Scalable Systolic Structure to Realize Arbitrary Reversible Symmetric Functions

Soo-Hong Kim and Sung Choi

Dep’t of Computer Software Engineering, Sangmyung University,
San 98-20, Anseo-dong, Cheonan, Chungnam, 330-180, Korea
E-mail: soohkim@smu.ac.kr

Dep’t of Computer Science, Namseoul University,
21 Maeju-ri, Seounghwan, Cheonan, Chungnam, 330-707, Korea
E-mail: sstar@nsu.ac.kr

Abstract: It is shown that for power not to be dissipated in any circuit, it is necessary that all system circuits must be built from the reversible gates. Reversible circuits which are hierarchically composed of reversible primitives have two types of outputs in general; functionality outputs, and outputs that are needed only to achieve reversibility, being called “garbage”. Minimizing the number of garbage in reversible logic synthesis is very important, since the addition of only one bit of garbage is very expensive or even impossible to implement in most technologies. In this paper we propose a regular and scalable systolic structure to realize arbitrary reversible symmetric functions with less complexity and smaller number of garbage than previously presented paper. Both complexity and garbage are reduced from $O(n^2)$ to $O(n)$ by adopting systolic structure into the reversible logic. The proposed structure can also handle incompletely specified arbitrary symmetric functions.

1. Introduction

As proved by Landauer [1], using traditional irreversible logic gates such as AND or multiplexer type inevitably lead to energy dissipation regardless of the realization technology. Bennett [2] showed that for power not to be dissipated in arbitrary circuits, it is necessary that the circuit be built from reversible gate. In principle, reversible logic gates dissipate negligibly little heat and the use of reversible operations is one of the most attractive new perspectives to build nearly energy lossless ultra-small and super-fast quantum computer. It should be noted that Bennett’s theorem is only a necessary but not sufficient condition. Its extreme importance lies in the technological necessity that every future technology will have to use reversible gates in order to reduce the power.

A circuit is said to be reversible if there is a one-to-one and onto mapping between the vectors of inputs and outputs; thus the vector of inputs can be always reconstructed from the vector of outputs [3]. Thus, the number of outputs in a reversible gate or circuits has to be the same as the number of inputs. Output functions of binary reversible logic gates equal to 1 for exactly half their input assignments are called balanced. Logic design of reversible circuits is quite different from designing conventional irreversible logic circuits. In reversible circuits we have
to use at least one gate to duplicate a signal. Moreover, for realization of non-balanced Boolean functions with a reversible circuit, it is necessary to add constant signals to input of circuits.

A systolic array [4-6] formed by interconnecting a set of identical data-processing cells in a uniform manner is a combination of an algorithm and a circuit that implements it, and is closely related conceptually to arithmetic pipeline. In a systolic array, data words flow from external memory in a rhythmic fashion, passing through many cells before the results emerge from the array's boundary cell and return to external memory. The external memory connected to the systolic array's boundary cell stores both input data and results. Upon receiving data words, each cell performs same operation and transmits the intermediate results and data words to adjacent cells synchronously. The underlying principle of systolic array is to achieve massive parallelism with a minimum communication overhead, and generally speaking, a systolic array is easy to implement because of its regularity and easy to reconfigure because of its modularity.

Little has been published on systematic logic synthesis and optimization methods for reversible and quantum logic. In theory, classical logic synthesis methods can be used, but they generate too many number of gate output signals, making the circuit extremely complex. A good reversible synthesis algorithm should minimize the number of garbage or waste outputs [7]. Shannon and Davio expansions are used for some outputs in fundamental reversible gates of Fredkin and Toffoli, and their generalizations can be used to create new multi-valued and multi-input reversible gates. A regular structure to realize symmetric functions in binary reversible and quantum logic has been presented by Perkowski and Kerntopf [3]. The structure considered by them is not completely regular but semi-regular.

In this paper we propose a scalable systolic structure to realize arbitrary reversible symmetric functions with less complexity and smaller number of garbage than previously presented structure by Perkowski and Kerntopf. The proposed structure can also handle incompletely specified arbitrary symmetric functions and is applicable to arbitrary reversible technologies such as quantum, CMOS, and optical [8].

