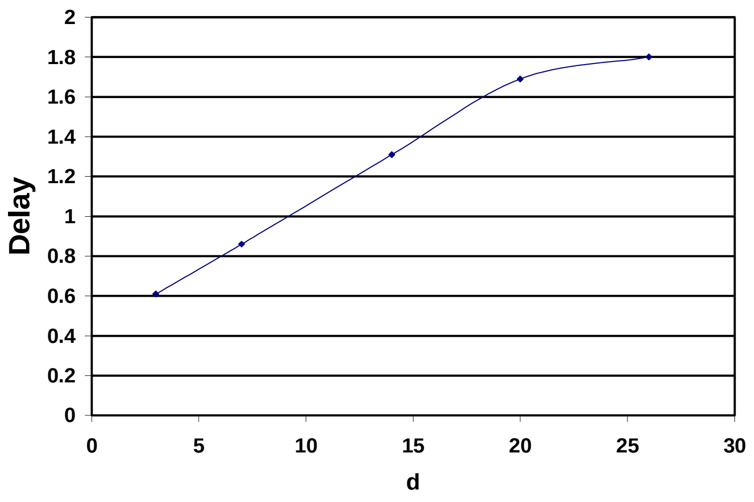

Figure 4.21: Delay of architecture 3 (FF).

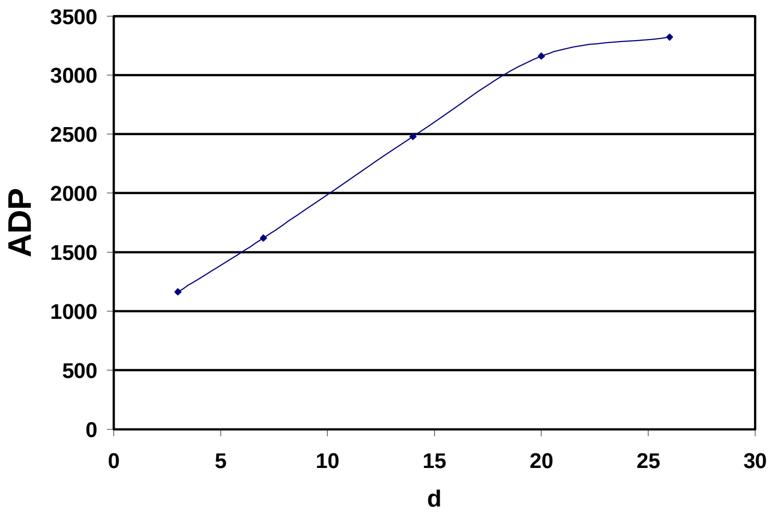

Figure 4.22: ADP of architecture 3 (FF).

4.15 Analysis of architecture 3 characterization results

In our analysis, we use the formula $\frac{value_{\max} - value_{\min}}{value_{\max}}$ to calculate the percentage of decrease in each metric.

Table 4.8: Analysis of architecture 3 characterization results.

Metric	Percentage of decrease formula	Percentage of decrease (FF)
Area	$\frac{d_3 - d_{26}}{d_3}$	3.3%
Delay	$\frac{d_{26} - d_3}{d_{26}}$	66.1%
ADP	$\frac{d_{26} - d_3}{d_{26}}$	64.9%

We conclude the following from the former table and the former figures:

1. Area characterization
    - In this architecture, the area per redundant or non-redundant digit is almost the same; hence the change in the area as the signed digits changes has a minor effect.
2. Delay characterization
    - As the number of signed digits increases in the architecture, it is possible to decrease the delay.
3. ADP characterization.
    - As the number of signed digits increases in the architecture, it is possible to decrease the ADP.
    - The ADP curve is similar to the delay curve which means that the decrease in the delay is dominant over the change in the area.

4. Comparison with the CLA adder/subtractor.
    - The CLA adder/subtractor delay is 3.7 times the delay of architecture 3 (d_3), the CLA adder/subtractor ADP is 1.8 times the ADP of architecture 3 (d_3), while the area of architecture 3 (d_3) is 2 times the area of the CLA adder/subtractor.
5. Comparison with architecture1.
    - Comparing these results with the results obtained from architecture 1. The delay and the ADP results of architecture 3 are lower than the delay and the ADP results of architecture 1, while the area results of architecture 3 is larger than the area results of architecture 1.

4.16 Summary

In this chapter, the basic concept of HSD subtraction has been analyzed. A HSD adder/subtractor and a modified version of this circuit have been proposed. Furthermore, a third HSD adder/subtractor architecture has been analyzed.

The characterization of the three architectures has been done by changing the distance (d) parameter. It was found that the most redundant design is the optimum with respect to the delay and the ADP; however the non-redundant architecture is the optimum with respect to the area of the circuit.

Chapter 5

5- Arithmetic unit

5.1 Introduction

In this chapter, we propose an arithmetic unit (AU) consisting of HSD adder/subtractor discussed in the previous chapter and register file to store the result of the HSD adder/subtractor. Data stored in the register file support the hybrid signed digit format.

We characterize this arithmetic unit versus the distance parameter in terms of its delay, area and ADP; also we discuss the conversion from and to the conventional number system.

5.2 Arithmetic unit architecture

We propose an arithmetic unit consisting of HSD adder/subtractor and register file to store the result of the HSD adder/subtractor, the main ideas in our proposal are as follows:

1- The HSD adder/subtractor is the main arithmetic unit; it is capable of doing the addition and the subtraction between two HSD numbers. We use architecture 1 in the previous chapter as our HSD adder/subtractor.

2- Three port register file is used to store the result of the HSD adder/subtractor, the storage supports the hybrid signed digit format, this is the main idea in our proposal such that we do all the operations within our AU in the hybrid signed digit format and then convert to/from the conventional system when needed only, for example: when we need to store the results in the memory of the system which supports the conventional system.

Our Arithmetic unit consists of the following modules:

1- The HSD adder/subtractor.
2- Three port register file. One write port and two read ports.
3- Multiplexer to choose between the output of the HSD adder/subtractor and the external data input feeding the write port of the register file.

Figure 5.1 shows the architecture of the arithmetic unit.

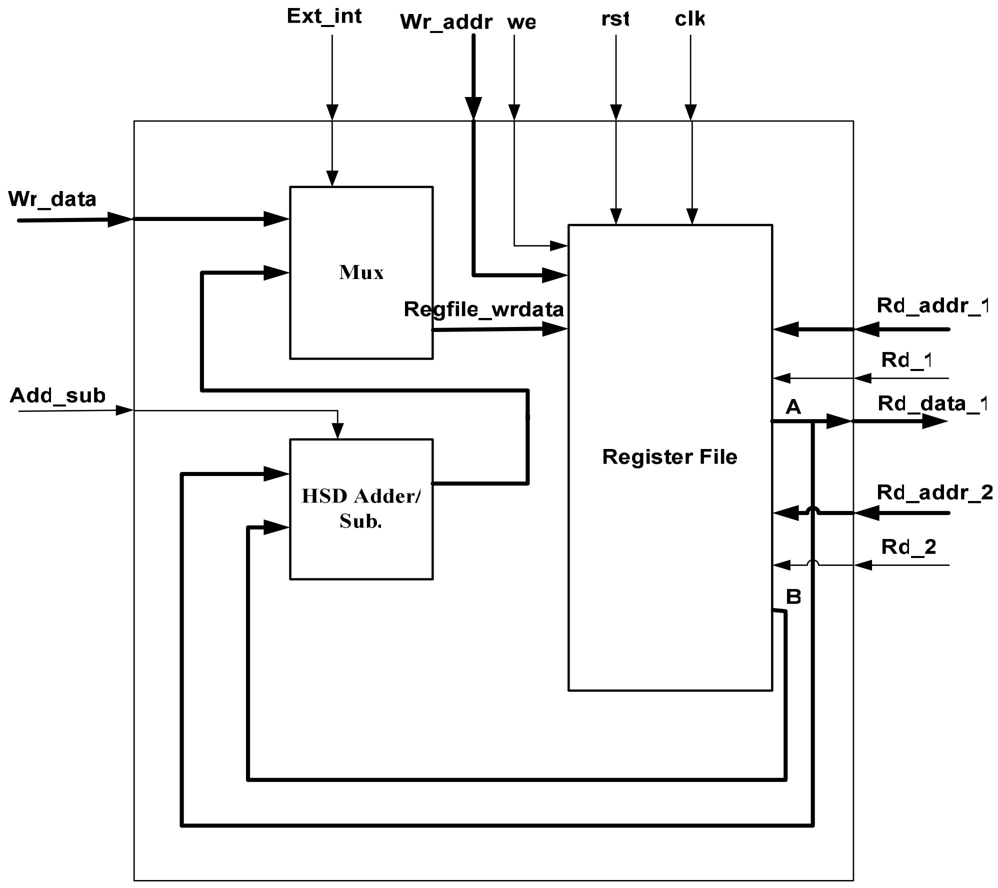

Figure 5.1: Arithmetic unit architecture.

The operation of the arithmetic unit is described as follows:

1. The HSD adder/subtractor adds or subtracts based on the **add_sub** signal: 0 means addition while 1 means subtraction. Inputs to the HSD adder/subtractor are **A** and **B**, both of them are data stored in the register file.
2. The register file has two read ports; each port has address, data and control signals. The output data from the two read ports feed the inputs of the HSD adder/subtractor. We use only the first port to read the contents of the register file externally, so its data port is connected to the output data port of the arithmetic unit.
3. The register file has one write port, which has address, data and control signals. The input data may be the output result of the HSD adder/subtractor or external data to be written into the register file.
4. A multiplexer selects the input write data to the register file based on the **ext_int** signal, 0 means the output result of the HSD adder/subtractor while 1 means the external data.
5. The data width of the register file is changed as the distance d changes in the characterization, for example: data width = 64 for d_0, data width = 32 for d_{32}.
6. The length of the register file is 32.

5.3 AU characterization

We use the same design flow used in section 3.6 except step 1; a characterization of the AU is done by changing the distance parameter from 0 to 32 and analyzing its effect on the area, delay and ADP of the circuit.

In the following sections, we refer to the fast library as “FF”, refer to the typical library as “TT” and refer to the slow library as “SS”.

In the following section, the AU characterization results are illustrated. The unit of the area is gate count, the unit of the delay is nano second (ns) and the unit of the ADP is (gate count * ns).

5.4 AU characterization results

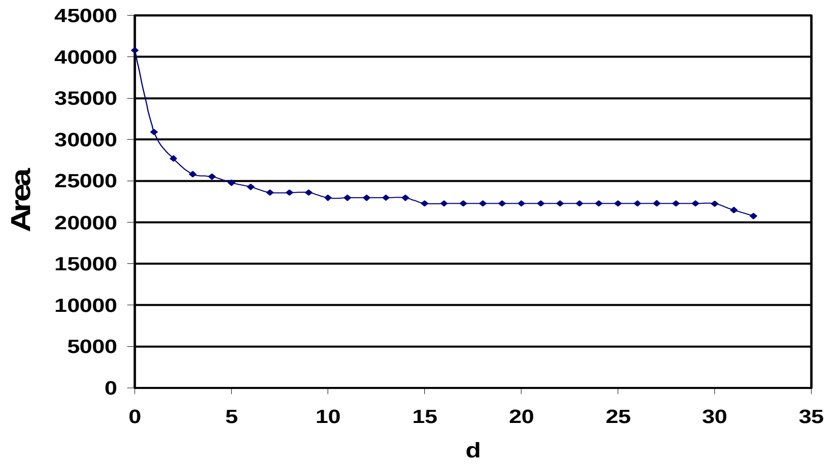

Figure 5.2: Area of the AU (FF).

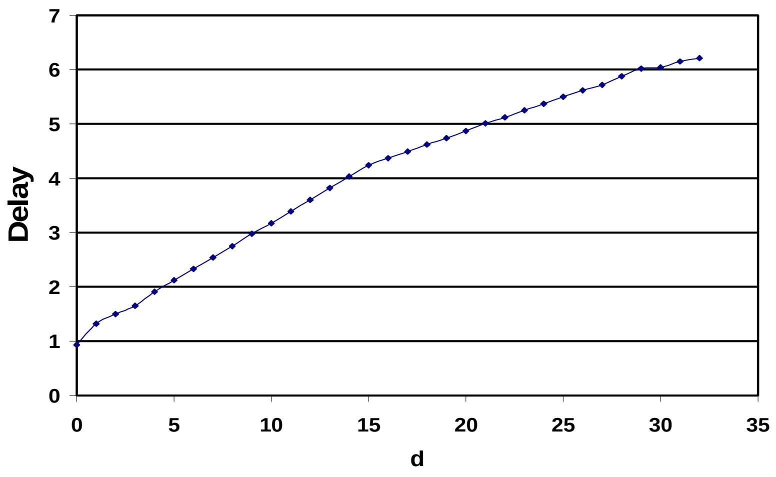

Figure 5.3: Delay of the AU (FF).

