Extended Sequential Logic for Synchronous Circuit Optimization and its Applications

Pramod Kumar Meher, Senior Member, IEEE

Abstract—In this paper, we present a new approach for the extension of sequential logic functionality of D flip-flop in order to perform an additional Boolean function simultaneously along with its usual bit-storage function. We show that a combinational function of the form \((a \cdot b), (a + b), (a + \bar{b})\) or \((\bar{a} \cdot b)\) which occurs frequently in a feed-forward path with a D flip-flop could be implemented efficiently by a D flip-flop with RESET or SET provision. Similarly, \((a \oplus b)\) or \(((a-b)\oplus c)\) in the feedback loop with a D flip-flop could be implemented by a T flip-flop by suitable modification of the clock. The use of such extended sequential logic is found to result in a significant reduction in critical-path delay and saving in area-complexity over the direct implementation. Moreover, we present a simple approach for the construction of CMOS T flip-flop by modification of clock signal of D flip-flop, which is found to be more efficient than the T flip-flop derived from JK flip-flop. The extended sequential logic is used for the implementation of finite field multiplication over \(GF(2^m)\) and carry-save addition of real numbers. In both these cases, the use of extended logic is found to offer a substantial saving in area and time-complexity over the conventional implementations.

Index Terms—Sequential logic, combinational logic, digital arithmetic, finite field arithmetic, carry-save addition, logic optimization.

I. INTRODUCTION

Synchronous digital circuits mostly consist of combinational components interspersed with sequential components, e.g., D flip-flops and T flip-flops. While the D flip-flop is used popularly as input/output registers, pipelining latches, and feedback elements in synchronous systems, T type flip-flops are used in counters and clock divider circuits. In this paper, we examine the possibilities and advantages of combining the functions of basic logic elements like AND, OR, and XOR gates with that of D flip-flop. Let us name such extension of functionality of sequential elements to incorporate the combinational Boolean logic as “extended sequential logic.” The extended logic could be of two basic types such as

(i) the combinational logic in feed-forward path with the sequential logic; and
(ii) the combinational logic in feedback loop with the sequential logic.

The combinational logic in feed-forward path with a sequential element like a D flip-flop serves usually the same function when combinational function precedes the sequential one and vice versa. The basic Boolean function in a feed-forward path could be \((a \cdot b), (a + b), (a + \bar{b})\) or \((\bar{a} \cdot b)\). Except \((a \oplus b)\) none of these functions appear to be practically relevant to be used in the feedback loop with a D flip-flop, since the output get stuck at fixed output state on particular input conditions. Some typical forms of extended sequential logic of practical relevance which we have investigated, and aim at presenting in this paper are:

- \((a \cdot b)\) followed by D flip-flop
- \((a + b)\) followed by D flip-flop
- \((a + \bar{b})\) or \((\bar{a} \cdot b)\) followed by D flip-flop
- \((a \oplus b)\) followed by D flip-flop
- \((a \oplus b)\) in feedback loop with D flip-flop
- \(((a \cdot b) \oplus c)\) in feedback loop with D flip-flop

where \(a, b\) and \(c\) are Boolean variables.

In a recent paper, we have shown that an XOR gate in the feedback loop with a D flip-flop could be replaced by a T flip-flop for reducing the critical-path and area-complexity in finite field accumulator [1]. But, we do not find any specific design for CMOS implementation of T flip-flop. In the existing CAD tools, we do not have library cell for the T flip-flop. Conventionally, T operation is performed either by a JK flip-flop or by a D flip-flop. The JK flip-flop-based implementation of T flip-flop usually involves much higher area and more propagation delay, while the conventional derivation of T flip-flop from D flip-flop is not suitable for some of the applications, where the state is required to be toggled by a control input. Therefore, we present a different derivation of T flip-flop from the CMOS implementation of D flip-flop. The proposed construction could be used to design a library cell for T flip-flop where the output could be toggled in each clock period when the control input is 1. We have shown here that the extended sequential logic could be used not only for achieving reduction in critical-path delay, but also to reduce the overall circuit area of a chip. To demonstrate its advantages, we have used it in carry-save addition of real numbers and finite field multipliers over \(GF(2^m)\). The extended sequential logic is found to offer substantial saving in area and time-complexities over the conventional implementations.

