An Analysis of Multipliers in a New Binary System

R.K. Dubey & Anamika Pathak

Department of Electronics and Communication Engineering,
Swami Vivekanand University,
Sagar (M.P.) India 470228

Abstract: Bit-sequential multiplier is being studied in a (+1, -1) binary representation. The maximum length of multiplier for a fix point numbers consists of n module. Each module is a five bit adder which contains 5 inputs and 3 outputs. It also contains an additional pin at the input as well as at the output. Computation time for multiplication is of the order of O(n). Hardware realization of module is being discussed. In this design the input generated bit sequentially and the output is also generated bit sequentially. The model is being compared with the bit sequential multiplier in conventional binary system and the merits and demerits are described in detail.

Index Terms: Add-shift multiplier, (+1, -1) Representation, Carry-save addition, unified way.

I. INTRODUCTION:
In digital signal processing design of fast efficient multipliers is currently an interesting area of research. Multipliers have been extensively studied by various authors [1-5] for VLSI design. Many scientific and engineering problems, such as inversion of materials, solution of line or equation, computation of eigen value, Fast Fourier Transform and Discrete Fourier Transform and modular arithmetic etc. require large number of multiplications. For computer hardware designers, it is most important to know current research concerning the multiplier for LSI technology.

The conventional add-shift technique of multiplication take time O(n^2) with n being the maximum lengths of multiplier and multiplicand. This time can be reduced to O(n log n) using carry look ahead technique for addition but it complicates the hardware design.

Bit serial arithmetic is often used in parallel systems with high connectivity to reduce the wiring to a reasonable level. When multiplication is required, a serial-parallel multiplier is a typical choice but this scheme is not always possible. When both the inputs are required to enter serially at the same time then serial multiplier is required. Bit serial input and output multipliers referred to as on line multipliers [1] are potentially attractive in speeding up the execution of arithmetic expression, where multiprocessor arithmetic structure is needed. Bit serial input and output multiplier was presented by Lyon [2].

Following [7] serial multiplier operation for positive and negative members in (+1, -1) representation can be constructed in the following way. 

\[ P_n \ldots P_1 = a_k \ldots a_1 \times b_k \ldots b_1 \]
error is removed by adding an extra bit +1. It also contains sum bit \( P_2 \) and two bits carry for the next step. The sum bit \( P_2 \) total 5 bits for addition. The addition of 5 bits produces a the carry bit from the previous stage. Thus, there will be \( P_3 \) contains the product bit \( a_2b_2 \) and bits \( a_2b_1 \) and \( b_2a_1 \). These bits are shifted one bit left and added to the sum bit \( P_1 \) after shifting one bit left. The sum bit \( P_2 \) contains the product bits \( a_2b_1 \) and \( b_2a_1 \) and bit –1 and –1 respectively. To overcome the effect of two bits, two extra bits of opposite nature +1 and +1 are added. Thus there will be 5 bits in all. The sum will produce \( P_2 \) with the two bits as carry for the next step.

Step-4 consists of the product bits \( a_4b_4 \), \( a_3b_3 \), \( a_4b_3 \), \( b_3a_2 \), \( b_2a_2 \) and \( b_2a_1 \). Addition of bits take place in similar fashion as discussed in step-3. Finally, we find that \( P_1 \), \( P_2 \), \( P_3 \) are free bits and they do not take part in addition. Four 5 bit adders are required for whole multiplication process. Fig. (2) points out the specific example for four bit numbers A = \( \overline{1} \overline{1} \overline{1} \overline{1} \) and \( B = 1 \overline{1} \overline{1} \overline{1} \). The carry as outputs. 4 bit serial multiplier for 4th clock cycle connection of module is shown in Fig. (4). The third module of adder produces a “two bit” sum where one bit of the sum is extra bit to make the result error free because an error bit +1 exists in the input. This extra bits is +1. Thus the sum bit contains an additional bit +1. Fourth adders contain fixed bits \( T \), \( T \), 1, 1 and one product bit \( a_4b_2 \). The sum bit is always the product bit. Hardware realisation of a four bit serial multiplier during 4th clock interval is shown in Fig. (5).

Multiplication of two bits is realised by XNOR gate. The module of 5 bit adder is discussed earlier. 1 bit latch contains D flip-flop. D-flip flop is discussed by Tiwari and Varma. 2 bit latch for carry or sum contain two D-flip-flops working synchronously. Two bits are latched together. It produces one clock delay because two bits are entered simultaneously and taken out together. Bits are entered in queue fashion. Queue register implementation is shown in Fig. (6). It consists of shift registers. Transistor switches and tristate inverter are connected in parallel and connected to shifts registers. Switch and tristate inverters are controlled by clock. When clock is high (+1), the switch becomes closed and on the other hand if it is low (-1), tristate inverters are energized. The two bits are multiplied by XNOR gate and the output bit energizes the D-flip-flops. Initially the LSB is kept +1 and other bits are at -1 in the register. First D-flip-flop is activated because clk is +1 and other flip-flops are inactive because clock is low (-1) and so, data is entered in 1st flip-flop. On the other hand, if the clock is low (-1) all the bits of registers get inverted and again 1\textsuperscript{st} flip-flop is at +1 and others at (-1). Thus the state of flip-flops remains unchanged. When the 2\textsuperscript{nd} clock enters, the 2nd bit of register becomes +1 and others becomes -1 and the 2\textsuperscript{nd} D-flip-flop becomes active and remaining flip-flops remain inactive. Thus, Data + 1 or -1 is entered sequentially in queue fashion in the flip-flops.
Step-1. \[\begin{array}{c}
a_1 \\
b_1 \\
\hline \\
1 \\
1 \\
\hline
P_1
\end{array}\]