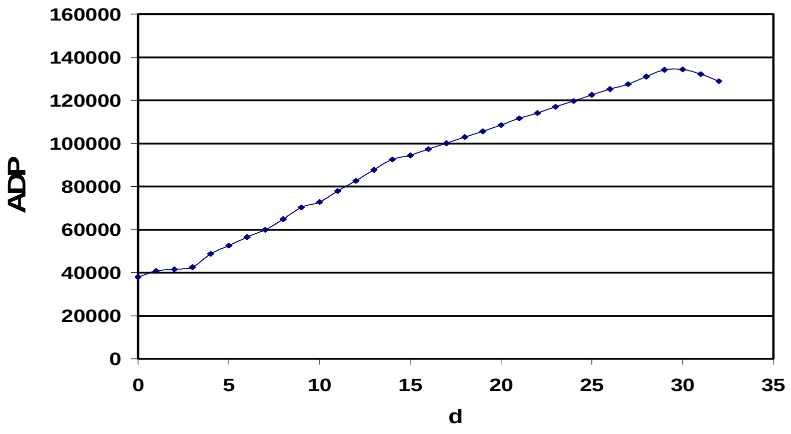

Figure 5.4: ADP of the AU (FF).

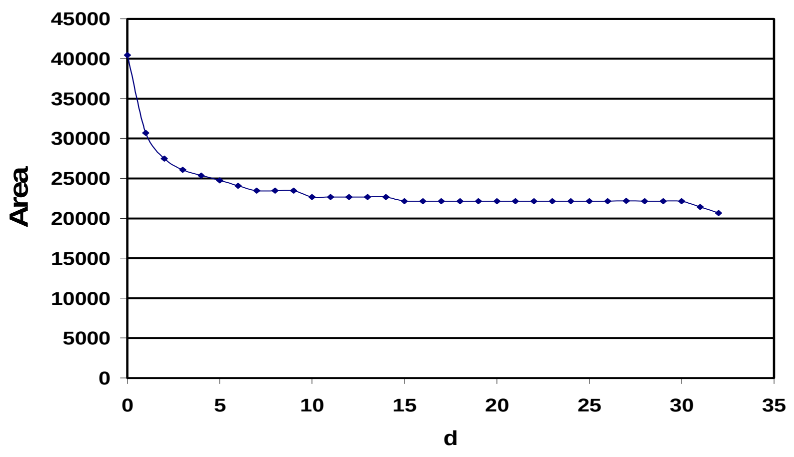

Figure 5.5: Area of the AU (TT).

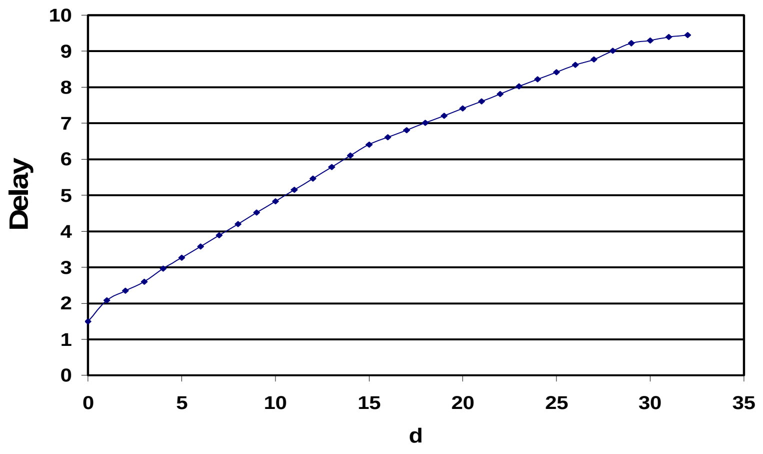

Figure 5.6: Delay of the AU (TT).

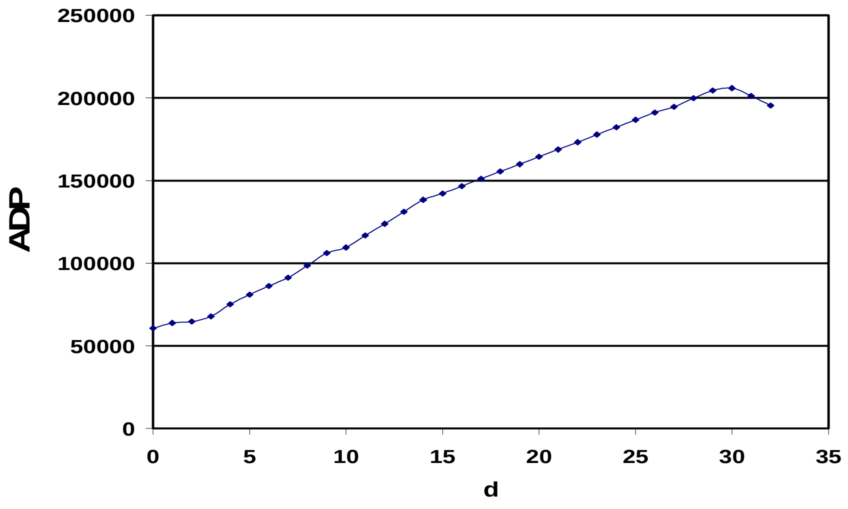

Figure 5.7: ADP of the AU (TT).

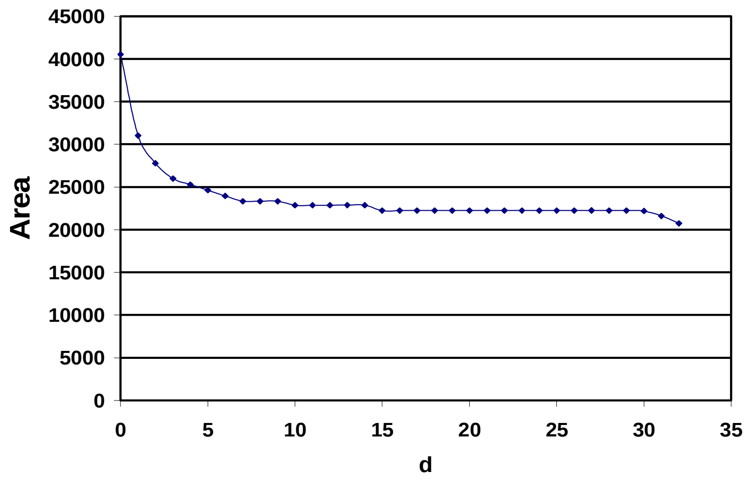

Figure 5.8: Area of the AU (SS).

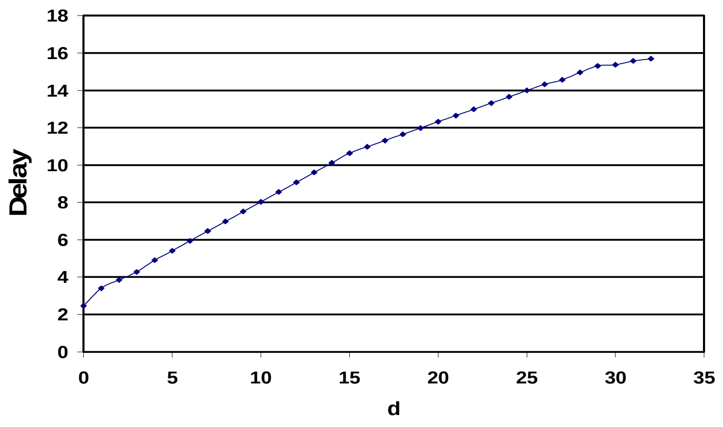

Figure 5.9: Delay of the AU (SS).

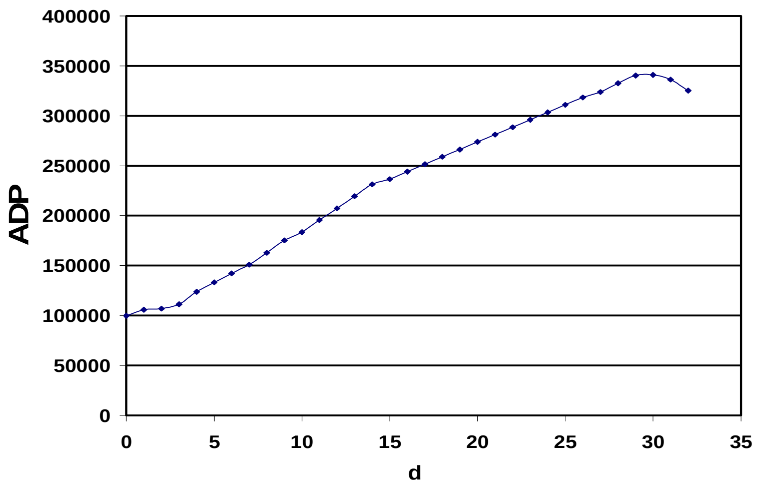

Figure 5.10: ADP of the AU (SS).

5.5 Analysis of the AU characterization results

In our analysis, we use the formula $\frac{value_{\max} - value_{\min}}{value_{\max}}$ to calculate the percentage of decrease in each metric.

Table 5.1: Analysis of the AU characterization results.

Metric	Percentage of decrease formula	Percentage of decrease (FF)	Percentage of decrease (TT)	Percentage of decrease (SS)
Area	$\frac{d_0 - d_{32}}{d_0}$	49%	48.9%	48.8%
Delay	$\frac{d_{32} - d_0}{d_{32}}$	85%	84.1%	84.3%
ADP	$\frac{d_{30} - d_0}{d_{30}}$	71.7%	70.5%	70.7%

We conclude the following from the former table and the former figures:

1. Area characterization
   - As the number of signed digits decreases in the architecture, it is possible to decrease the area.
2. Delay characterization.
   - As the number of signed digits increases in the architecture, it is possible to decrease the delay.
3. ADP characterization.
   - As the number of signed digits increases in the architecture, it is possible to decrease the ADP, as a special case is the range from d_{30} to d_{32}, which has a dominant area over the delay.
   - The ADP curve is almost similar to the delay curve which means that the decrease in the delay is dominant over the increase in the area.(area is dominant only in the range from d_{30} to d_{32})

Table 5.2: Our AU comparison with the CLA AU.

Metric	Comparison formula	(FF)	(TT)	(SS)
Area	$\frac{d_0}{CLA}$	1.9	1.9	1.9
Delay	$\frac{CLA}{d_0}$	3	3.1	3.2
ADP	$\frac{CLA}{d_0}$	1.5	1.5	1.6

We can conclude that, the delay of the CLA AU is larger than the delay of our AU (d_0) as well as the ADP. However, the area of our AU (d_0) is larger than the area of the CLA AU.

5.6 AU with conversion

5.6.1 Input conversion

In this section, we discuss the input conversion algorithm from the conventional number system to the HSD number system.
Our number system use the encoding $X = x^l - 2x^h$.

First, we analyze the case of d_0, which is the SD system. Input vector from the conventional number system consists of 32 digits allows only {0,1}. We refer to the conventional number system digit as U. The following figure shows a 32 digit conventional number.

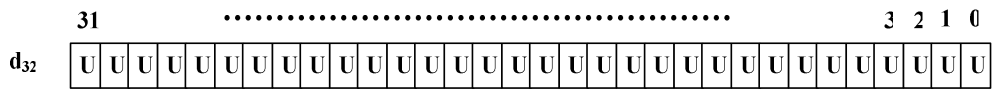
31
3 2 1 0
d_{32}
U U U U U U U U U U U U U U U U U U U U U U U U U U U U U U U U

Figure 5.11: 32 digit conventional number.

Input vector to the SD system consists of digits that allows {$\bar{1}$,0,1}, we refer to this digit as S. The following figure shows a 32 digit SD system.

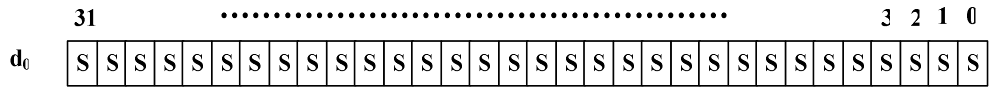
31
3 2 1 0
d_0
S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S S

Figure 5.12: 32 digit SD number (HSD d_0).

To do the conversion, we group each two digits “UU” in the conventional system and convert them into the equivalent two signed digits “SS”. The truth table of the conversion is shown in the following table.

Table 5.3: Input conversion rules (1).

Decimal	U U	SS	Encoding
0	00	00	0000
1	01	$1\bar{1}$	0111
2	10	10	0100
3	11	11	0101

In the second line of the table (decimal 1), the output signed digits can be (0)(1) or (1)($\bar{1}$). We choose the second choice to have $\bar{1}$ as input to our system.

Second, we analyze the case of HSD d_1. The conventional number system input vectors consist of U digits only; while the hybrid signed number system vectors consist of a mixture of U and S digits.

The following figure shows a 32 digit HSD number (d_1)

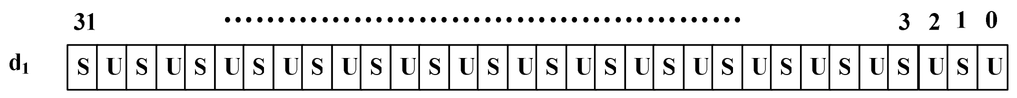
31
3 2 1 0
d_1
S U S U S U S U S U S U S U S U S U S U S U S U S U S U S U S U

Figure 5.13: 32 digit HSD number (d_1).