The rest of this paper, is organized as follows. The proposed extension of sequential logic is discussed in Section II; and the CMOS construction of T flip-flop is discussed in Section III. The complexity of implementation of extended sequential logic and its relative advantages over the conventional realizations are discussed in Section IV. The application of proposed techniques in carry-save addition and multiplication in \(GF(2^m)\) along with the resulting advantages are discussed in Section V. Conclusions are placed in Section VI.
II. SEQUENTIAL LOGIC EXTENSION

We discuss here the proposed scheme for integrating a basic Boolean operation with a neighboring sequential element. The key idea we use here is based on the fact that when a combinational function of the form \(a \cdot b\) is followed by the storage of output-bit \(c = a \cdot b\) by a D flip-flop on a feed-forward path, the SET and RESET signals of the flip-flop could be utilized to perform some combinational functions. Synchronous SET and RESET should, however, be preferably used because, they are simpler to implement, and do not impose significant area/delay overhead. Besides, if an asynchronous SET/RESET is used, and that arrives long after the clock, the output may appear at the next cycle. In most practical situations, the flip-flop is not required to be set or reset during normal operation. The set or reset is required mostly for register initialization, and could be performed by the initialization of inputs \(a\) and \(b\) of the combinational logic. Similarly, when \((a \oplus b)\) or \(((a \cdot b) \oplus c)\) is in the feedback loop with a D flip-flop, clock of the flip-flop could be utilized to have simpler realizations.

A. Feed-Forward Sequential Logic Extension

1) \((a \cdot b)\) Followed by a D flip-flop: The logic symbol of an AND operation \((c = a \cdot b)\), and subsequent storage of output bit \(c\) by a D flip-flop is shown in Fig.1(a). A combination of truth table of AND gate and state-transition table of a positive-edge triggered D flip-flop is shown as D flip-flop in Fig.1(a). The same function as that of AND gate followed by D flip-flop can be obtained by using \(a\) as D input and \(b\) as active-low synchronous RESET of a D flip-flop, as shown in Fig.1(c). It is easy to find that the arrangement of Fig.1(c) has the same truth table as that in Fig.1(b).

2) \((a + b)\) Followed by a D flip-flop: The logic symbol of \(c = (a + b)\) followed by storage of output bit \(c\) in a D flip-flop is shown in Fig.2(a). The state-transition-truth table for
Fig. 5. (a) Logic symbol of an XOR gate in feedback loop with a D flip-flop. (b) Characteristic table. \((Q : \text{present state and } Q^+: \text{next state})\)

Fig. 6. Modification of D flip-flop for T operation using control input \(T=b\).

Fig. 7. Implementation of \(((a \cdot b) \oplus c)\) in feedback loop with D flip-flop. (a) Logic diagram. (b) Characteristic table. (c) Simplified implementation using T flip-flop. (d) Alternate simplified implementation using T flip-flop.

**III. CMOS T FLIP-FLOP CONSTRUCTION**

Instead of using basic logic gates, like NAND or NOR gates as in case of BJT-based flip-flop realizations, transmission gates are used in CMOS technology to control the connections this extended logic is shown in Fig. 2(b). The same function as that of \(c = (a \cdot b)\) followed by D flip-flop can be performed when \(b\) is used as active-high SET signal of the D flip-flop and \(a\) is fed as the D input, as shown in Fig. 2(c).

3) \((a + \bar{b})\) or \((\bar{a} \cdot b)\) Followed by a D flip-flop: The logic symbol of \(c = (a + \bar{b})\) followed by storage of output bit \(c\) in a D flip-flop is shown in Fig. 3(a). The corresponding state-transition-truth table is shown in Fig. 3(b). The same function as that of \((a + \bar{b})\) followed by D flip-flop can be performed in two other ways as shown in Fig. 3(c) and (d). One of them is derived by feeding \(b\) as active-low RESET and \(a\) as D input (shown in Fig. 3(c)). It can be found that the \(Q^\prime\) output of D flip-flop of Fig. 3(c) is identical to the \(Q\) output of the D flip-flop of Fig. 3(a). \(\bar{a} 

4) \((a \oplus b)\) Followed by a D flip-flop: The logic symbol of \(c = (a \oplus b)\) followed by the storage of bit \(c\) in a D flip-flop is shown in Fig. 4(a). The corresponding state-transition-truth table is shown in Fig. 4(b). \((a \oplus b)\) cannot be incorporated within a D flip-flop alone for the extended logic implementation, but using a NAND gate and a NOR gate at the D input and RESET of the D flip-flop, it would be possible to realize the same function as that of \((a \oplus b)\) followed by D flip-flop as shown in Fig. 4(c), where \((a \oplus b)\) is used as the D input, and \((a \oplus b)\) is used as the RESET.