Step-2. \[\begin{array}{c}
a_2 \\
b_2 \\
\hline \\
1 \\
1 \\
\hline
P_1
\end{array}\]

\[\begin{array}{c}
a_2b_2 \\
a_2b_1 \\
\hline \\
1 \\
1 \\
\hline
P_3 \\
P_2 \\
P_1
\end{array}\]

Step-3. \[\begin{array}{c}
a_3 \\
b_3 \\
\hline \\
1 \\
1 \\
\hline
P_3 \\
P_2 \\
P_1
\end{array}\]

\[\begin{array}{c}
a_3b_3 \\
a_3b_2 \\
a_3b_1 \\
\hline \\
1 \\
1 \\
\hline
P_5 \\
P_4 \\
P_3 \\
P_2 \\
P_1
\end{array}\]

Step-4. \[\begin{array}{c}
a_4 \\
b_4 \\
\hline \\
1 \\
1 \\
\hline
P_3 \\
P_4 \\
P_5 \\
P_2 \\
P_1
\end{array}\]

\[\begin{array}{c}
a_4b_4 \\
a_4b_3 \\
a_4b_2 \\
a_4b_1 \\
\hline \\
1 \\
1 \\
\hline
P_7 \\
P_6 \\
P_5 \\
P_4 \\
P_3 \\
P_2 \\
P_1
\end{array}\]

Fig. 1 Step by Step multiplication process of four bit numbers.
\[ A = \overline{1111} = -11 \]
\[ B = \overline{111} = +3 \]

**Step-1.**
\[
\begin{array}{cccc}
\text{a}_1 & = & 1 \\
\text{b}_1 & = & 1
\end{array}
\]
\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{array}
\]
\[ 1 \rightarrow \text{Carry} \]
\[ 1 \rightarrow \text{Sum} \]

**Step-2.**
\[
\begin{array}{cccc}
\text{a}_2 & = & 1 \\
\text{b}_2 & = & 1
\end{array}
\]
\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{array}
\]
\[ 1 \rightarrow \text{Carry} \]
\[ 1 \rightarrow \text{Sum} \]

**Step-3.**
\[
\begin{array}{cccc}
\text{a}_3 & = & 1 \\
\text{b}_3 & = & 1
\end{array}
\]
\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{array}
\]
\[ 1 \rightarrow \text{Carry} \]
\[ 1 \rightarrow \text{Sum} \]

**Step-4.**
\[
\begin{array}{cccc}
\text{a}_4 & = & 1 \\
\text{b}_4 & = & 1
\end{array}
\]
\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{array}
\]
\[ 1 \rightarrow \text{Carry} \]
\[ \text{Ignored} \]
\[ 1 \rightarrow \text{Sum} \]

**Carry**
\[
\begin{array}{cccc}
1 & 1 & 1 & 1
\end{array}
\]

**Carry**
\[
\begin{array}{cccc}
1 & 1 & 1 & 1
\end{array}
\]

**Carry**
\[
\begin{array}{cccc}
1 & 1 & 1 & 1
\end{array}
\]

Thus the result:
\[
\begin{array}{cccc}
1 & 1 & 1 & 1
\end{array}
\]

\[ (-1)^3 \times 2^7 + \overline{1} \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + \overline{1} \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \]
\[ = 0 \times 64 + 32 + 16 - 16 - 4 + 2 + 1 \]
\[ = -16 + 2 + 1 = -33 \]

*Fig. 2 Example for four bit multiplication.*
Fig. 3 Block Diagram of 5 bit Adder

Fig. 4 Connection of 5-bit adders in module.

Fig. 5 Hardware Realization of a four bit serial multiplier during 4th cycle interval
CONCLUSION:
The system (+1, -1) is well suited for sequential multiplier. The system requires n number of five bit adders for n bit multiplication of positive as well as negative numbers. The algorithm used by author [7] is well applicable to this system with certain modifications. The hardware realization is different. It requires 5 bit adders in different way. It produces two bit carry and two bit sum having the same weight. This is one of extra feature of 5 bit adder. One requires two bit latch in addition to one bit. Multiplication of two bits is acquired by XNOR gate instead of AND gate. The queue register implementation requires additional hardware a switch and tristate inverter. Positive as well as negative numbers are multiplied with the same type hardware. In this way, the hardware realization of multiplier is straight and simple. The main advantage of this system is the unified implementation of positive as well as negative numbers. The two's complement multipliers are realized in different manners, as discussed by authors [8-10] compared to positive numbers. Our model is well suited for digital signal processing where parallel processing for positive and negative number are required simultaneously. A multiplier cell can be constructed and connected in modules, which helps in LSI design. The draw back of this system is that it is partially suitable for computation because even numbers can not be generated.

REFERENCES:
(14) A. Reyhani- Masolrh, “ Low complexity word level sequential normal basis multiplier” IEEE Transaction on Computers-2005