The conversion is done as follows:

1- Digit 0 is the same in the two vectors as a U digit, so it will be a connection only without any conversion.

2- We group digits 1 and 2 in the conventional number "UU" and convert them into digits 1 and 2 in the hybrid signed digit number "US" according to the following truth table.

Table 5.4: Input conversion rules (2).

Decimal	U U	US	Encoding
0	00	00	000
1	01	$1\bar{1}$	111
2	10	10	100
3	11	11	101

3- Continue grouping "UU" digits in the first vector and convert them into "US" digits in the second vector according to the previous table.

4- Digit 31 is converted from U to S according to the following table.

Table 5.5: Input conversion rules (3).

Decimal	U	S	Encoding
0	0	0	00
1	1	1	01

Finally, by analyzing the case of d_0 and d_1, we discussed all different situations to make the conversion from the conventional number system to the HSD number system. We use the above truth tables and rules to convert any number with different d parameter from the conventional number system to the hybrid signed digit number system.

5.6.2 Output conversion

We need to convert back from the HSD number system to the conventional number system. The encoding of the SD satisfies the following relation:

$$X = x^l - 2x^h$$

The main algorithm for conversion is to shift the sequence $(x^h_{n-1},\dots\dots x^h_0)$ to the left one digit , then subtract this sequence from the sequence $(x^l_{n-1}, \dots\dots\dots\dots x^l_0)$ using two's complement arithmetic.

5.7 AU with conversion characterization

In this section, conversion from the HSD number system to the conventional number system is done and the reverse. In real systems, all operations are done within the AU and the conversion is done only when we need to access the memory of the system which supports the conventional number system. As a consequence, the conversion is not done per addition or subtraction operation.

To model this behavior, we design a system consisting of six cascaded HSD adder/subtractor (architecture 1), then apply the input conversion before the first stage and apply the output conversion after the final stage. In this case, we assume that the probability of the conversion occurs after six times of addition or subtraction operations.

We use the same design flow used in section 3.6 except step 1; a characterization of the AU with conversion is done by changing the distance parameter from 0 to 32 and analyzing its effect on the area, delay and ADP of the circuit.

We refer to the fast library as “FF”, refer to the typical library as “TT” and refer to the slow library as “SS”.

In this section, the AU with conversion characterization results are illustrated. The unit of the area is gate count, the unit of the delay is nano second (ns) and the unit of the ADP is (gate count * ns).

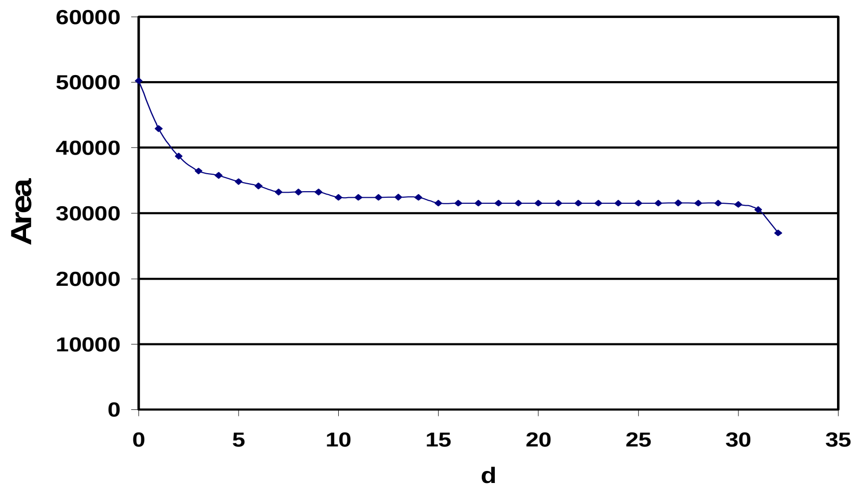

Figure 5.14: Area of the AU with conversion (FF).

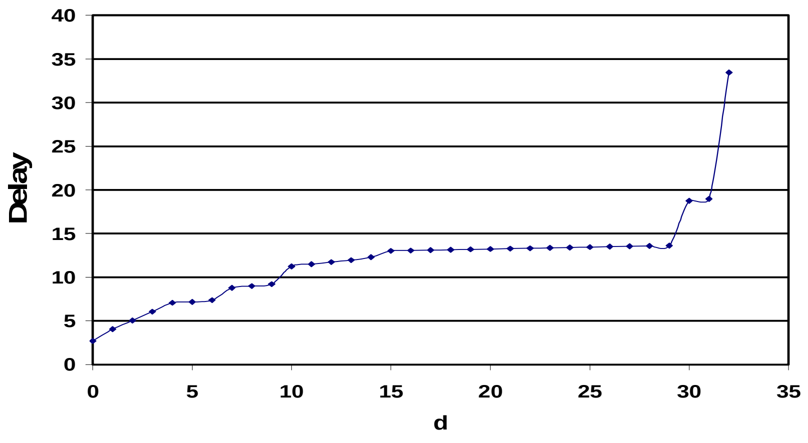

Figure 5.15: Delay of the AU with conversion (FF).

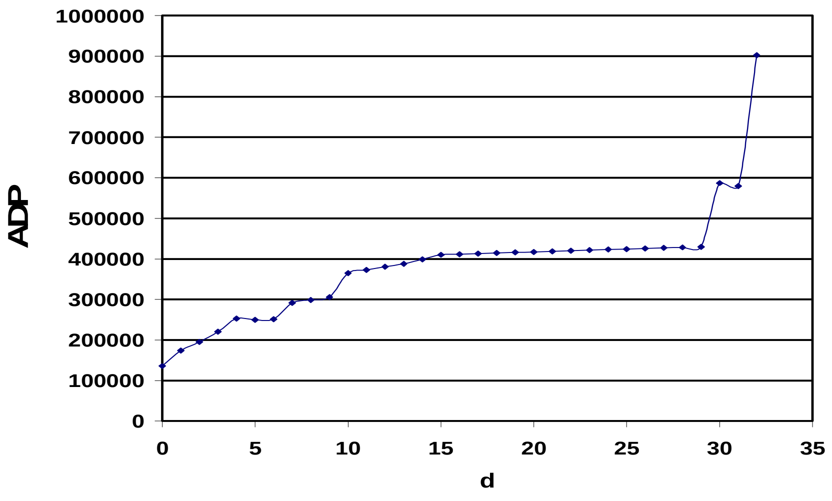

Figure 5.16: ADP of the AU with conversion (FF).

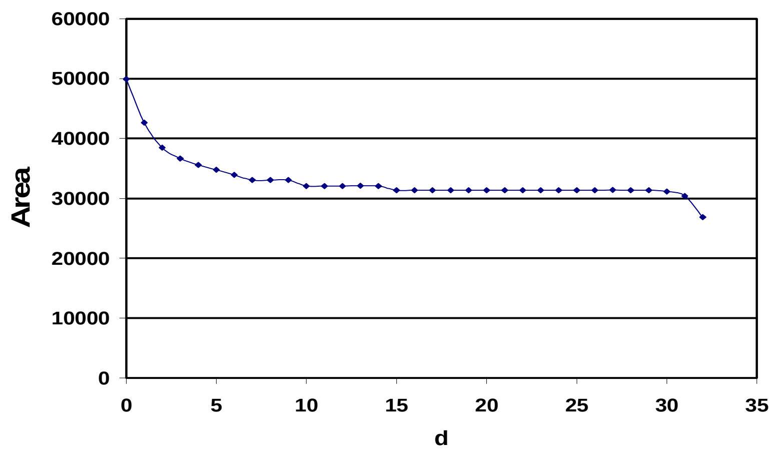

Figure 5.17: Area of the AU with conversion (TT).

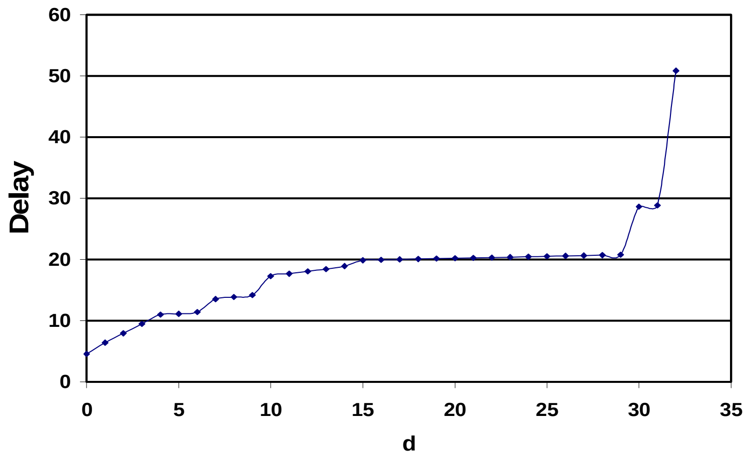

Figure 5.18: Delay of the AU with conversion (TT).

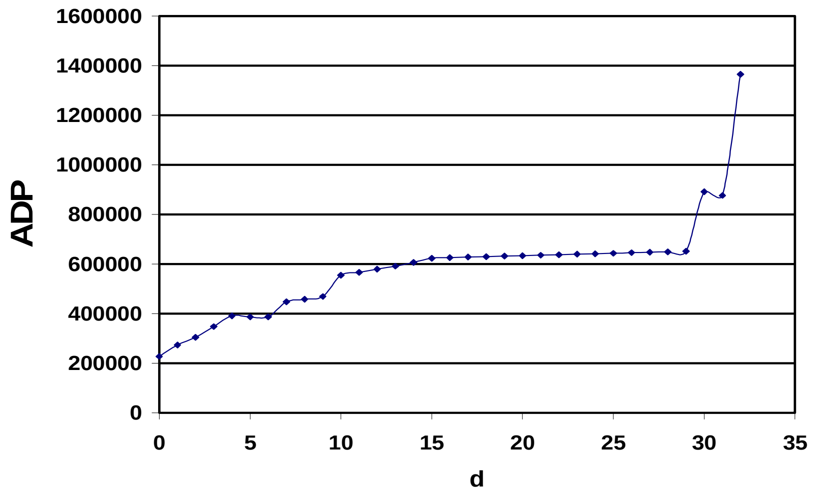

Figure 5.19: ADP of the AU with conversion (TT).

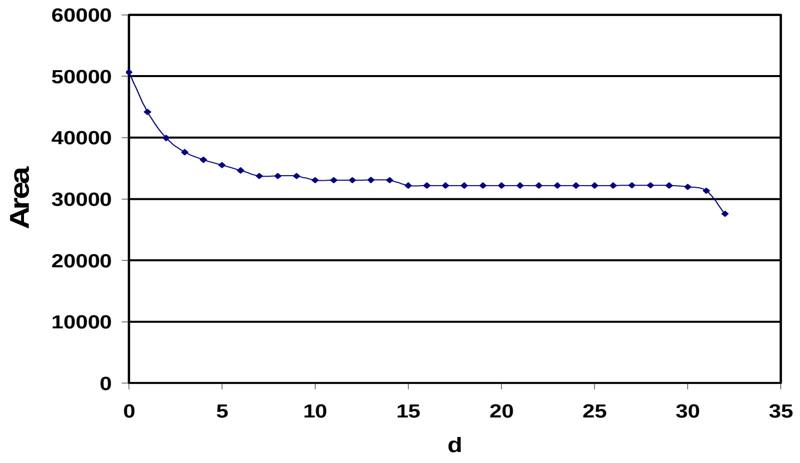

Figure 5.20: Area of the AU with conversion (SS).

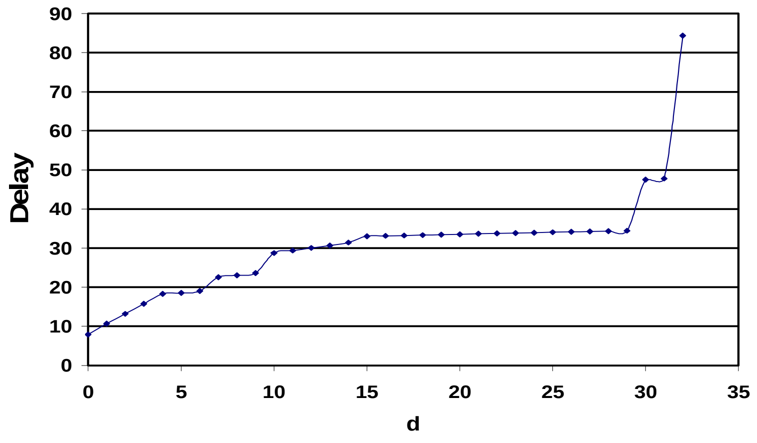

Figure 5.21: Delay of the AU with conversion (SS).

