STUDY OF PARRALLEL ADDERS AND ALGORITHM TO OPTIMIZE ADDER WITH DELAY PROFILE

Nazimudeen Naveen and Chip-Hong Chang
School of Electrical & Electronic Engineering
Nayang Technological University, Nanyang Avenue, Singapore 639798
e-mail: b2149207@ntu.edu.sg

Abstract
This report presents a study on some of the commonly used adders. The adders studied include the linear time ripple carry and Manchester carry chain adders, the logarithmic time carry look ahead adder and its variations such as the ELM adder, the square root time carry skip and carry select adder, the constant time signed-digit and carry save adders and the residue adders. Algorithms of these adders are analyzed and using the ternary operators they are formulated into mutually comparable forms, based on which their performance can be compared. Algorithms to optimize the performance of two operand addition for a given operands’ delay profile are studied. Graph based representations of the algorithms were developed, and suggestions for their functional and timing simulations are also given.

1. Introduction
This UROP research investigates an interesting topic of the hybrid adder design whereby the architectural features of different adders are exploited to fulfill the application constraints or improve a specific performance metric of a complex arithmetic. One of the typical applications for hybrid adder is in cases where the operands do not arrive at the same time. In some applications there is a specific arrival time distribution for the operands. A best example for this is in multiplication, after the partial products have been reduced to two by a tree or an array adder [1, 6, 8].

The aim of this project is to develop an adder algorithm and subsequently an architecture best suited for additions involving operands with pre-characterized delay profile. To achieve this, a comprehensive study of all the adder architectures was conducted. Algorithms of various kinds of adders were studied; their merits and demerits were also analyzed.

This report can be broadly divided into three parts. The first part of the report reviews some of the adders. The necessary details pertaining to the operations of these adders are also dealt upon. The second part of the report discusses the Generalized Earliest-First Algorithm developed by Wen-Chang Yeh and Chein-Wei Jen [4, 10]. The third part elaborates on further work that needs to be done to successfully implement the algorithm.

2. Adders
Addition is the most frequently used operation in digital signal processing. Adders have never ceased to be an area of intensive research and consequently a great number of adder designs have been made available in the open literatures [3]. The best adder would depend on the application for which it is intended. Adders are generally graded based on its size, speed, power consumption, and fault tolerance.

2.1 Ripple Carry Adder
Ripple Carry adders (RCA) [6-8] consist of cascaded full adders. It is an O(n) time and O(n) area adder. In the worst case a carry can propagate from the Least Significant Bit (LSB) to the Most Significant Bit (MSB). RCA is the simplest adder with the least hardware complexity amongst the adders. Fig. 1 shows the RCA adder and the CMOS implementation of the 1-bit full adder that can be cascaded to make the RCA.

Fig. 1 (a) Block diagram of RCA (b) CMOS implementation of full adder.
2.2 Manchester Carry Chain Adder

The Manchester Carry chain adder (MCC) [3, 8, 9] works on the same principle as that of the RCA with a separate chain to propagate the carry quickly. The architecture of MCC is shown in Fig. 2. It uses the carry generate, propagate and kill functions [8] as follows:

\[
\begin{align*}
\text{Propagate: } & P_i = a_i \oplus b_i \quad (1) \\
\text{Generate: } & G_i = a_i b_i \quad (2) \\
\text{Kill: } & K_i = a_i b_i \quad (3)
\end{align*}
\]

Once a carry is generated, it is quickly propagated along a chain of pass transistors until it is killed or reaches the most significant bit position. MCC is commonly implemented using buffered dynamic CMOS (domino) logic [3, 7]. The most important feature of MCC adder is every stage has only one transistor delay as opposed to one adder delay in RCA.

2.3 Carry Lookahead Adder

Carry Lookahead Adder (CLA) [2, 6-8] is very popular due to its speed and modularity. It has \(O(\log n)\) time and \(O(n\log n)\) area complexities, respectively. It can be shown that

\[
C_{i+1} = \sum_{k=1}^{i+1} G_{k-1} \prod_{j=k}^{i} P_{j-1} 
\]

where \(G_i = a_i b_i\) and \(P_i = a_i \oplus b_i\). This would theoretically mean that the entire carry can be calculated in one stage. However, due to the restriction imposed by the limited fan-in of the gates, the carry is calculated in stages. The concept of group delay and group propagate is applied to alleviate the fan-in problem. Fig. 3 shows the architecture of a 3 stage carry lookahead adder.