**B. Sequential Logic Extension with Feedback Loop**

1) \((a \oplus b)\) in Feedback Loop with D flip-flop: The logic symbol of \(c = (a \oplus b)\) in the feedback loop with a D flip-flop is shown in Fig. 5(a). The characteristic table for this extended logic (shown in Fig. 5(b)), is identical to that of a T flip-flop, where \(b\) is taken as input for the T flip-flop. An XOR gate in the feedback loop with a D flip-flop, therefore, could be realized by a T flip-flop with \(b\) as its control input [1]. T flip-flops are traditionally derived from JK flip-flops (for \(J = K\), but a relatively simpler input modification of D flip-flop for T operation with \(T\) as control input can be obtained as shown in Fig. 6. For \(T = 0\), the clock input to the D flip-flop remains fixed at zero so that the state of the flip-flop does not change, while for \(T = 1\) the clock becomes available as usual, and the output toggles on the rising edge of clock. The clock modification in Fig. 6, therefore, makes the D flip-flop behave like a T flip-flop where the \(T\) input is used as a control.

2) \(((a \cdot b) \oplus c)\) in Feedback Loop with D flip-flop: The logic symbol of \(((a \cdot b) \oplus c)\) having the XOR in the feedback loop with a D flip-flop is shown in Fig. 7(a). The characteristic table for this extended logic is shown in Fig. 7(b). A straightforward simplification of this logic could be achieved by replacing the \((a \cdot b)\) in the feedback loop with a D flip-flop with a T flip-flop as shown in Fig. 7(c). Another variation of implementation of this logic which might be more useful in some applications is shown in Fig. 7(d). One of the inputs is fed as control input to the T flip-flop while the other is ANDed with the clock. The output of the flip-flop toggles on arrival of the positive edge of the clock if and only if both the inputs are in state 1. The implementations of Figs. 6 and 7(d) would be very much useful in some specific situations, where a single input-bit is used to control multiple flip-flops. Apart from the area and time saving, it would help in two other ways in such situations. Firstly, an additional wire for the input-bit would no longer be required. Secondly, when the input \(b = 0\), it would perform auto clock-gating for power minimization.
and necessary data movement for the construction of flip-flops. Besides, CMOS flip-flops are typically constructed as D flip-flops since it leads to simpler and efficient implementation through transmission gates [2]. CMOS JK flip-flops are derived from D flip-flops by suitable input modifications. T flip-flops are conventionally derived either from JK (where \( J = K = 1 \)) or from D flip-flop. For conversion of a D flip-flop for T operation, the complement output \( \overline{Q} \) is connected back as input to the D flip-flop. The T flip-flop derived accordingly from a D flip-flop has the same complexity as that of the D flip-flop; and works fine as a clock divider circuit or an oscillator. But, it cannot accept the control input like that of JK flip-flop-based implementation. The T type operation by a JK flip-flop, on the other hand, involves more area and more propagation-delay than the direct realization by a D flip-flop.

A simple input modification of D flip-flop for T operation with T as control input is shown in Fig.6. The T flip-flop construction of Fig.6 is relatively simpler (and may involve less area and time) compared with the JK flip-flop-based implementation. But, its complexity is still higher than that of a D flip-flop due to the additional AND gate on the clock line. For further improvement on using a D flip-flop for T operation, we can modify the clock derivation circuit in the CMOS realization of D flip-flop, instead of using the additional AND gate. The functional schematic of CMOS realization of a D flip-flop is shown in Fig.8(a) and its clock derivation circuit is shown separately in Fig.8(b). The clock derivation circuit requires two inverters to derive a pair of balanced outputs to derive a pair of mutually complementary clock inputs \( c \) and \( cn \). The modified form of a D flip-flop for T operation is shown in Figs.8(c), 8(d) and 8(e). The output \( \overline{Q} \) is taken as the D input of D flip-flop in Fig.8(c), and the clock derivation circuit is controlled by the \( T \) input as shown in Figs.8(d) and 8(e). The circuits in Figs.8(c) and 8(d) or 8(e) result in the same transition table as that of Figs.5 and 6. Besides, this modification of the clock-derivation circuit, involves only a very small increase in the area and propagation delay of D flip-flop, and offers a significant saving over the modification according to Fig.6. The clock derivation of Figs.8(d) or 8(e) could also be used in the D flip-flop of Fig.8(a) to have a controlled-register operation as shown in the Table of Fig.9. The \( D \) input in this case could be transferred to the output state or the output state could be kept unchanged whenever required. This could be used in controlled-registers (or shift-registers) where the register is required to deliver the output (or shift the content) at a specific clock period or under certain conditions through a single-bit control.