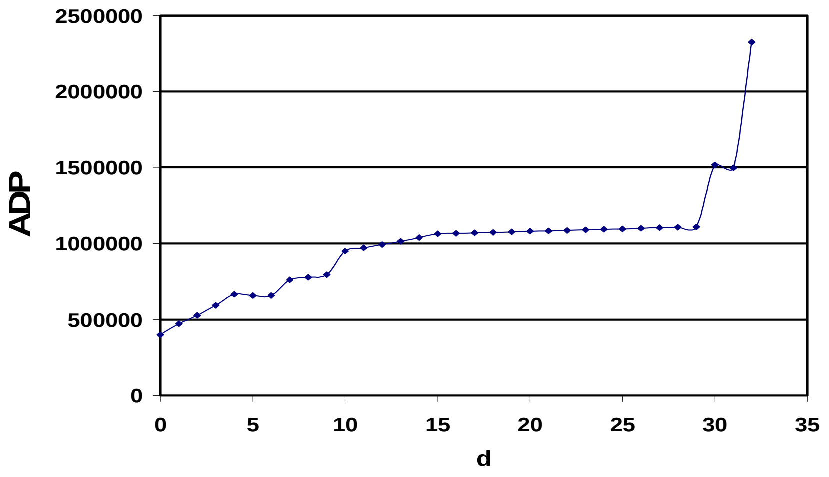

Figure 5.22: ADP of the AU with conversion (SS).

5.8 Analysis of the AU with conversion characterization results

In our analysis, we use the formula $\frac{value_{\max} - value_{\min}}{value_{\max}}$ to calculate the percentage of decrease in each metric.

Table 5.6: Analysis of the AU with conversion characterization results.

Metric	Percentage of decrease formula	Percentage of decrease (FF)	Percentage of decrease (TT)	Percentage of decrease (SS)
Area	$\frac{d_0 - d_{32}}{d_0}$	46.2%	46.2%	45.6%
Delay	$\frac{d_{32} - d_0}{d_{32}}$	91.9%	91%	90.6%
ADP	$\frac{d_{32} - d_0}{d_{32}}$	84.9%	83.3%	82.8%

We conclude the following from the former table and the former figures:

1. Area characterization
   - As the number of signed digits decreases in the architecture, it is possible to decrease the area.
2. Delay characterization.
   - As the number of signed digits increases in the architecture, it is possible to decrease the delay.
3. ADP characterization.
   - As the number of signed digits increases in the architecture, it is possible to decrease the ADP. (A special case is the range from (d_4 to d_5) and from (d_{30} to d_{31})).
   - The ADP curve is almost similar to the delay curve which means that the decrease in the delay is dominant over the increase in the area. (A special case is the range from (d_4 to d_5) and from (d_{30} to d_{31}), since the two ranges have a dominant area over the delay).

Table 5.7: Our AU with conversion comparison with the CLA AU with conversion.

Metric	Comparison formula	(FF)	(TT)	(SS)
Area	$\frac{d_0}{CLA}$	2	2	2
Delay	$\frac{CLA}{d_0}$	5.1	5.2	5.1
ADP	$\frac{CLA}{d_0}$	2.5	2.5	2.5

We can conclude that, the delay of the CLA AU with conversion is larger than the delay of our AU with conversion (d_0) as well as the ADP. However, the area of our AU with conversion (d_0) is larger than the area of the CLA AU with conversion.

5.9 Summary

In this chapter, an arithmetic unit has been proposed. A characterization of this AU has been done by changing the distance parameter. Also, the conversion from and to the conventional number system has been analyzed.

It was found that the most redundant architecture is the optimum with respect to the delay and the ADP. However, the non-redundant architecture is the optimum with respect to the area of the circuit.

Chapter 6

6- Energy characterization

6.1 Introduction

Historically, characterization of a digital circuit is mainly related to the characterization of the speed and the area of the circuit, however the power and energy are two important metrics nowadays especially for portable consumer devices like laptops and cellular phones.

In this chapter, we characterize the average energy consumed by the circuit with respect to the distance parameter.

6.2 Power and energy concepts

The power consumption of a circuit determines how much heat the circuit dissipates and how much energy is consumed per unit time, while energy consumption of a circuit is the integration of the power consumption over the period of operation.

The power consumption in digital CMOS circuits is expressed as

$$P = I_{leakage} V_{dd} + I_{sc} V_{dd} + \alpha C_L V^2_{dd} f_{clk} \qquad (6.1)$$

Where

- The leakage current $I_{leakage}$ is primarily determined by the fabrication technology, caused by:
    1. The reverse bias current in the parasitic diodes formed between source and drain diffusions and the bulk region in a MOS transistor.
    2. The subthreshold current that arises from the inversion that exists at the gate voltages below the threshold voltage.
- The short-circuit current I_{sc} is due to the DC path between the supply rails during output transitions.
- The last term refers to the capacitive power dissipation with α (referred to as the switching activity) being the average number of output transitions, C_L is the load capacitance at the output node, V_{dd} is the power supply voltage and f_{clk} is the clock frequency.

The details of each current component equations and detailed explanations are found in [5,11].

The term static power consumption refers to the leakage power dissipations, while the term dynamic power consumption refers to the sum of the short-circuit and the capacitive power dissipations [11].

The energy is the integration of the power consumption over the period of operation.

$$E = \int_0^T p(t)\ dt. \tag{6.2}$$

Where E is the energy consumption, p(t) is the total power consumption and T is the period of operation.

The power consumption is an important metric when analyzing the following two concerns:

1. Design of supply voltage

   The following equation relates the power consumption p(t) to the supply voltage V_{dd} and the current drawn from the supply $I_{dd}(t)$

   $$p(t) = V_{dd} * I_{dd}(t) \qquad (6.3)$$

   If the circuit consumes more power, this leads to a question whether the supply can support the required current to the circuit or not.

2. Design of cooling system

   As the power consumption increases, the heat dissipated in the circuit increases, so the requirement of a better cooling system is a mandatory issue. Otherwise the circuit temperature increases and it may affect the reliability of the system or cause damage to the system.

The energy consumption is an important metric when analyzing the following two concerns:

1. Running cost of the system

   The running cost of the system depends on the energy consumed from the system.
2. Battery life time

   In a battery operated system, the life time of the battery depends on the energy consumed from the system. As the energy consumed increases, it leads to a shorter life time. This factor is very important in portable devices.

Reducing the clock frequency reduces the power consumed by the circuit, however it does not directly change the energy used in the operation but it spreads the same energy used over a longer time. So, we use the energy consumed by the circuit as a metric rather than the power consumed in the characterization of our systems.

6.3 Energy characterization flow

The following flow is used in the energy characterization.

1- Synthesize the RTL on ASIC technology Artisan (TSMC 0.13), library “ff_1v26_m40c” is used in the synthesis process.(using design compiler tool)

2- Convert the gate netlist into spice netlist using v2spice tool.

3- Run RTL verification test bench using Questasim tool and use it to generate random vectors and store them into a file to be used as the input vectors during the next step.

4- Setup a test bench in Eldo simulator, in which the generated random vectors will stimulate the spice netlist of our circuit.

5- Compute $I_{dd}(t)$ which is the current drawn from the supply V_{dd}.

6- Multiply $I_{dd}(t)$ by V_{dd} which results in p(t) the power consumed by the circuit.

7- Integrate the power consumed by the circuit over the period of operation which results in the total energy consumed by the circuit.

8- Divide the total energy consumed by the number of inputs random vectors which results in the average energy consumed per vector by the circuit.

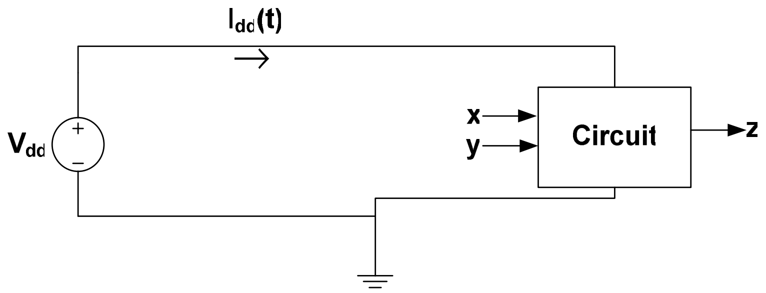

Figure 6.1: A simple test bench for measuring the energy consumption.

Figure 6.1 shows a simple test bench, it consists of

1. Supply voltage V_{dd} .
2. Circuit under test.

3. Inputs of the circuit which are stimulated from random input vectors stored in a file.

6.4 Input random vectors selection

Deciding the number of the input random vectors is very important as a large number of vectors leads to a long simulation time which may not be practical to achieve.

We do our selection by developing a test bench for a 16 bit carry look-ahead adder and make the following steps:

1. Generate 65536 (2^{16}) random vectors.
2. Apply them to the circuit, we choose the time between vectors as 10 ns which is enough for the circuit to stabilize completely before the next addition is performed.
3. Evaluate the total energy consumed according to the pervious flow and then divide it by 65536 to get the average energy per vector.
4. Repeat the previous steps for 10000, 4000, 2000, 1000, 100, 10, 5 and 4 random vectors.
5. Compare the average energy per vector in all trials.
6. Four random vectors give acceptable results, and have only 5% deviation from the results obtained by the 65536 vectors.
7. The ratio between 65536 and 4 is 16384(2^{14}).
8. Suppose we have a 32 bit adder, the number of acceptable input vectors is (2^{32} / 2^{14}) = 262144(2^{18}).
9. We use 262144 random vectors to evaluate the energy for a 32 bit carry look-ahead adder, the simulation run tends to take a 3.5 weeks.
10. We want to do a characterization for our circuits in terms of the energy consumption as the same we have done before for the area, delay and ADP, so that the above simulation time is not practical for finishing our simulation in an acceptable time.

11. We try more than one machine and they have the time from (3.5 - 4) weeks.
12. We try 131072 (2^{17}) vectors, simulation time tends to be from (1.5 - 2) weeks.
13. We try 65536 (2^{16}) vectors, simulation time tends to be from (4 - 7) days.
14. Finally, we try 32768 (2^{15}) vectors, simulation time tends to be from (1 - 3) days.

We do one trial of 131072 vectors, one trial of 65536 vectors and one trial of 32768 vectors and then compute the average energy per vector for each trial, the results almost the same with 2% deviations only, so according to the previous trials, we choose 32768 vectors as the number of input random vectors for all trials.

6.5 Energy characterization results

We do energy characterization for the following architectures:

Table 6.1: Energy characterization trials details.

Design	Time between vectors	Distances used
HSD adder	10 ns	d_0,d_4,d_9,d_{14},d_{20},d_{26} and d_{32}
HSD adder/sub Architecture 1	10 ns	d_0,d_4,d_9,d_{14},d_{20},d_{26} and d_{32}
HSD adder/sub Architecture 2	10 ns	d_0,d_4,d_9,d_{14},d_{20},d_{26} and d_{32}
HSD adder/sub Architecture 3	10 ns	d_3,d_7,d_{14},d_{20} and d_{26}
AU	10 ns (clk period)	d_0,d_4,d_9,d_{14},d_{20},d_{26} and d_{32}

Energy characterized in the following figures is measured in pJ (pico joule).

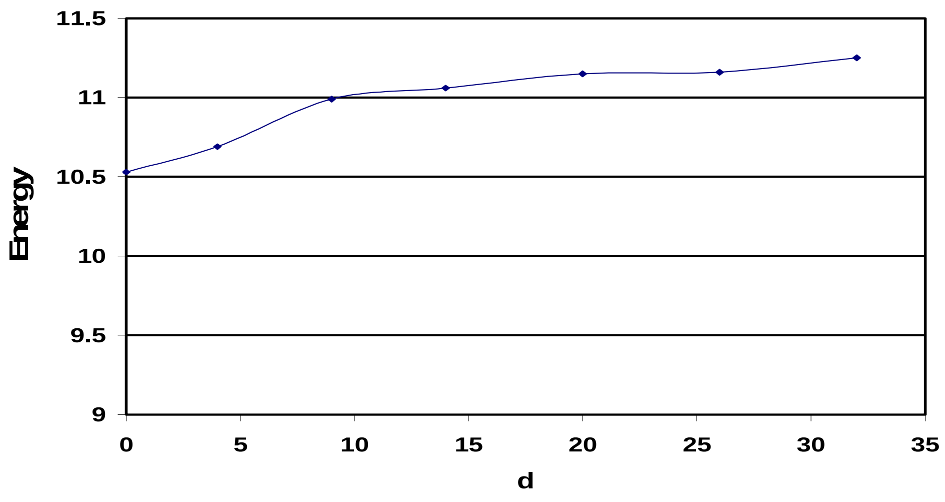

Figure 6.2: Energy of the HSD adder.

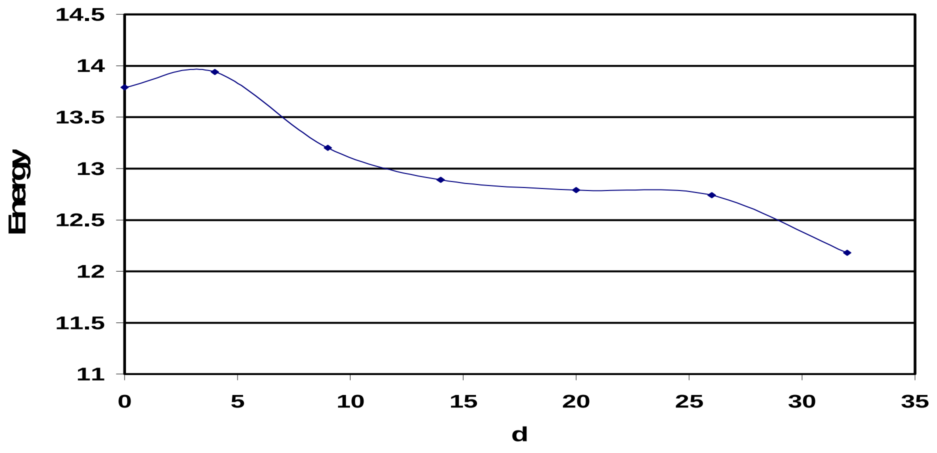

Figure 6.3: Energy of architecture 1.