2. Fundamental Reversible Logic Circuits

Reversible circuits which are hierarchically composed of reversible primitives have two types of outputs in general; functionality outputs, and outputs that are needed only to achieve reversibility, being called “garbage” [7]. Many reversible gates have been proposed as building blocks for reversible logic, and consequently quantum computing. Fig. 1 shows some of the binary \((k, k)\) reversible gates that are commonly used in the reversible logic [9-15].
It is noted from Fig. 1 that while wire(buffer), inverter, and swap gates are naturally reversible, others are not, and thus “garbage” has to be added. There are several (2, 2) gates in reversible logic and they are linear. A gate is linear when all its outputs are linear functions of input variables [3]. The (2, 2) Feynman gate in Fig. 1(d) is also called controlled-not or quantum EXOR or reversible EXOR gate. This gate is one-through gate, which means that one of its input variables is also an output. When \( A=0 \) then \( Q=B \), when \( A=1 \) then \( Q=B' \). With \( B=0 \) the (2, 2) Feynman gate is used as a fanout or copying gate. Every linear reversible function can be built by using only (2, 2) Feynman gates and inverters. Two of the universal (3, 3) reversible gate is Toffoli gate and Fredkin gate. The (3, 3) Toffoli gate is shown in Fig. 1(e). This gate is also called (3, 3) Feynman gate or controlled-controlled-not gate. Toffoli gate is a two-through gate, because two of its inputs are returned unmodified as its output. When \( C=0 \) then \( R=AB \), so AND gate is realized on \( R \) output. When \( A=1 \) then \( R = B \oplus C \), so EXOR gate is realized on \( R \) output. When \( B=1 \) then \( R = A \oplus C \), so again EXOR gate is realized on \( R \) output. The (3, 3) Fredkin gate is shown in Fig. 1(f). In term of classical logic, the gate is just two multiplexers controlled on a flipped (permutated) way from the same control input \( C \). A Fredkin gate is also a one-through gate. The (3, 3) Kerntopf gate [12] is shown in Fig. 1(g). When \( C=0 \) then \( P = A + B \), \( Q = AB \), \( R = B' \), so OR and AND gates are realized on outputs \( P \) and \( Q \), respectively, with \( C \) being the controlling input. When \( C=0 \) then \( P = A'B' \), \( Q = A + B' \), \( R = A \oplus B \),
therefore, for control input 0, the gate realizes NOR, IMPLICATION, and EXOR on its outputs \( P, Q \) and \( R \). Despite the theoretical advantages of Kerntopf gate over classical Fredkin and Toffoli gates, there are no published results on optical or CMOS realization of this gate so far [16].

3. Scalable Systolic Structure for Realization of Reversible Symmetric Functions

The structure proposed in this section allows a realization of arbitrary symmetric functions in a systolic array of reversible gates with little garbage. As discussed in [17], every non-symmetric function can be made symmetric, or “is symmetrizable”.

3.1 Symmetric Function and Symmetrization

It is known in logic synthesis that certain classes of logic functions exhibit specific types of symmetries [18]. Such symmetries include symmetries between functions under negation, symmetries within a logic function under the negation of its variables, and symmetries within a logic function under the permutation of its variables. Accordingly, the following is one possible classification of logic functions:

1. P-equivalence class: a family of identical functions obtained by the operation of permutation of variables.
2. NP-equivalence class: a family of identical functions obtained by the operations of negation or permutation of one or more variables.
3. NPN-equivalence class: a family of identical functions obtained by the operations of negation or permutation of one or more variables, and also negation of function.

A single-index symmetric function, denoted as \( S^k = (x_1, x_2, \ldots, x_n) \) has value 1 when exactly \( k \) of its \( n \) inputs are equal to 1, and exactly \( (n-k) \) of its remaining inputs are 0. Analogously, by \( S^{[i,j,k]} \) we denote the function that is 1 when \( i, j, \) or \( k \) of its variable are equal to 1. Obviously, this notation can be extended to any number of indices, so every symmetric function can be written as \( S^I = (x_1, x_2, \ldots, x_n) \), where \( I \) is any subset of indices \( \{0, 1, 2, \ldots, n\} \). It can be shown that, for \( f = S^A \) and \( g = S^B \), the following are obtained:

\[
\begin{align*}
    f \cdot g &= S^{A \cap B} \\
    f + g &= S^{A \cup B} \\
    f \oplus g &= S^{A \oplus B} \\
    \overline{S} &= S^{\overline{A}}
\end{align*}
\]

The Fig. 2(a) shows symmetric indices of 3-variable binary Boolean function. A symmetric index \( S^I \) specifies a K-map cell that counts value “1” in the specified
minterm $i$ number of times. Fig. 2(b) and Fig. 2(c) show K-map of Boolean function for sum and carry of full adder. These two functions are naturally symmetric functions.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>$c$</th>
<th>$a$</th>
<th>$b$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>$c$</th>
<th>$a$</th>
<th>$b$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(c)

Fig. 2. (a) Symmetric indices of three-input Boolean functions (b) $\text{sum} = a \oplus b \oplus c = S^{13}$

(c) $\text{carry} = ab \oplus be \oplus ca = S^{23}$

It has been shown in [17] that a non-symmetric function can be symmetrized by repeating its variables. This method of variable repetition transforms the values of K-map cells which make the function non-symmetric into don’t cares which make the function symmetric. The following K-map demonstrates the symmetrization by repeating the variables of a non-symmetric Boolean function: $F = a' + b$ [19].

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>$b$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

(b)

Fig. 3. (a) non-symmetric Boolean function, (b) symmetric Boolean function obtained by repeating variable $\{a\}$.

One notes that while in Fig. 3(a) conflicting values occur for symmetry index $S^1$ in minterm $a'b$ and $ab'$, thus producing a non-symmetric function, non-conflicting values are produced for the same non-symmetric function in Fig. 3(b) by repeating variable $\{a\}$.