**IV. Complexity Considerations**

We estimate and compare here the area-complexities and propagation delays of conventional implementations of combinational logic in feed-forward path and feed-back loop with a D flip-flop, and those of corresponding extended sequential logic. Using the cell areas and propagation delays of different gates and D flip-flops with and without SET and RESET for TSMC 0.18um process 1.8-Volt SAGE-X™ standard cell library [3] for different drive-strengths we have

<table>
<thead>
<tr>
<th>clock</th>
<th>( T )</th>
<th>( D )</th>
<th>( Q )</th>
<th>( Q^+ )</th>
<th>remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>rising</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>( Q^+ \leftarrow D )</td>
</tr>
<tr>
<td>rising</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>1</td>
<td>( Q^+ \leftarrow D )</td>
</tr>
<tr>
<td>rising</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>output unchanged</td>
</tr>
<tr>
<td>rising</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>output unchanged</td>
</tr>
<tr>
<td>not rising</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>output unchanged</td>
</tr>
<tr>
<td>not rising</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>output unchanged</td>
</tr>
</tbody>
</table>

**Fig. 9. Transition Table of modified D flip-flop using the controlled clock derivation circuit of Fig.8(d).**

**TABLE I**

<table>
<thead>
<tr>
<th>drive strength</th>
<th>( \text{AND} + \text{FF} ) area</th>
<th>( \text{extended logic} ) area</th>
<th>( \text{savings} ) area delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>XL</td>
<td>66.53</td>
<td>0.49</td>
<td>56.55</td>
</tr>
<tr>
<td>X1</td>
<td>66.53</td>
<td>0.53</td>
<td>56.55</td>
</tr>
<tr>
<td>X2</td>
<td>79.83</td>
<td>0.50</td>
<td>69.85</td>
</tr>
<tr>
<td>X4</td>
<td>89.81</td>
<td>0.48</td>
<td>83.16</td>
</tr>
</tbody>
</table>
obtained the complexities of the conventional and the proposed extended sequential logic implementations. The complexities of feed-forward extended logic and conventional realizations are compared in Tables I, II and II, while those of XOR within feedback loop are compared in Table IV for comparison. The proposed implementation of OR gate could be assumed to be almost the same as that of equivalent D flip-flop along with a D flip-flop. It is shown in Section III that complexity of proposed CMOS construction of T flip-flop (with active-low RESET) is nearly 11% less area and 23% less delay in average over the conventional implementation using D flip-flop with active-high RESET. But that could be realized by a D flip-flop in place of the input NAND gate of D flip-flop with active-low RESET, e.g., the DFFTR cell of TSMC 0.18um process 1.8-Volt SAGE-X™ standard cell library [3]. Accordingly, we have obtained the areas and the delays of the proposed and the conventional circuits (listed in Table III). Compared with the conventional one, it involves either less or the same area except for drive strength X2 and less delay except for drive strength X1. In average, it involves more than 1% less area and nearly 11% less delay over XOR followed by D flip-flop without RESET.

### B. Complexity of XOR in Feedback Loop with D flip-flop

The proposed implementation of \((a \oplus b)\) in the feedback loop with a D flip-flop, as discussed in Sections II, could be implemented by a T flip-flop. Conventionally, it could be implemented either by a JK flip-flop or by an XOR gate along with a D flip-flop. It is shown in Section III that the complexity of proposed CMOS construction of T flip-flop is almost the same as that of equivalent D flip-flop except that it uses a NAND gate instead of an inverter in the clock derivation circuit. Since the complexity of a NAND gate could be assumed to be almost the same as that of an inverter, the complexity of T flip-flop would accordingly be assumed to be the same as that of a D flip-flop. Based on the area and worst case delays of XOR gate, D flip-flop and JK flip-flop (for TSMC 0.18 micron process) for different drive-strengths we have obtained the complexities of conventional and the proposed implementations as listed in Table IV for comparison. The proposed implementation involves nearly 40% less area and 29% less propagation delay, in average, for different drive-strengths over the conventional implementation using D flip-flop with XOR gate. It involves nearly 11% less area and 23% less delay in average over the conventional implementation using JK flip-flop. The proposed implementation of \(((a \oplus b) \oplus c)\) in feedback loop with a D flip-flop, also has similar advantages over the conventional one, since it has additional complexity of only one AND gate over the implementation of \((a \oplus b)\) in feedback loop with D flip-flop in each of the cases.