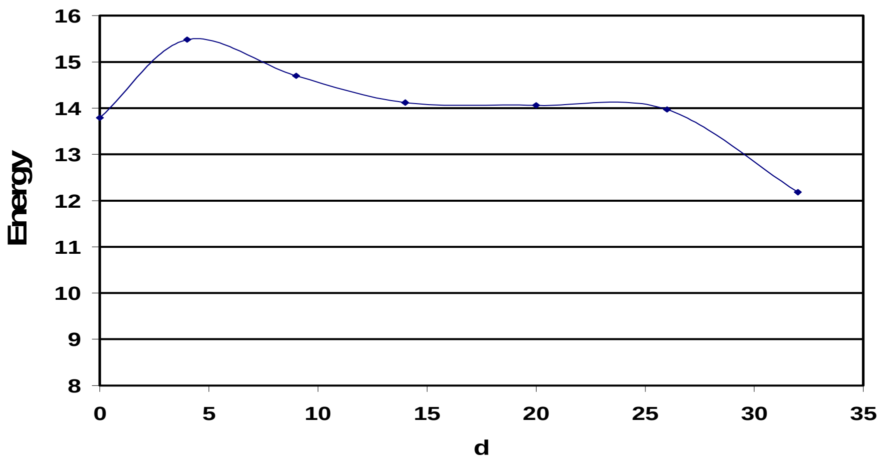

Figure 6.4: Energy of architecture 2.

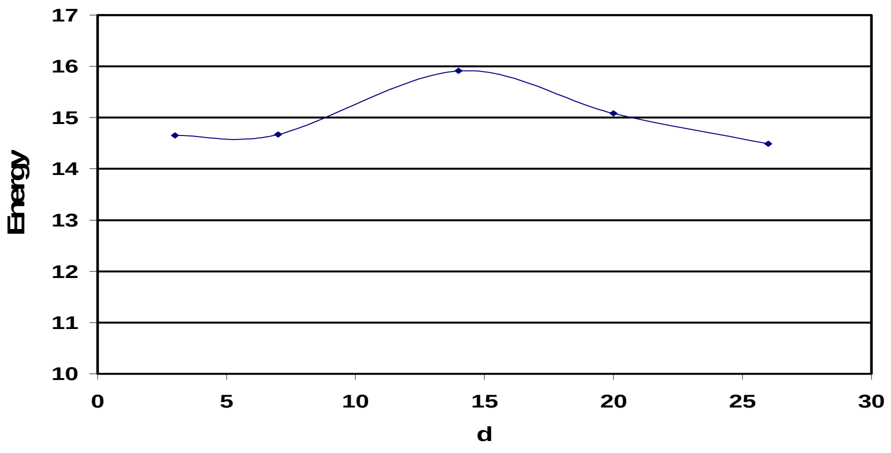

Figure 6.5: Energy of architecture 3.

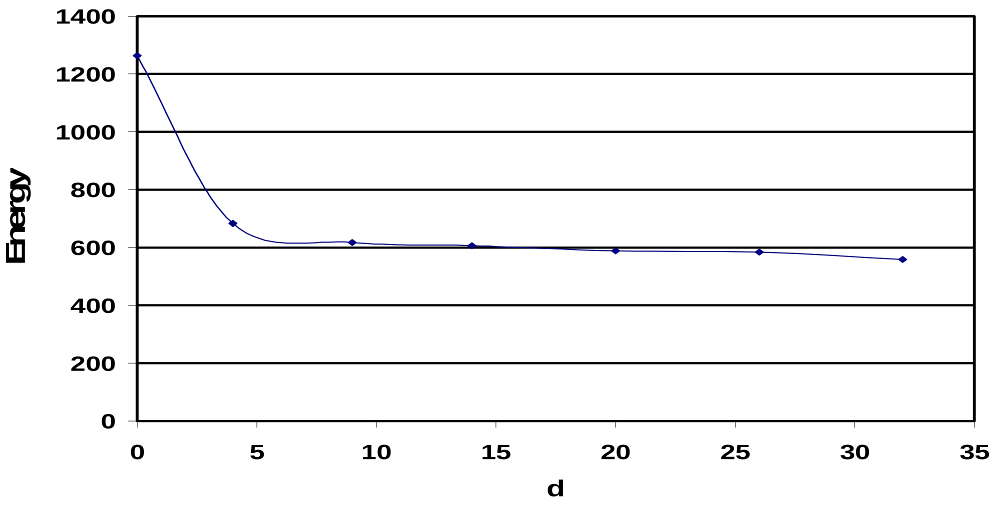

Figure 6.6: Energy of the AU.

In our analysis, we use the formula $\frac{value_{\max} - value_{\min}}{value_{\max}}$ to calculate the percentage of decrease.

Table 6.2: Energy characterization results.

Design	Percentage of decrease formula	Percentage of decrease
HSD adder	$\frac{d_{32} - d_0}{d_{32}}$	6.4%
HSD adder/sub Architecture 1	$\frac{d_4 - d_{32}}{d_4}$	12.6%
HSD adder/sub Architecture 2	$\frac{d_4 - d_{32}}{d_4}$	21.3%
HSD adder/sub Architecture 3	$\frac{d_{14} - d_{26}}{d_{14}}$	8.9%
AU	$\frac{d_0 - d_{32}}{d_0}$	55.7%

Table 6.3: Energy comparison with CLA architectures.

Design	Comparison formula	Normalized value
HSD adder	$\frac{d_0}{CLA}$	1.4
HSD adder/sub Architecture 1	$\frac{d_0}{CLA}$	1.6
HSD adder/sub Architecture 2	$\frac{d_0}{CLA}$	1.6
HSD adder/sub Architecture 3	$\frac{d_3}{CLA}$	1.7
AU	$\frac{d_0}{CLA}$	2.2

Finally, we conclude the following from the previous tables and figures:

1- The HSD adder characterization results is similar to Gopi et al [2] results, which indicates that the SD adder consumes less energy than the ripple-carry adder, or in other words the most redundant adder consumes less energy than the non-redundant one.

2- We do the comparison of the SD adder (HSD adder d_0) with the CLA adder which indicates that the SD adder consumes more energy than the CLA adder, so the conclusion in [2] is not accurate.

3- The characterization of the HSD adder/subtractor (architecture 1), the HSD adder/subtractor (architecture 2) and the AU indicates that as the number of signed digits increases in the architecture the energy consumption increases or in other words the redundant architecture consumes more energy than the non redundant one.(There is a special case in architecture 1 and architecture2 results. d_0 energy is less than

d_4, since the delay is dominant over the power in this region, so this leads to a decrease in the energy with respect to d_4).

4- Architecture 1 (d_0), Architecture 2 (d_0) and the AU (d_0) consume more energy than their CLA architectures which is consistent with the result that the HSD adder (d_0) consumes more energy than the CLA adder.

5- The HSD adder/subtractor (architecture 3) has a different behavior as the peak of the curve is not located near d_0 or d_{32}, however it is at d_{14}. Energy is the power multiplied by the delay, the delay increases as the signed digits decrease. The delay is minimum near d_0 and hence it can overcome the power increase, so that the energy has a lower value near (d_0). In the middle of the curve the multiplication of the delay and the power gives a large value. Near d_{32}, the power decrease overcome the increase in the delay hence the energy return back to a lower value again.

6- We can conclude that, the energy consumption depends mainly on the architecture. Some architectures have a minimum energy near the most redundant distances, others have a minimum energy near the non-redundant distances, while some architectures have a minimum near both the most redundant and the non-redundant distances. So it is the responsibility of the designer to select the architecture that satisfies his requirements.

7- Comparing architecture 1, 2 and 3. Architecture 1 is better than architecture 2 and better than architecture 3 from the energy point of view.

6.6 EDP characterization results

Delay is an indication of the speed and hence the performance of the circuit. Circuit with lower delay values means high speed circuit and hence high performance circuit. As we discussed earlier in this chapter, Energy is an indication of the running cost of the system.

In this section, we do an analysis of the EDP. We can use this term to make a comparison between two systems, if both have the same energy the one with the least delay is the optimum one. Also if we have two systems with the same delay the one with the least energy consumption is the optimum one. So we can conclude that the EDP is a useful metric when comparing two systems. Unit of the EDP is (pJ * ns).

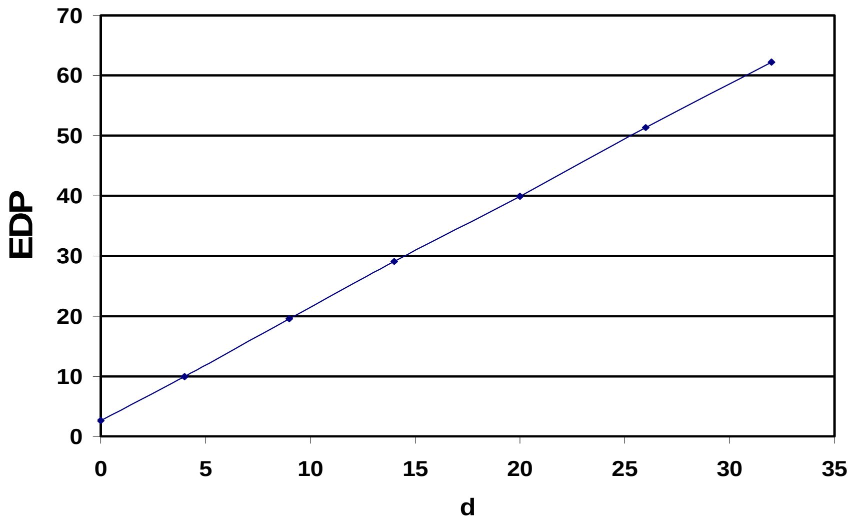

Figure 6.7: EDP of the HSD adder.

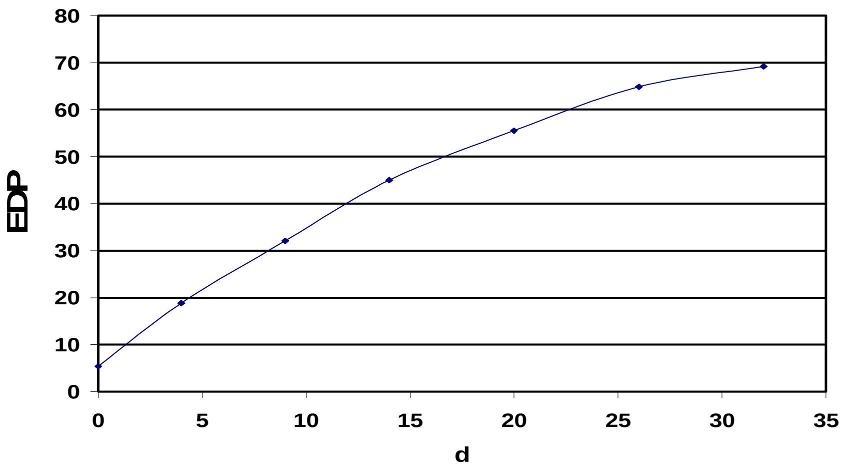

Figure 6.8: EDP of architecture 1.

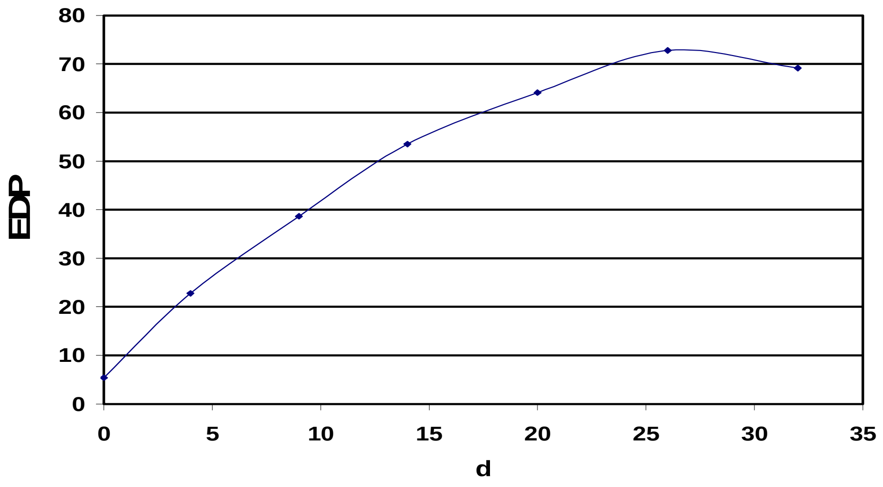

Figure 6.9: EDP of architecture 2.