2.4 ELM Adder

The ELM adder [5, 7] is a special type of CLA. The novelty of ELM adder is that it directly computes the sum bits in parallel, thereby reducing the amount of interconnects. C. Nagendra et. al. [7] found ELM adder the best amongst the two’s complement adders in terms of the overall power-delay-area performance.

ELM algorithm uses a binary tree of simple processors depicted in Fig. 4 to perform the addition. The leaves of the tree receive the inputs and the outputs are taken from the rightmost node at each level of the tree. One output bit is obtained from each level of 0 and 1 of the tree, and \(2^{i-1}\) bits from the rightmost node of each of the remaining levels.

The leaves of the tree calculate the partial sums and pass them up. At higher levels, the tree nodes receive the partial sums from their immediate predecessors one level below, along with the information required to update the partial sum at that level of the tree. For \(0 \leq i < j \leq n-1\), the following group signals are defined:

\[
\begin{align*}
P(j,i) &= p_j P(j-1,i) \quad (5a) \\
P(i,i) &= p_i \quad (5b) \\
G(j,i) &= g_j + p_j G(j-1,i) \quad (5c) \\
G(i,i) &= g_i \quad (5d)
\end{align*}
\]

The leaf nodes at the lowest level have to calculate the propagate and carry signals. The following functions to be calculated by the other nodes at higher levels are illustrated by an excerpt of the node structure shown in Fig 5.

\[
\begin{align*}
G_A(r,p) &= G_B(r,q) + P_B(r,q)G_C(q-1,p) \quad (6a) \\
P_A(q+j,p) &= P_a(q+j,q)P_C(q-1,p) \quad (6b)
\end{align*}
\]
\[ p_{q+j}^s = p_{s+q}^a \oplus P_b(q+j-1, q)G_C(q-1, p) \]  

(6c)

### 3. Ternary Operators

Fast addition can be achieved by using parallel adders which can be categorized into two classes of algorithms: carry look ahead and conditional sum based algorithms. To understand the distinctive behavioral differences and to be able to compare the adders, a unified succinct representation to analytically model the mathematical implications of the different fast adders is essential.

Wen-Chang Yeh and Chein-Wei Jen [4] suggested three ternary operators and two fundamental properties of their operations. The sum and carry signals of all the abovementioned parallel fast adders can be elegantly formulated by logical equations based on these ternary operators. These equations establish the basis for the performance comparison of these adder schemes.

The three ternary operators \( \otimes, \nabla \) and \( \Delta \) are defined as follows:

\[
\begin{align*}
& a \otimes b \oplus c = a \otimes (b, c) \\
& b \oplus c \otimes a = a \nabla (b, c) = c \nabla (b, a) \\
& b \Delta c \otimes a = a \Delta (b, c) = c \Delta (b, a)
\end{align*}
\]

(7)

(8)

(9)

The following rules apply when they are operating on two pairs of Boolean variables.

\[
\begin{align*}
(a, b) = (\alpha, \beta) \nabla (\gamma, \delta) \Rightarrow \begin{cases} 
\alpha = \alpha \nabla (\gamma, \delta) \\
\beta = \beta \nabla (\gamma, \delta)
\end{cases}
\end{align*}
\]

(10)

\[
\begin{align*}
(a, b) = (\alpha, \beta) \Delta (\gamma, \delta) \Rightarrow \begin{cases} 
\alpha = \alpha \Delta (\gamma, \delta) \\
\beta = \beta \Delta (\gamma, \delta)
\end{cases}
\end{align*}
\]

(11)

(12)

The ternary operators defined above have two important properties:

Associativity:

\[
[(a, b) \text{ op } (c, d)] \text{ op } (e, f) = (a, b) \text{ op } [(c, d) \text{ op } (e, f)]
\]

Non-commutative:

\[
(a, b) \text{ op } (c, d) \neq (c, d) \text{ op } (a, b)
\]

(13)

(14)

The associative property of the ternary operation can be exploited later to improve the adder algorithms to suit operands with a specified delay profile.