### Table IV

<table>
<thead>
<tr>
<th>drive strength</th>
<th>XOR + D flip-flop (1)</th>
<th>JK flip-flop (2)</th>
<th>proposed construction (3)</th>
<th>saving of (3) over (1)</th>
<th>saving of (3) over (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>area</td>
<td>delay</td>
<td>area</td>
<td>delay</td>
<td>area</td>
</tr>
<tr>
<td>XL</td>
<td>106.44</td>
<td>0.75</td>
<td>86.49</td>
<td>0.68</td>
<td>79.83</td>
</tr>
<tr>
<td>X1</td>
<td>106.44</td>
<td>0.75</td>
<td>93.14</td>
<td>0.74</td>
<td>79.83</td>
</tr>
<tr>
<td>X2</td>
<td>119.75</td>
<td>0.76</td>
<td>93.14</td>
<td>0.74</td>
<td>86.49</td>
</tr>
<tr>
<td>X4</td>
<td>159.67</td>
<td>0.66</td>
<td>116.42</td>
<td>0.63</td>
<td>103.12</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>drive strength</th>
<th>NOT+OR+FF</th>
<th>extended logic</th>
<th>savings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>area</td>
<td>delay</td>
<td>area</td>
</tr>
<tr>
<td>XL</td>
<td>73.18</td>
<td>0.52</td>
<td>56.55</td>
</tr>
<tr>
<td>X1</td>
<td>73.18</td>
<td>0.56</td>
<td>56.55</td>
</tr>
<tr>
<td>X2</td>
<td>89.81</td>
<td>0.52</td>
<td>69.85</td>
</tr>
<tr>
<td>X4</td>
<td>106.44</td>
<td>0.50</td>
<td>83.16</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>drive strength</th>
<th>XOR + FF</th>
<th>proposed logic</th>
<th>savings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>area</td>
<td>delay</td>
<td>area</td>
</tr>
<tr>
<td>XL</td>
<td>79.83</td>
<td>0.59</td>
<td>76.51</td>
</tr>
<tr>
<td>X1</td>
<td>79.83</td>
<td>0.55</td>
<td>76.51</td>
</tr>
<tr>
<td>X2</td>
<td>99.79</td>
<td>0.54</td>
<td>103.12</td>
</tr>
<tr>
<td>X4</td>
<td>129.73</td>
<td>0.50</td>
<td>129.73</td>
</tr>
</tbody>
</table>
V. APPLICATIONS OF EXTENDED SEQUENTIAL LOGIC

There could be many possible applications of the extended sequential logic, but to show the advantages of extended logic over the conventional implementation of synchronous digital circuits, we discuss here its application in two simple but popular cases of arithmetic circuits, e.g., finite field multiplier over \(GF(2^m)\) and carry-save adder.

A. Finite Field Multiplication over \(GF(2^m)\)

Multiplication over \(GF(2^m)\) is a basic field operation which is frequently encountered in elliptic curve cryptography (ECC) and error control coding [4], [5]. Multiplication in polynomial basis is relatively simpler, offers scalability for the fields of higher orders, and does not require a basis conversion [6]. A large number of architectures have been proposed in the literature for efficient polynomial basis multiplication over \(GF(2^m)\) in dedicated hardware platforms [7]–[12]. Serial-parallel polynomial-basis multipliers are well-suited for small embedded systems since the cost and size of hardware and bandwidth are major constraints in such systems [7]–[9]. We discuss here, the application of proposed extended sequential logic for efficient implementation of serial-parallel polynomial-basis multiplier for \(GF(2^m)\).

1) Mathematical Formulation: Let the finite field over \(GF(2^m)\) be defined, in general, by an irreducible polynomial of degree \(m\), given by

\[
Q(z) = z^m + q_{m-1}z^{m-1} + \ldots + q_2z^2 + q_1z + 1
\]

where \(\{q_j\} \text{ for } 1 \leq j \leq m - 1 \} \in GF(2)\). The polynomial basis \(\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}\), (where \(\alpha\) is a root of \(Q(z)\)), can be used to represent the field elements, so that any two arbitrary elements \(A\) and \(B\) in \(GF(2^m)\), can be represented in the form of polynomials of degree \((m - 1)\) as

\[
A = \sum_{j=0}^{m-1} a_j \alpha^j \quad \text{and} \quad B = \sum_{j=0}^{m-1} b_j \alpha^j
\]

where, \(a_j\) and \(b_j \in GF(2)\), for \(j = 0, 1, \ldots, m - 1\).