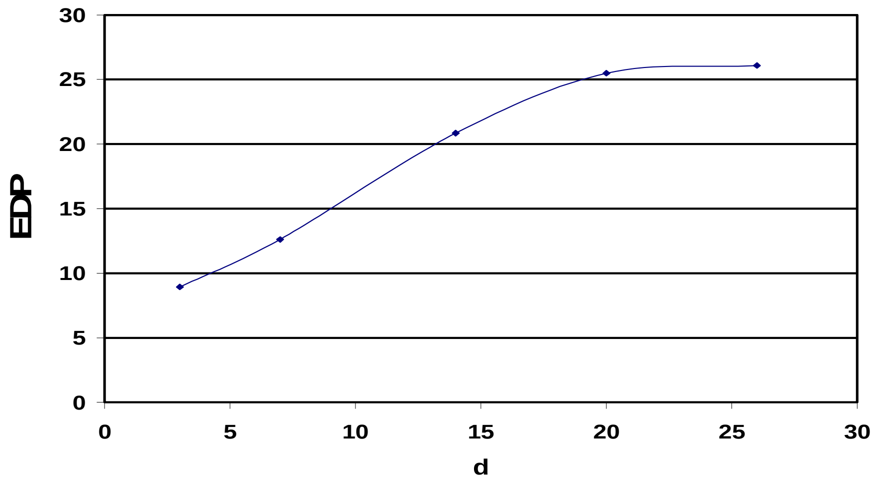

Figure 6.10: EDP of architecture 3.

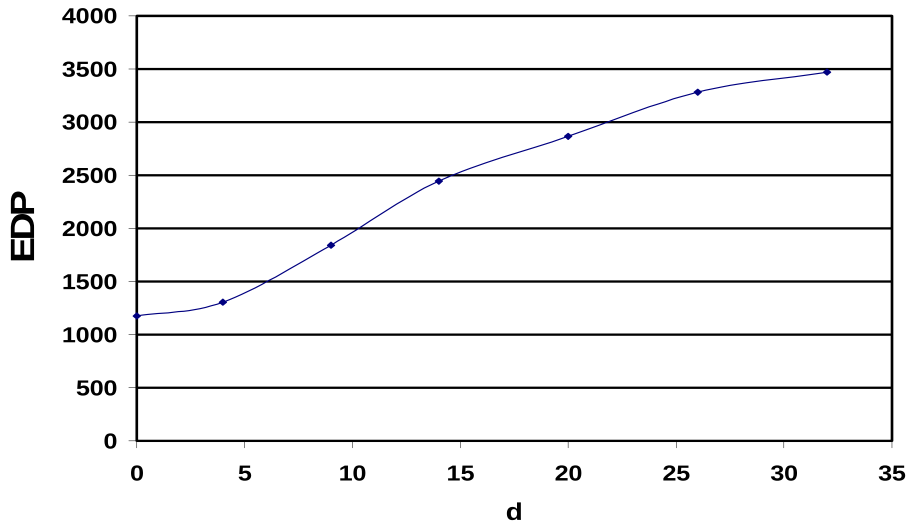

Figure 6.11: EDP of the AU.

Table 6.4: EDP characterization results

Design	Percentage of decrease formula	Percentage of decrease
HSD adder	$\frac{d_{32}-d_0}{d_{32}}$	95.7%
HSD adder/sub Architecture 1	$\frac{d_{32}-d_0}{d_{32}}$	92.2%
HSD adder/sub Architecture 2	$\frac{d_{26}-d_0}{d_{26}}$	92.5%
HSD adder/sub Architecture 3	$\frac{d_{26}-d_3}{d_{26}}$	65.7%
AU	$\frac{d_{32}-d_0}{d_{32}}$	66.1%

Table 6.5: EDP comparison with CLA architectures.

Design	Comparison formula	Normalized value
HSD adder	$\frac{CLA}{d_0}$	6
HSD adder/sub Architecture 1	$\frac{CLA}{d_0}$	3.5
HSD adder/sub Architecture 2	$\frac{CLA}{d_0}$	3.5
HSD adder/sub Architecture 3	$\frac{CLA}{d_3}$	2.1
AU	$\frac{CLA}{d_0}$	1.3

We conclude the following from the previous figures and tables:

1. The EDP decreases as the number of signed digits increases in the architecture.
2. There is a special case in the HSD adder/subtractor (architecture 2) at d_{32}, the decrease in the energy is dominant over the increase in the delay hence its EDP is lower than d_{26}.The rest of the curve satisfy conclusion in step 1.
3. The EDP curves are almost similar to delay curves, which mean the delay is dominant over the energy.
4. We can conclude that, as the number of signed digits increases in the architecture, it is possible to decrease the EDP. In other words, the most redundant design is the optimum from the EDP point of view.
5. Comparing architecture 1, 2 and 3. Architecture 3 is the optimum from the EDP point of view.

6.7 EDAP characterization results

As we discussed in the previous section, the delay is an indication of the performance of the system and the energy is an indication of the running cost of the system. On the other hand, the area is an indication of the initial cost of the system.

As the area of the design increases, it indicates the following:

1- More design efforts and time.
2- More verification efforts and time.
3- Silicon cost is a function of the circuit area.

Items 1 and 2 can be transferred into cost since more design efforts and verification efforts is mainly more cost, hence we can conclude that the area is an indication of the initial cost of the system.

In the following figures, characterization of the EDAP is done with respect to the distance d. The Unit of the EDAP is (pJ * ns * gate count).

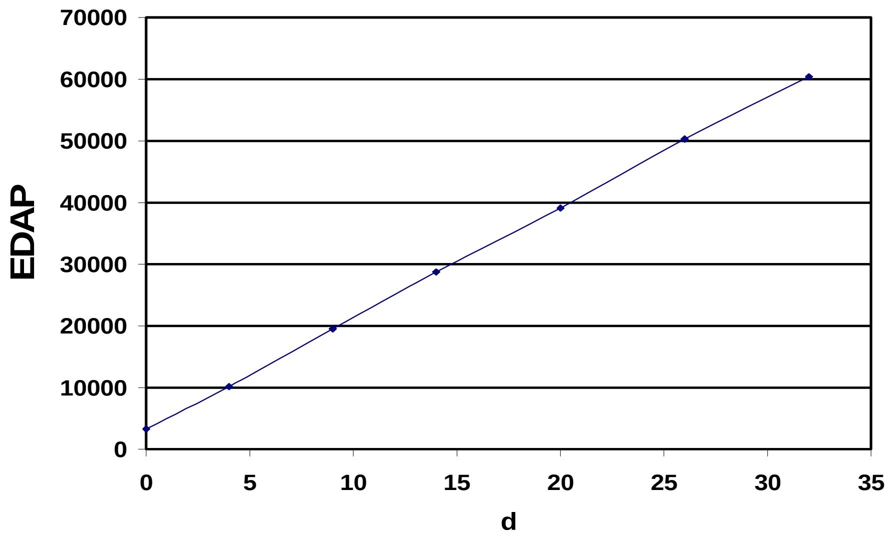

Figure 6.12: EDAP of the HSD adder.

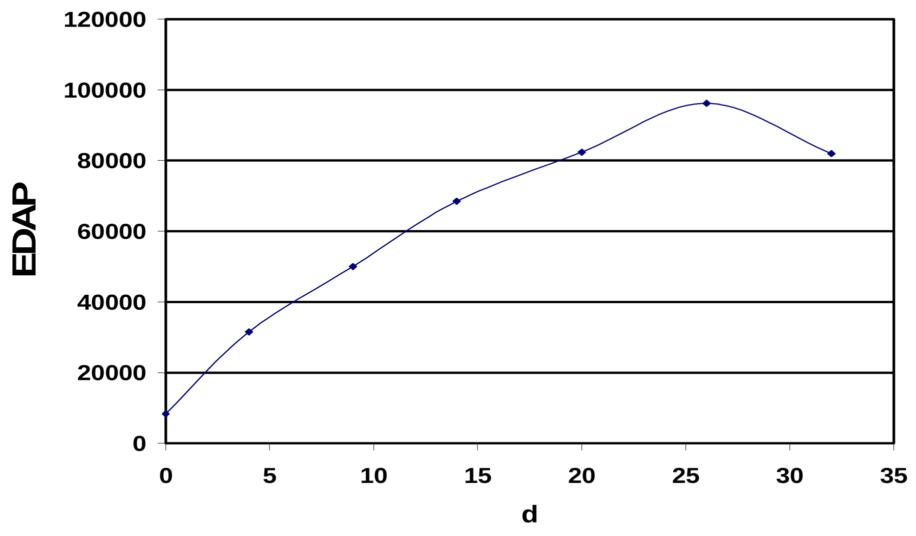

Figure 6.13: EDAP of architecture 1.

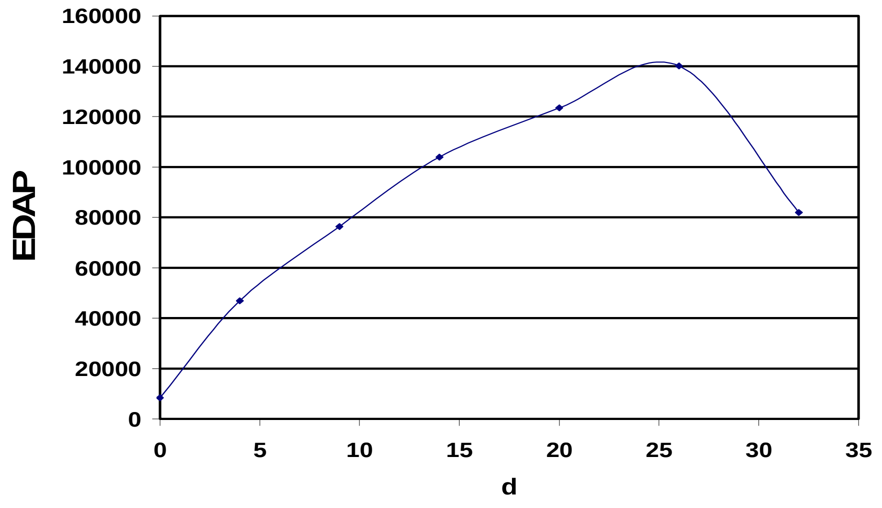

Figure 6.14: EDAP of architecture 2.

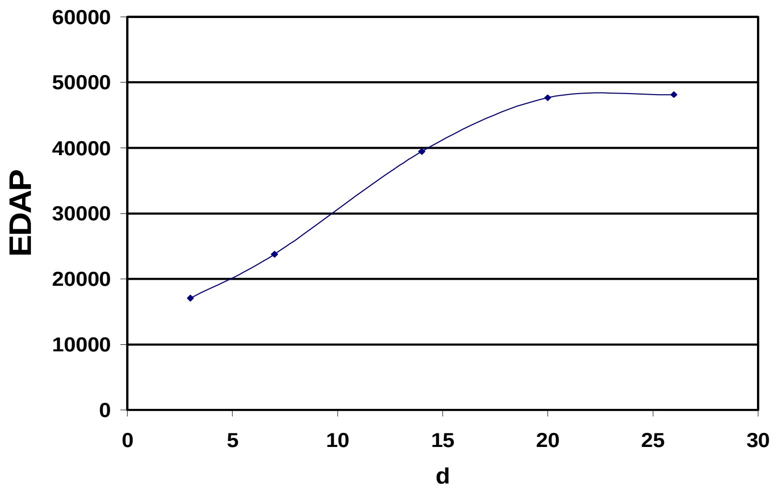

Figure 6.15: EDAP of architecture 3.

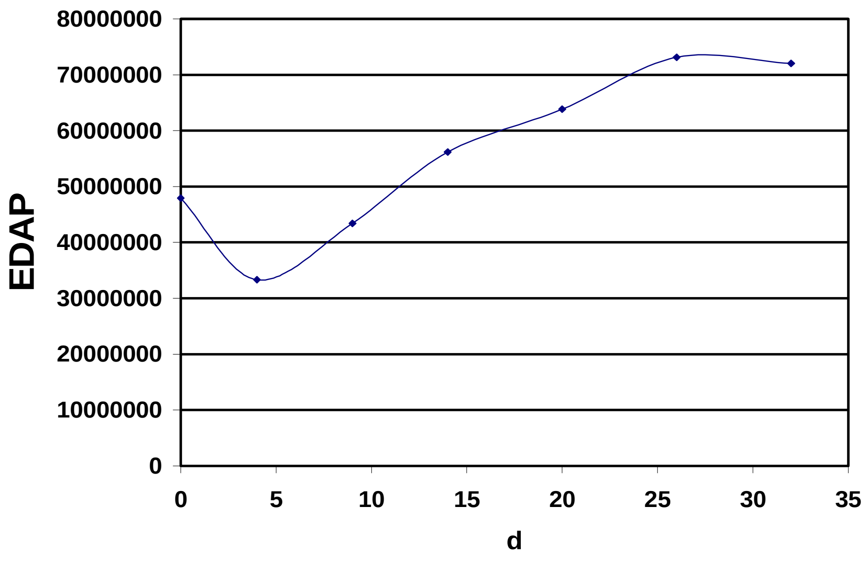

Figure 6.16: EDAP of the AU.

Table 6.6: EDAP characterization results.