Using these operators the sum equation of the ELM adder can be written as

\[
s_i = [c_0 \nabla (g_i, p_i)] \nabla ... \nabla g_{i-2, p_{i-2}} \Delta (p_i \otimes g_{i-1, p_{i-1}})
\]

(15)

and the carry propagation signal can be written as

\[
c_i = c_0 \nabla (g_i, p_i) \nabla ... \nabla (g_{i-2, p_{i-2}}) \nabla (g_{i-1, p_{i-1}}) \nabla (g_i, p_i)
\]

(16)

where the ‘generate’ signal, \( g = a \oplus b \), and the ‘propagate’ signal, \( p = a \otimes b \).

Fig 6 shows the calculation of \( s_7 \) of ELM adder. The process is expressed in terms of ternary operators in a tree structure. The generation of the other sum bits can be similarly expressed based on Eqn. (15). The entire adder structure can be obtained by superimposing all of them one over the other. It should be noted that the nodal tree structure described in the earlier section can be obtained by appropriate reorganization of the ternary operators.

### 4. Generalized Earliest-First Algorithm

This algorithm [10] uses the associative property of the ternary operators to improve the addition speed. When some bits arrive earlier than the rest, those bits are combined first provided the associative property or non commutative property is not violated. The combination also does not have to start from the LSB as the combined terms can still be treated as primitive inputs to be combined later with other terms.

The algorithm consists of five steps for an \( L \)-bit addition:

1) Setup: Initialize an array of \( L \) elements. For \( 0 \leq i \leq L-1 \).

Each entry in \( P_{-List} \) consists of the signal arrival timings and
the bit position. Initialize a \( T_{-List} \) with zero elements and the
structure of each entry is equivalent to that of the \( P_{-List} \).

**Fig. 6.** Generation of \( s_7 \) of the ELM adder by structuring ternary operators in a tree.

It is worth noting that the ternary operators alone are inherently incapable of generating an optimal and superior architecture of the originally proposed ELM adder [5]. All that the operators convey is the operations (processes) that happened in the adder. Specifically, only the algorithmic details are represented, not the implementation and structural details.
2) Generate new T_List: Sort the P_List in an ascending order of the arrival timing. Move the elements that are equivalent to P_List[0] from P_List to T_List. Sort T_List in an ascending order of the bit position.

3) Combine adjacent bits: In T_List, if there are signals adjacent to each other, combine them with ternary operator. Insert the bit positions and timings of the newly generated terms back into the P_List.

4) Repeat: Repeat Steps 2 and 3 until only one element is left in the P_List.

5) Generate sum signals and end.

Two terms are considered adjacent to each other when there is no other term between them. The algorithm can be further optimized by considering the blocking factor, r. For example if there are three adjacent bits in the T_List, two ternary operators can be cascaded to compute a three bit block, i.e. r=3. The optimal blocking factor depends on the types of adder used. For CLA, the optimal blocking factor is generally greater than 2.

A MATLAB program was written to compute the algorithm for a given delay profile. The output of the program for a sample delay profile of 3 4 5 5 5 5 5 5 4 4 4 4 4 3 3 2 1 (20 bits) is shown in Fig. 7.

![Fig. 7. MATLAB output of the GEF algorithm](image)

5. Conclusion

In this report some adder configurations were studied. With excellent performance and well structured architecture, ELM adder is the preferred adder for two’s complement addition. The complicated ELM adder architecture can be better understood if a tree-like structure could be used to analyze the adder that is being designed as ELM adder. Such a structure has eliminated the massive interconnection details and focuses on its functional behavior. By using the ternary operators, adders can be described by recursive mathematical expression independent of their topologies. This made comparison of performance at algorithmic level possible. Although the critical path timing analysis of the adder has not been emphasized in this paper, they can be readily modeled by summing up the timings of the ternary operators in the critical path.

Generalized Earliest-first Algorithm [10] improves the speed of addition when the operands have a delay profile. The algorithm has been implemented in MATLAB in this project. Successful implementation of the algorithm provides new insight and opportunities into the formulation of novel hybrid fast adders. Whether the optimisation task of the adders can be left to the EDA tools or some other methods can be used, remain to be explored.

References