The product of elements \(A\) and \(B\) over \(GF(2^m)\) is given by

\[
C = A \cdot B \mod Q(z) = \sum_{i=0}^{m-1} b_i A^i
\]

where \(A^i = [\alpha^i \cdot A \mod Q(z)]\), \(A^0 = A\), and \(A^{i+1}\) can be obtained from \(A^i\) recursively as:

\[
A^{i+1} = \alpha_i A^i \mod Q(z)
\]

By polynomial expansion of right-side of (4), we can find

\[
A^{i+1} = [a_0^i \alpha + a_1^i \alpha^2 + \ldots + a_{m-2}^i \alpha^{m-1} + a_{m-1}^i \alpha^m] \mod Q(z)
\]

where \(A^i = \sum_{j=0}^{m-1} a_j^i \alpha^j\).

Since \(\alpha\) is a root of \(Q(z)\) given by (1), one can have

\[
\alpha^m = q_{m-1} \cdot \alpha^{m-1} + \ldots + q_2 \cdot \alpha^2 + q_1 \cdot \alpha + 1
\]

Substituting it on (5), \(A^{i+1}\) can be obtained as

\[
A^{i+1} = a_0^{i+1} + a_1^{i+1} \alpha + \ldots + a_{m-1}^{i+1} \alpha^{m-1}
\]

where \(a_j^{i+1} = a_j^{i} \oplus q_j\) for \(1 \leq j \leq m - 1\),

\[
a_0^{i+1} = a_{m-1}^{i}
\]

2) Conventional Implementation of Multiplier over \(GF(2^m)\) and its Optimization: The finite field multiplication could be performed in two stages of recursive operations, where modular reduction is performed according to (7) in the first stage and AND-accumulate operation is performed according to (3) in the second stage. A conventional implementation of serial-parallel multiplier over \(GF(2^m)\) is shown in Fig.10. It consists of a reduction section and an AND-accumulate section. The reduction section consists of \(m\) reduction cells (RC) and \(m\) D flip-flops to perform successive reductions of operand \(A\) in every cycle according to (7). The D flip-flops of the reduction section are initialized

![Fig. 10. Conventional implementation of serial parallel multiplier over \(GF(2^m)\). (a) Structure of the multiplier. (b) Function of \(i\)th reduction cell.](image-url)
by the bits of input word $A$, which get shifted from one D flip-flop to the next in each cycle across the reduction section. Function of each RC is depicted in Fig. 10(b). The $i$th RC consists of an XOR gate if $q_i = 1$, otherwise the RC could be removed. The AND-accumulate section consists of $m$ number of AND-accumulate nodes. One such node is shown in the gray oval. Each AND-accumulate node performs successive AND-accumulate operations (corresponding to the bits $b_j$ of $0 \leq j \leq m - 1$ of input word $B$) according to (3). After $m$ cycles the AND-accumulate section generates the product word $C$ and transfers that to the output register. The $(a b) \oplus c$ operation in feed-back loop with a D flip-flop to be performed by each AND-accumulate node could be implemented efficiently by a T flip-flop where the input $b$ is fed as the control input and the clock input of the T flip-flop is derived by ANDing of the clock with input $a$ as shown in Fig. 7(d) in Section II. The AND-accumulate section of the conventional implementation of bit-serial multiplier (Fig.10) could therefore be implemented by $m$ T flip-flops where the input $b_j$ is ANDed with clock input and fed to T flip-flop as clock as shown in Fig.11.

3) Complexity Considerations: The reduction section is identical in the conventional and optimized implementations of Figs. 10 and 11, respectively. It consists of $m$ D flip-flops and maximum of $(m - 1)$ 2-input XOR gates (if each $q_i$ for $0 \leq i \leq m - 1$ is 1). The AND-accumulate section of Fig.10 consists of $m$ D flip-flops, $m$ 2-input XOR gates and $m$ 2-input AND gates. The AND-accumulate section of Fig.11 on the other hand needs $m$ D flip-flops and one AND gate (used for ANDing with the clock). Both the structures perform a field multiplication over $GF(2^m)$ in $m$ cycles, but they have different duration of clock periods due to the difference in their critical-paths. The critical-path of conventional structure (Fig.10) is $T_{CC} = T_A + T_X + T_{FF}$, where $T_A$, $T_X$ and $T_{FF}$ are the delays of a 2-input AND gate, 2-input XOR gate and D flip-flop, respectively. The critical-path of the optimized structure (Fig.11) is $T_{CO} = \max\{T_X,T_{FF},T_{TF}\}$, where $T_{TF}$ is the delay of a T flip-flop. Since $T_X$ is less than $T_{FF}$ and $T_{TF}$, while $T_{TF}$ could be assumed to be the same as that of $T_{FF}$ as discussed in Section II, we can have $T_{CO} = T_{FF}$.