Design	Percentage of decrease formula	Percentage of decrease
HSD adder	$\frac{d_{32}-d_0}{d_{32}}$	94.5%
HSD adder/sub Architecture 1	$\frac{d_{26}-d_0}{d_{26}}$	91.3%
HSD adder/sub Architecture 2	$\frac{d_{26}-d_0}{d_{26}}$	94%
HSD adder/sub Architecture 3	$\frac{d_{26}-d_3}{d_{26}}$	64.5%
AU	$\frac{d_{26}-d_4}{d_{26}}$	54.4%

Table 6.7: EDAP comparison with CLA architectures.

Design	Comparison formula	Normalized value
HSD adder	$\frac{CLA}{d_0}$	3.4
HSD adder/sub Architecture 1	$\frac{CLA}{d_0}$	2.1
HSD adder/sub Architecture 2	$\frac{CLA}{d_0}$	2.1
HSD adder/sub Architecture 3	$\frac{CLA}{d_3}$	1.04
AU	$\frac{d_0}{CLA}$	1.5

We conclude the following from the previous figures and tables:

1. As the number of signed digits increases in the architecture, it is possible to decrease the EDAP; however there are two special cases.

2. In architecture 1, 2 and the AU, d_{32} has a lower area value so it dominates the EDP; hence it has a smaller EDAP than d_{26}, which violates the previous conclusion.
3. In the AU, d_0 has a large area so it dominates the decrease in the EDP which results in a larger EDAP than d_4, and hence violates the previous conclusion.
4. Comparing architecture 1, 2 and 3. Architecture 3 is the optimum from the EDAP point of view.

6.8 Summary

In this chapter, the characterization of the energy, EDP and EDAP has been done. The energy consumption depends mainly on the architecture. The most redundant design is the optimum from the EDP point of view.

Chapter 7

7- Conclusions and future work

7.1 Conclusions

In this thesis, the analysis of the SD addition has been done and the SD adder has been implemented and compared with the traditional adder architecture in terms of its area and delay. The SD adder trades off area for a higher speed.

A HSD architecture has been analyzed as intermediate solution between the SD architecture and the conventional architecture. A HSD adder architecture has been analyzed, a HSD adder/subtractor and a modified version of this circuit have been proposed and a third HSD adder/subtractor has been analyzed. Also an arithmetic unit has been proposed and the analysis of the conversion to and from the conventional system has been done.

All the above architectures have been characterized by changing their distance parameter d and analyzing its effect on the architecture delay, area, ADP, energy, EDP and EDAP.

As the number of signed digits decreases in the mentioned architectures, it is possible to decrease the area. The decrease percentage ranges from (3% to 49%) depending on the architecture. The area is the only aspect that is always negatively affected by the increase of redundancy.

Depending on the architecture, as the number of signed digits increases it is possible to decrease the delay (percentage ranging from 66% to 95%) and the ADP (percentage ranging from 65% to 94%). The ADP curves are almost similar to the delay curves which mean that the decrease in the delay is dominant over the increase in the area in most of the cases.

Comparison with CLA architectures indicates that the SD architectures have improvement in their delay and ADP over the CLA architecture; however the SD architectures have a small increase in their area compared with the area of the CLA architectures.

We recommend using the most redundant architecture to achieve a small delay and small ADP. However, in low area design it is recommended to use the traditional architecture.

The energy consumption depends mainly on the architecture. Some architectures have a minimum energy near the most redundant distances; others have a minimum energy near the non-redundant distances, while some architectures have a minimum near both the most redundant and the non-redundant distances. It is the responsibility of the designer to select the architecture that satisfies the intended requirements.

As the number of signed digits increases in the architecture, it is possible to decrease the EDP. The decrease percentage ranges from (65% to 95%) depending on the architecture. In other words, the most redundant design is the optimum from the EDP point of view.

As the number of signed digits increases in the architecture, it is possible to decrease the EDAP in most cases. However in some cases the area curve is dominant over the EDP curve.

We recommend using the most redundant architecture to achieve a small EDP.

7.2 Future work

Based on the work presented in this thesis and the results obtained, we recommend the following items as the future work

1. Apply the characterization to 64 digits and 128 digits and analyze its effect on the architecture delay, area, ADP, energy, EDP and EDAP.
2. Use the HSD adder and the HSD adder/subtractor circuits implemented in this thesis in more complex arithmetic circuits such as multipliers and dividers.
3. Include the results of this work in the library of future CAD tools.

Appendix A

A- Characterization results

In this appendix, we show the details of all characterization results.

The following table shows the unit of each metric.

Table A.1: Unit of each metric.

Metric	Unit
Delay	nano second (ns)
Area	gate count
Energy	Pico joule (pJ)
ADP	(gate count * ns)
EDP	(pJ * ns)
EDAP	(pJ * ns * gate count)

A.1 Delay, Area and ADP results

In this section, we show the characterization results of all architectures in terms of their delay, area and ADP.

Table A.2: **HSD adder characterization results (FF).**

d	Area	Delay	ADP
0	1248	0.25	312
1	1110	0.42	466.2
2	1058	0.59	624.22
3	1040	0.76	790.4
4	1023	0.93	951.39
5	1014	1.1	1115.4
6	1006	1.27	1277.62
7	1006	1.44	1448.64
8	997	1.61	1605.17
9	997	1.78	1774.66
10	988	1.95	1926.6
11	988	2.12	2094.56
12	988	2.29	2262.52
13	988	2.46	2430.48
14	988	2.63	2598.44
15	988	2.8	2766.4
16	980	2.9	2842
17	980	3.07	3008.6
18	980	3.24	3175.2
19	980	3.41	3341.8
20	980	3.58	3508.4
21	980	3.75	3675
22	980	3.92	3841.6
23	980	4.09	4008.2
24	980	4.26	4174.8
25	980	4.43	4341.4
26	980	4.6	4508
27	980	4.77	4674.6
28	980	4.94	4841.2
29	980	5.11	5007.8
30	980	5.28	5174.4
31	980	5.45	5341
32	971	5.53	5369.63

Table A.3: HSD adder characterization results (TT).

d	Area	Delay	ADP
0	1248	0.43	536.64
1	1104	0.69	761.76
2	1050	0.94	987
3	1032	1.2	1238.4
4	1014	1.46	1480.44
5	1005	1.72	1728.6
6	996	1.98	1972.08
7	996	2.24	2231.04
8	987	2.49	2457.63
9	987	2.75	2714.25
10	978	3.01	2943.78
11	978	3.27	3198.06
12	978	3.53	3452.34
13	978	3.79	3706.62
14	978	4.04	3951.12
15	978	4.3	4205.4
16	969	4.39	4253.91
17	969	4.65	4505.85
18	969	4.91	4757.79
19	969	5.17	5009.73
20	969	5.43	5261.67
21	969	5.68	5503.92
22	969	5.94	5755.86
23	969	6.2	6007.8
24	969	6.46	6259.74
25	969	6.72	6511.68
26	969	6.98	6763.62
27	969	7.23	7005.87
28	969	7.49	7257.81
29	969	7.75	7509.75
30	969	8.01	7761.69
31	969	8.27	8013.63
32	960	8.39	8054.4

Table A.4: HSD adder characterization results (SS).

d	Area	Delay	ADP
0	1398	0.71	992.58
1	1232	1.14	1404.48
2	1170	1.57	1836.9
3	1150	2	2300
4	1129	2.43	2743.47
5	1119	2.85	3189.15
6	1108	3.28	3634.24
7	1108	3.71	4110.68
8	1098	4.14	4545.72
9	1098	4.57	5017.86
10	1088	5	5440
11	1088	5.43	5907.84
12	1088	5.86	6375.68
13	1088	6.28	6832.64
14	1088	6.71	7300.48
15	1088	7.14	7768.32
16	1077	7.3	7862.1
17	1077	7.73	8325.21
18	1077	8.15	8777.55
19	1077	8.58	9240.66
20	1077	9.01	9800.7
21	1077	9.44	10166.88
22	1077	9.87	10629.99
23	1077	10.3	11093.1
24	1077	10.73	11556.21
25	1077	11.16	12019.32
26	1077	11.58	12471.66
27	1077	12.01	12934.77
28	1077	12.44	13397.88
29	1077	12.87	13860.99
30	1077	13.3	14324.1
31	1077	13.73	14787.21
32	1067	13.94	14873.98

Table A.5: Architecture 1 characterization results (FF).

d	Area	Delay	ADP
0	1547	0.39	603.33
1	2027	0.76	1540.52
2	1740	0.9	1566
3	1587	1.1	1745.7
4	1675	1.35	2261.25
5	1637	1.56	2553.72
6	1583	1.78	2817.74
7	1560	1.99	3104.4
8	1560	2.21	3447.6
9	1560	2.43	3790.8
10	1522	2.64	4018.08
11	1522	2.85	4337.7
12	1522	3.06	4657.32
13	1507	3.28	4942.96
14	1523	3.49	5315.27
15	1484	3.71	5505.64
16	1484	3.83	5683.72
17	1484	3.96	5876.64
18	1484	4.08	6054.72
19	1484	4.21	6247.64
20	1484	4.34	6440.56
21	1484	4.47	6633.48
22	1484	4.59	6811.56
23	1484	4.71	6989.64
24	1484	4.84	7182.56
25	1484	4.96	7360.64
26	1484	5.09	7553.56
27	1469	5.18	7609.42
28	1476	5.31	7837.56
29	1485	5.43	8063.55
30	1451	5.48	7951.48
31	1446	5.65	8169.9
32	1185	5.68	6730.8

Table A.6: Architecture 1 characterization results (TT).

d	Area	Delay	ADP
0	1547	0.67	1036.49
1	2022	1.19	2406.18
2	1733	1.4	2426.2
3	1579	1.72	2715.88
4	1667	2.07	3450.69
5	1628	2.39	3890.92
6	1574	2.71	4265.54
7	1551	3.03	4699.53
8	1551	3.35	5195.85
9	1551	3.69	5723.19
10	1513	4	6052
11	1513	4.32	6536.16
12	1513	4.64	7020.32
13	1497	4.96	7425.12
14	1513	5.28	7988.64
15	1474	5.6	8254.4
16	1474	5.8	8549.2
17	1474	5.99	8829.26
18	1474	6.19	9124.06
19	1474	6.39	9418.86
20	1474	6.58	9698.92
21	1474	6.79	10008.46
22	1474	6.97	10273.78
23	1474	7.17	10568.58
24	1474	7.36	10848.64
25	1474	7.56	11143.44
26	1474	7.75	11423.5
27	1459	7.92	11555.28
28	1466	8.11	11889.26
29	1475	8.28	12213
30	1441	8.4	12104.4
31	1436	8.57	12306.52
32	1175	8.63	10140.25

Table A.7: Architecture 1 characterization results (SS).

d	Area	Delay	ADP
0	1664	1.06	1763.84
1	2235	2.1	4693.5
2	1907	2.41	4595.87
3	1734	2.95	5115.3
4	1816	3.5	6356
5	1772	4.03	7141.16
6	1710	4.56	7797.6
7	1682	5.09	8561.38
8	1682	5.62	9452.84
9	1682	6.15	10344.3
10	1637	6.68	10935.16
11	1637	7.21	11802.77
12	1637	7.74	12670.38
13	1622	8.27	13413.94
14	1639	8.8	14423.2
15	1593	9.33	14862.69
16	1593	9.66	15388.38
17	1593	9.99	15914.07
18	1593	10.31	16423.83
19	1593	10.64	16949.52
20	1593	10.97	17475.21
21	1593	11.29	17984.97
22	1593	11.62	18510.66
23	1593	11.95	19036.35
24	1593	12.27	19546.11
25	1593	12.6	20071.8
26	1593	12.93	20597.49
27	1577	13.23	20863.71
28	1584	13.54	21447.36
29	1594	13.76	21933.44
30	1554	13.8	21445.2
31	1549	14.13	21887.37
32	1282	14.23	18242.86

Table A.8: Architecture 2 characterization results (FF).

d	Area	Delay	ADP
0	1547	0.39	603.33
4	2063	1.47	3032.61
9	1976	2.63	5196.88
14	1943	3.79	7363.97
20	1925	4.56	8778
26	1925	5.21	10029.25
32	1185	5.68	6730.8

Table A.9: Architecture 3 characterization results (FF).

d	Area	Delay	ADP
3	1908	0.61	1163.88
7	1884	0.86	1620.24
14	1892	1.31	2478.52
20	1870	1.69	3160.3
26	1845	1.8	3321

Table A.10: CLA architectures characterization results (FF).

Architecture	Area	Delay	ADP
Adder	720	2.16	1555.2
Adder/sub	944	2.28	2152.32
AU	20396	2.79	56904.84

Table A.11: CLA architectures characterization results (TT).

Architecture	Area	Delay	ADP
Adder	704	3.7	2604.8
Adder/sub	928	3.87	3591.36
AU	20312	4.67	94857.04

Table A.12: CLA architectures characterization results (SS).