### 3.2 Programmable Symmetric Function Generator

Fig. 4 shows the programmable $n$-input symmetric function generator which consists of two planes for $n=3$.  

GESTS-Oct.2005
3.3 Systolic Structure

A structure to realize symmetric functions in binary reversible and quantum logic has been presented by Perkowski and Kerntopf. As shown in Fig. 5 the structure considered by them is not completely regular but semi-regular, therefore it is not systolizable. Here our motivation is twofold; first, develop the structure to have a regular structure, and then derive the modular and scalable reversible systolic structure which has a smaller number of garbage than that of Perkowski and Kerntopf.

The proposed structure and corresponding systolic structure consisting of two planes to realize arbitrary reversible symmetric functions are shown in Fig. 6 and Fig. 7, respectively. The structure shown in Fig. 6 is based on symmetric function generator shown in Fig. 5. The structure is simply extended to realize \( n \)-input symmetric functions. The figure shows a realization of three symmetric indices of 4-input symmetric functions as an example. Two symbols of Feynman gate, white square and gray square in Fig. 6 and Fig. 7, are used to make the structure look neat.

The first plane from left in Fig. 6 is a levelized triangular structure to realize all positive unate function. The Feynman gates in the first plane are used to make the first plane has regular structure. The second plane in Fig. 6 is a plane of Feynman
gates that uses their internal EXOR gates to realize every output function as an EXOR of symmetric functions. Horizontal outputs from the first plane are EXOR-ed using Feynman gates in the second plane to create arbitrary symmetric functions at the bottoms. Additional garbage outputs of Kerntopf gates must be forwarded to the primary outputs to satisfy reversibility. The programmability is obtained in the second plane in the form of interconnecting or disconnecting the Feynman gate to generate certain symmetric functions at the bottom of second plane.

The first plane of regular structure in Fig. 6 is used as DG [4], dependency graph, to derive systolic structure for realization of arbitrary reversible symmetric functions. By applying the projection along $ij$-direction and the default schedule, we can obtain the 1-D systolic array as shown in Fig. 7. Our paper focuses on inducing a regular structure which lends itself to systolization on the first plane in Fig. 6. The second plane in Fig. 6 is programmable parts, and is used without modification as shown in Fig. 7.

As shown in Fig. 7 each output is obtained at different cell sequentially and each input is steered inside the corresponding cell. Fredkin gate in each cell is used to route either input from outside or inflow data from the adjacent cell to Kerntopf gate. Control signal of Fredkin gate must be forwarded to primary outputs to guarantee reversibility. Feynman gate between cells is used to produce outputs as a fanout gate.

We compare Perkowski and Kerntopf’s suggestion to our proposal and summarize the result in Table 1, where $n$ is the number of input variables. Our proposal in terms of the total complexity is estimated to be better than that of Perkowski and Kerntopf since the number of the most complex Kerntopf gates in the structure is drastically reduced from $\frac{n(n-1)}{2}$ to $(n-1)$, and an important factor, the number of garbage is definitely cut down from $\frac{n(n-1)}{2} + n$ to $(4n - 2)$ in our approach.

Table 1. Comparison of systolic structure to Perkowski and Kerntopf’s suggestion

<table>
<thead>
<tr>
<th># of gates and garbage bits</th>
<th>Structure of Perkowski and Kerntopf</th>
<th>Our systolic structure</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Kerntopf gates</td>
<td>$\frac{n(n-1)}{2}$</td>
<td>$n-1$</td>
</tr>
<tr>
<td># of Feynman gates</td>
<td>$n$</td>
<td>$n$</td>
</tr>
<tr>
<td># of Fredkin gates</td>
<td>$n-1$</td>
<td>$n-1$</td>
</tr>
<tr>
<td># of delay units</td>
<td>$2(n-1) + \frac{n(n-1)}{2}$</td>
<td>$4n - 2$</td>
</tr>
<tr>
<td># of garbage bits</td>
<td>$\frac{n(n-1)}{2} + n$</td>
<td>$4n - 2$</td>
</tr>
</tbody>
</table>
Fig. 5. Structure for realization of arbitrary reversible symmetric function proposed by Perkowski and Kerntopf.

Fig. 6. A regular structure for realization of arbitrary reversible symmetric function
4. Conclusions

In this paper we propose regular and scalable systolic structure for realization of arbitrary reversible symmetric functions with less complexity and smaller number of garbage than structure proposed by Perkowski and Kerntopf. Both complexity and garbage are reduced from $O(n^2)$ to $O(n)$ by adopting systolic structure into the reversible logic. The structure presented here is technology independent and can be thus used in association with any known or future reversible technology.

We believe that the proposed regular and scalable reversible systolic structure is good for reversible logic synthesis because it is easy to re-use its components, and to extend the structure.

References


Scalable Systolic Structure to Realize Arbitrary


©GESTS-Oct.2005