In Table V, we have listed the hardware requirements of the proposed structure, as well as, the existing structures. All the architectures listed in Table V have the same throughput per cycle and the same latency in terms of number of cycles, but amongst the existing designs, the structure of [8] involves the minimum of area and the minimum cycle time. The proposed conventional structure of Fig.10 involves the same time complexity as the structure of [8] with significantly less number of gates and flip-flops compared with the other. The optimized structure of Fig.11, however, involves nearly (1/3)rd number of gates and has a cycle time of duration nearly half that of proposed conventional design.

The conventional design and the optimized design differ only in the implementation of their AND-accumulate section. The critical-path of both these designs is the same as the critical-path of their respective AND-accumulate section. Moreover, the major part of the gate and register complexities of conventional design is contributed by the AND-accumulate section. The area and the critical-path of AND-accumulate sections of the conventional and the optimized implementations of serial-parallel finite field multiplier are listed in Table VI. The AND-accumulate section of the optimized design is found

![Fig. 11. Proposed implementation of serial parallel multiplier for trinomial-based over $GF(2^m)$ using extended sequential logic.](image-url)
to have less than half of the area and half the cycle time, in average, of the conventional one.

B. Carry-Save Addition

Carry-save adders (CSA) find wide applications as 3-to-2 counters, for addition of partial-products for fast multiplication and pipelined additions [2], [13]. The structure of a 4-bit carry-save adder is shown in Fig.12. It consists of four nodes, one of those nodes being shown by the gray oval in Fig.12. Each node consists of a full-adder (FA) where the sum $S_i$ is fed back as one of the inputs at the next cycle through a D flip-flop of the $i$th node, and the carry $C_i$ is transferred to the next node on its right through a D flip-flop. The critical-path of the CSA is $T_{FA} + T_{FF}$, where $T_{FA}$ and $T_{FF}$ are the propagation delays for a full-adder and a D flip-flop.

1) Optimization of Carry-Save Addition using Extended Sequential Logic: The function of each node could be represented by either of the pair of circuits shown in Figs.13(a) and 13(b). The circuit in Fig.13(a) corresponds to the implementation of full-adder of bits $x$, $y$ and $s$ according to the logic relations [14], [15]:

\[
\text{carry} = \overline{a} + b \quad \text{(8a)}
\]

and

\[
\text{sum} = r \oplus s \quad \text{(8b)}
\]

where $a = r \cdot s$, $b = x \cdot y$ and $r = x \oplus y \quad \text{(8c)}$

Implementation of Fig.13(a) consists of three main blocks enclosed by gray boxes along with a NAND gate. Box 1 consists of an XOR gate in the feedback loop with D flip-flop, which could be implemented by a T flip-flop as discussed in Section II. Box 2 is a half adder and Box 3 has $\pi + b$ followed by a D flip-flop, which could be implemented by a D flip-flop with a RESET as shown in Fig.3(c). The only difference of the circuit of Fig.13(b) from that of Fig.13(a) is in the implementation of the logic expression of $a$ given by

\[
a = (x \cdot s) + (y \cdot s) = (x + y) \cdot s \quad \text{(9)}
\]

Instead of a NAND gate of Fig.13(a), it therefore requires Box 4 for realization of $a$.

2) Complexity of the Carry-Save Adder Optimized by Extended Sequential Logic: The implementation of CSA with nodes of Fig.13(a) has a critical-path $\max\{(T_X + T_{NAND} + T_{FF}), (T_X + T_{FF})\}$ and that of Fig.13(b) is $\max\{(T_{OR} + T_{NAND} + T_{FF}), (T_X + T_{FF})\}$. Circuit of Fig.13(b) leads to smaller critical-path at the cost of one OR gate per node. The area and critical-path of implementations of CSA based on extended sequential logic and the conventional one (for TSMC 0.18 micron process) are listed in Table VII. The proposed designs of Fig.13(a) for CSA nodes using extended sequential logic involves nearly 12% less area and 15% less critical-path, in average, for different drive-strengths over the conventional implementation. The proposed circuit in Fig.13(b) offers a saving of nearly 8.5% area and 19.7% delay, in average, over the conventional one for lower drive-strengths. At drive-strength 4, however, it involves nearly 6.6% more area and 22.2% less critical-path over the other. The proposed design based on Fig.13(a) involves slightly more critical-path but requires less area than that of Fig.13(b). It should, therefore, be preferred if area is to be minimized, while that of Fig.13(b) could be used for time minimization.