Architecture	Area	Delay	ADP
Adder	768	6.42	4930.56
Adder/sub	992	6.71	6656.32
AU	20304	8.01	162635.04

Table A.13: AU characterization results (FF).

d	Area	Delay	ADP
0	40771	0.93	37917.03
1	30910	1.32	40801.2
2	27712	1.5	41568
3	25808	1.65	42583.2
4	25521	1.91	48745.11
5	24790	2.12	52554.8
6	24277	2.33	56565.41
7	23592	2.54	59923.68
8	23592	2.75	64878
9	23592	2.98	70304.16
10	22970	3.17	72814.9
11	22970	3.39	77868.3
12	22970	3.6	82692
13	22975	3.82	87764.5
14	22970	4.03	92569.1
15	22285	4.24	94488.4
16	22285	4.37	97385.45
17	22285	4.49	100059.65
18	22285	4.62	102956.7
19	22285	4.74	105630.9
20	22285	4.87	108527.95
21	22285	5.01	111647.85
22	22285	5.12	114099.2
23	22285	5.25	116996.25
24	22285	5.37	119670.45
25	22285	5.5	122567.5
26	22285	5.62	125241.7
27	22290	5.72	127498.8
28	22285	5.88	131035.8
29	22285	6.02	134155.7
30	22250	6.04	134390
31	21490	6.15	132163.5
32	20760	6.21	128919.6

Table A.14: AU characterization results (TT).

d	Area	Delay	ADP
0	40475	1.5	60712.5
1	30690	2.08	63835.2
2	27502	2.35	64629.7
3	26089	2.6	67831.4
4	25385	2.96	75139.6
5	24770	3.27	80997.9
6	24088	3.58	86235.04
7	23467	3.89	91286.63
8	23467	4.2	98561.4
9	23467	4.52	106070.84
10	22682	4.83	109554.06
11	22682	5.15	116812.3
12	22682	5.46	123843.72
13	22686	5.78	131125.08
14	22682	6.1	138360.2
15	22178	6.41	142160.98
16	22178	6.61	146596.58
17	22178	6.81	151032.18
18	22178	7.01	155467.78
19	22178	7.21	159903.38
20	22178	7.41	164338.98
21	22178	7.61	168774.58
22	22178	7.81	173210.18
23	22178	8.02	177867.56
24	22178	8.22	182303.16
25	22178	8.42	186738.76
26	22178	8.62	191174.36
27	22183	8.77	194544.91
28	22178	9.01	199823.78
29	22178	9.22	204481.16
30	22143	9.3	205929.9
31	21430	9.39	201227.7
32	20677	9.45	195397.65

Table A.15: AU characterization results (SS).

d	Area	Delay	ADP
0	40538	2.46	99723.48
1	31013	3.41	105754.33
2	27764	3.85	106891.4
3	25996	4.28	111262.88
4	25273	4.9	123837.7
5	24616	5.41	133172.56
6	23948	5.94	142251.12
7	23328	6.47	150932.16
8	23328	6.98	162829.44
9	23328	7.51	175193.28
10	22863	8.03	183589.89
11	22863	8.56	195707.28
12	22863	9.07	207367.41
13	22870	9.6	219552
14	22863	10.12	231373.56
15	22233	10.64	236559.12
16	22233	10.98	244118.34
17	22233	11.31	251455.23
18	22233	11.65	259014.45
19	22233	11.98	266351.34
20	22233	12.32	273910.56
21	22233	12.65	281247.45
22	22233	12.98	288584.34
23	22233	13.32	296143.56
24	22233	13.65	303480.45
25	22233	13.99	311039.67
26	22233	14.32	318376.56
27	22240	14.56	323814.4
28	22233	14.96	332605.68
29	22233	15.31	340387.23
30	22191	15.37	341075.67
31	21600	15.57	336312
32	20735	15.69	325332.15

Table A.16: AU with conversion characterization results (FF).

d	Area	Delay	ADP
0	50237	2.7	135639.9
1	42928	4.05	173858.4
2	38709	5.05	195480.45
3	36432	6.05	220413.6
4	35760	7.07	252823.2
5	34832	7.17	249745.44
6	34145	7.37	251648.65
7	33239	8.77	291506.03
8	33239	8.99	298818.61
9	33239	9.21	306131.19
10	32420	11.25	364725
11	32420	11.49	372505.8
12	32420	11.74	380610.8
13	32448	11.96	388078.08
14	32420	12.3	398766
15	31537	13.01	410296.37
16	31537	13.06	411873.22
17	31537	13.1	413134.7
18	31537	13.14	414396.18
19	31537	13.19	415973.03
20	31537	13.23	417234.51
21	31537	13.28	418811.36
22	31537	13.32	420072.84
23	31537	13.37	421649.69
24	31537	13.41	422911.17
25	31537	13.45	424172.65
26	31537	13.5	425749.5
27	31565	13.54	427390.1
28	31533	13.59	428533.47
29	31537	13.63	429849.31
30	31327	18.74	587067.98
31	30544	18.97	579419.68
32	26988	33.44	902478.72

Table A.17: AU with conversion characterization results (TT).

d	Area	Delay	ADP
0	49927	4.56	227667.12
1	42668	6.41	273501.88
2	38450	7.93	304908.5
3	36659	9.48	347527.32
4	35570	11	391270
5	34755	11.12	386475.6
6	33897	11.41	386764.77
7	33054	13.53	447220.62
8	33054	13.85	457797.9
9	33054	14.18	468705.72
10	32069	17.28	554152.32
11	32069	17.66	566338.54
12	32069	18.06	579166.14
13	32097	18.42	591226.74
14	32069	18.91	606424.79
15	31366	19.88	623556.08
16	31366	19.95	625751.7
17	31366	20.01	627633.66
18	31366	20.07	629515.62
19	31366	20.14	631711.24
20	31366	20.2	633593.2
21	31366	20.26	635475.16
22	31366	20.32	637357.12
23	31366	20.39	639552.74
24	31366	20.45	641434.7
25	31366	20.51	643316.66
26	31366	20.58	645512.28
27	31394	20.64	647972.16
28	31362	20.7	649193.4
29	31366	20.76	651158.16
30	31156	28.62	891684.72
31	30418	28.82	876646.76
32	26852	50.84	1365155.68

Table A.18: AU with conversion characterization results (SS).

d	Area	Delay	ADP
0	50659	7.89	399699.51
1	44192	10.68	471970.56
2	39926	13.2	527023.2
3	37628	15.76	593017.28
4	36382	18.3	665790.6
5	35495	18.51	657012.45
6	34634	19	658046
7	33747	22.54	760657.38
8	33747	23.05	777868.35
9	33747	23.58	795754.26
10	33052	28.72	949253.44
11	33052	29.38	971067.76
12	33052	30.05	993212.6
13	33096	30.63	1013730.48
14	33052	31.43	1038824.36
15	32192	33.04	1063623.68
16	32192	33.14	1066842.88
17	32192	33.24	1070062.08
18	32192	33.34	1073281.28
19	32192	33.44	1076500.48
20	32192	33.55	1080041.6
21	32192	33.65	1083260.8
22	32192	33.75	1086480
23	32192	33.85	1089699.2
24	32192	33.95	1092918.4
25	32192	34.05	1096137.6
26	32192	34.16	1099678.72
27	32236	34.26	1104405.36
28	32208	34.36	1106666.88
29	32192	34.46	1109336.32
30	31944	47.5	1517340
31	31329	47.81	1497839.49
32	27553	84.37	2324646.61

A.2 Energy, EDP and EDAP results

In this section, we show the characterization results of all architectures in terms of their energy, EDP and EDAP.

Table A.19: HSD adder characterization results.

D	Energy	EDP	EDAP
0	10.53	2.6325	3285.36
4	10.69	9.9417	10170.3591
9	10.99	19.5622	19503.5134
14	11.06	29.0878	28738.7464
20	11.15	39.917	39118.66
26	11.16	51.336	50309.28
32	11.25	62.2125	60408.3375

Table A.20: Architecture 1 characterization results.

D	Energy	EDP	EDAP
0	13.79	5.3781	8319.9207
4	13.94	18.819	31521.825
9	13.2	32.076	50038.56
14	12.89	44.9861	68513.8303
20	12.79	55.5086	82374.7624
26	12.74	64.8466	96232.3544
32	12.18	69.1824	81981.144

Table A.21: Architecture 2 characterization results.

d	Energy	EDP	EDAP
0	13.79	5.3781	8319.9207
4	15.48	22.7556	46944.8028
9	14.7	38.661	76394.136
14	14.12	53.5148	103979.2564
20	14.06	64.1136	123418.68
26	13.97	72.7837	140108.6225
32	12.18	69.1824	81981.144

Table A.22: Architecture 3 characterization results.

D	Energy	EDP	EDAP
3	14.65	8.9365	17050.842
7	14.67	12.6162	23768.9208
14	15.91	20.8421	39433.2532
20	15.08	25.4852	47657.324
26	14.49	26.082	48121.29

Table A.23: AU characterization results.

d	Energy	EDP	EDAP
0	1263.6	1175.148	47911959.11
4	683.28	1305.0648	33306558.76
9	617.54	1840.2692	43415630.97
14	606.67	2444.8801	56158895.9
20	588.23	2864.6801	63839396.03
26	583.98	3281.9676	73138647.97
32	558.71	3469.5891	72028669.72

Table A.24: CLA architectures characterization results.

design	Energy	EDP	EDAP
adder	7.32	15.8112	11384.064
Adder/sub	8.27	18.8556	17799.6864
AU	551.46	1538.5734	31380743.07

References

[1] G. Jaberipur and B. Parhami, “Constant-Time Addition with Hybrid-Redundant Numbers: Theory and Implementations,” Intergration, The VLSI Journal, vol. 41, no. 1, pp. 49-64, Jan. 2008.

[2] S. K. Gopi, H. A. H. Fahmy, and A. P. Vinod, “Redundant Adders Consume Less Energy,” IEEE Asia Pacific Conference on Circuits and Systems, pp. 422-425, Dec. 2006.

[3] G. Jaberipur, B. Parhami, and M. Ghodsi, “An Efficient Universal Addition Scheme for All Hybrid-Redundant Representations with Weighted Bit-Set Encoding,” Journal of VLSI Signal Processing, vol. 42, no. 2, pp. 149-158, Feb. 2006.

[4] J. Deschamps, G. J. A. Bioul, and G. D. Sutter, “Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems,” John Wiley & Sons Inc., 2006.

[5] N. H. E. Weste, and D. Harris, “CMOS VLSI Design: A Circuits and Systems Perspective,” Pearson Education Inc., 2005.

[6] W. Chen, “Logic Design,” CRC Press, Florida, 2003.

[7] G. Jaberipur, B. Parhami, and M. Ghodsi, “Weighted Bit-Set Encodings for Redundant Digit Sets: Theory and Applications,” Proceedings of the 36th Asilomar Conference on Signals, Systems and Computers, pp. 1629-1633, Nov. 2002.

[8] I. Koren, “Computer Arithmetic Algorithms,” A k Peters, Natick, Massachusetts, 2002.

[9] M. J. Flynn, and S. F. Oberman, “Advanced Computer Arithmetic Design,” John Wiley & Sons Inc., New York, 2001.

[10] B. Parhami, “Computer Arithmetic: Algorithms and Hardware Designs,” Oxford University Press, Oxford, 2000.

[11] K. K. Parhi, “VLSI Digital Signal Processing Systems: Design and Implementation,” John Wiley & Sons Inc., 1999.

[12] J. M. Rabaey, “Digital Integrated Circuits: A Design Perspective,” Prentice Hall International, New Jersey, 1996.

[13] D. S. Phatak, and I. Koren, “Hybrid Signed-Digit Number System: A Unified Framework for Redundant Number Representations with Bounded Carry Propagation Chains,” IEEE Transactions on Computers, vol. 43, no. 8, pp. 880-891, Aug. 1994.

[14] P. Kornerup, “Digit-Set Conversions: Generalizations and Applications,” IEEE Transactions on Computers, vol. 43, no. 5, pp. 622-629, May 1994.

[15] M. M. Mano, “Computer System Architecture,” Prentice-Hall International, New Jersey, 1993.

[16] M. M. Mano, “Digital Design,” Prentice-Hall International, New Jersey, 1991.

[17] B. Parhami, “Generalized Signed-Digit Number Systems: A Unifying Framework for Redundant Number Representations,” IEEE Transactions on Computers, vol. C-39, pp. 89-98, Jan. 1990.

[18] S. Kuninobu, T. Nishiyama, H. Edamatsu, T. Taniguchi, and N. Takagi, "Design of High Speed MOS Multiplier and Divider Using Redundant Binary Representation," Proceedings of the 8th IEEE Symposium on Computer Arithmetic, pp. 80-86, July 1987.

[19] N. Takagi, H. Yasuura, and S. Yajima, "High Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree," IEEE Transactions on Computers, vol. C-34, no. 9, pp. 789-796, Sep. 1985.

[20] J. J. F. Cavanagh, "Digital Computer Arithmetic: Design and Implementation," McGraw-Hill, 1984.

[21] O. J. Bedrij, "Carry-Select Adder," IRE Transactions on Electronic Computers, vol. EC-11, pp. 340-346, June 1962.

[22] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," IRE Transactions on Electronic Computers, vol. EC-10, pp. 389-400, Sep. 1961.