VI. CONCLUSIONS

A new approach for extending the sequential logic functionality of D flip-flop is suggested to perform an additional Boolean function simultaneously along with its usual bit-storage function. It is shown that combinational functions of
the form \((a \ast b)\) in a feed-forward path with a D flip-flop could be implemented by a D flip-flop with a RESET or SET provision. Such an integrated implementation is more efficient than the conventional one, due to the fact that, the additional area and delay involved in incorporating SET and RESET in a D flip-flop (due to the optimization of logic blocks inside it) is much less than that of implementing the combinational elements outside the flip-flop. Some other functions like \((a \oplus b)\) and \((a \cdot b)\oplus c\) in the feedback loop with a D flip-flop could be implemented by a T flip-flop by a simple modification of its clock input, where the complexity of XOR gate is saved. The use of extended sequential logic is found to provide a substantial saving in critical-path and area over the conventional implementations. We have also presented a simple approach for the construction of CMOS T flip-flop, which is more efficient than the T flip-flop derived from JK flip-flop. To show the advantage of extended sequential logic, it is used for optimizing a finite field multiplier over \(GF(2^m)\) and carry-save-adder of real numbers. Using the extended logic in case of finite field multiplier, the number of gates is reduced to nearly \((1/3)rd\), and the clock period is reduced to nearly half that of the conventional design. In case of carry-save adder also, the use of extended logic is found to result in a significant saving in area and/or time-complexity over the conventional design. The extended logic provides substantial benefit when the sequential elements are very much interspersed with the combinational elements, and full pipeline is covered with this simple optimization. Particularly when the logic maps into the clock pin, not only it provides area and time saving but can eliminate the wiring for an input bit-line and perform auto clock-gating (by fixing the input at zero) for power minimization. It would be useful to have a family of library cells for the extended sequential logic for optimal realization of synchronous digital systems.

REFERENCES


<table>
<thead>
<tr>
<th>drive strength</th>
<th>conventional design (1)</th>
<th>optimization of Fig.13(a) (2)</th>
<th>optimization of Fig.13(b) (3)</th>
<th>saving of (2) over (1)</th>
<th>saving of (3) over (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>area CP</td>
<td>area CP</td>
<td>area CP</td>
<td>area CP</td>
<td>area CP</td>
</tr>
<tr>
<td>XL</td>
<td>186.28 0.79</td>
<td>159.67 0.66</td>
<td>172.97 0.62</td>
<td>16.67% 19.49%</td>
<td>7.69% 27.40%</td>
</tr>
<tr>
<td>X1</td>
<td>189.60 0.77</td>
<td>162.99 0.69</td>
<td>176.30 0.67</td>
<td>16.33% 11.06%</td>
<td>7.55% 13.82%</td>
</tr>
<tr>
<td>X2</td>
<td>252.81 0.73</td>
<td>216.22 0.66</td>
<td>229.52 0.62</td>
<td>16.92% 10.91%</td>
<td>10.14% 17.78%</td>
</tr>
<tr>
<td>X4</td>
<td>282.74 0.68</td>
<td>282.74 0.58</td>
<td>302.70 0.55</td>
<td>0.00% 17.97%</td>
<td>-6.59% 22.24%</td>
</tr>
</tbody>
</table>

ADDFH, ADDH and DFFTR cells [3] are used for the full adder, half adder and D flip-flops. CP stands for clock period. Area and CP are in sq.µm and nanosecond, respectively.

Pramod Kumar Meher (SM’03) received the first class degrees of B.Sc. (Honours) and M.Sc. in Physics, and Ph.D. in Science from Sambalpur University, Sambalpur, India in 1976, 1978, and 1996, respectively.

He has a wide scientific and technical background covering Physics, Electronics and Computer Engineering. Currently, he is a Senior Fellow in the School of Computer Engineering, Nanyang Technological University, Singapore. He was a Professor at Utkal University, Bhubaneswar, India since 1997-2002, a Reader in Electronics at Berhampur University, Berhampur, India during 1993-1997, and a Lecturer in Physics in various Government Colleges (in India) during 1981-1993. His research interest includes design of dedicated and reconfigurable architectures for computation-intensive algorithms pertaining to signal processing, image processing, secure communication and bioinformatics. He has published more than 100 technical papers in various reputed journals and conference proceedings. Currently, he is serving as Associate Editor of the IEEE Transactions on Circuits and Systems-II: Express Briefs and The Journal of Circuits and Systems for Signal Processing.

Dr. Meher was conferred with the Samanta Chandrasekhar Award for excellence in research in Engineering & Technology for the year 1999. He is a Chartered Engineer of the Engineering Council of United Kingdom, a Senior Member of IEEE, a Fellow of The Institution of Electronics and Telecommunication Engineers (IETE) of India, and a Fellow of the Institution of Engineering and Technology (IET), (formerly known as the Institution of Electrical Engineers), UK.