Cost Efficient Design Approach for Reversible Programmable Logic Arrays

Md. Riazur Rahman

(Department of Computer Science and Engineering, Daffodil International University, Bangladesh)

ABSTRACT: Reversible programmable logic arrays (PLA) are at the heart of designing efficient low power computers. This paper presents an efficient approach to design Reversible PLAs that maximizes the usability of garbage outputs and also reduces the number of ancilla inputs generated. The design for proposed essential components and the architecture of reversible grid network for designing AND and EX-OR planes are also presented. Several algorithms have been proposed and presented to describe the programming interfaces in context of Reversible PLAs construction. Lastly, recent result on the trade-off between cost factors of standard benchmark circuits shows that the proposed design clearly outperforms the existing ones in terms of various cost factors.

Keywords: Reversible Logic Circuits, Quantum Computing, Low Power Computing.

I. INTRODUCTION

During execution of every single instruction, stuff wastes $kTln(2)$ joule of energy that converted into heat due to per bit erase and reload where $l$ is the operating temperature and $k$ is the Boltzmann constant [1]. Solution to such input energy loss mechanism after publishing the tremendous approach called Reversible Computation was introduced by Bennett [2] in 1973. It opens the tunnel of designing robust architecture of low power consumption where total input energy loss is zero, supports the behavior of optical computing, quantum computing, etc. Generic prototype of designing low power programmable devices [3] has obtained popularity in recent days. So, the development of reversible PLAs would be another application that enhances capability of low power computing. Proposed idea presents the novel architecture of PLAs in reversible computing by attaining 100% use of every logical units/gates that propagate all primary inputs to outputs. Proposed architecture reflects the following ideology:

- Maximize the usability of primary input signals
- Avoiding any type of EX-OR operations in AND plane
- Reduce number of garbage outputs and ancilla inputs

Rest of the paper has been organized as: section II has described reversible logic and the standards of measuring the performance of reversible circuits. Section III has presented the details of proposed gates and demonstrate organizational placement of logical units (gates) in Reversible PLAs grid. Section IV has illustrated the corresponding algorithms for constructing AND and EX-OR planes using reversible {UMGand UNG} and CNOT gates, respectively. Comparative performance analysis based on benchmark standard circuits has been showed in section V. Finally, section VI has concluded this paper with the summary and future directions. The contents of each section may be provided to understand easily about the paper.

II. LITERATURE REVIEW

2.1 Reversible Gates

Bidirectional or irreversible circuit prevents input loss due to unique mapping between input and output states. Like classical computing, any reversible operational unit identity is called n x n, i.e. any reversible gate contains:

- n-input lines and n-output lines
- Unique mapping between input and output states
For example, controlled NOT (CNOT), widely known as Feynman gate [4] is reversible has two inputs (a, b) and two outputs (p, q) is shown in Fig. 1. Total number of input and output states are same (i.e. 4) and the mapping between input and output states is unique or vice versa.

There are many reversible gates have been populated based on conservative logic [5], universality of reversible circuit [6], fault tolerant mechanism [7], online testability [8], programmable devices [9], etc. Several reversible gates are self-reflexive, backs primary input signals by attaching self-copy and other gates stuck signals need extra circuitry to return in its initial state. In this paper, two new 3x3 reversible gates called Universal MUX (UMG) and Universal NOR (UNG) are used to design AND plane of reversible PLAs where UNG is self-reflexive reversible gate performs basic OR (or universal NOR) operation and returns primary inputs to output. On the other hand, UMG also performs AND operation and returns primary inputs as like UNG but not self-reflexive.

Fig. 1: Reversible CNOT and unique I/O states mapping

2.2 Performance Measurement Standards
Operational Competency of any circuit is always related to its technical design encroachment. In any particular technology, greater number of logical units slows down the strength of signal hampers net processing speed of circuit. But interestingly, logical minimization provides better opportunity to reduce the number of operational units and total cost.

2.2.3 Total Number of Gates:
In reversible circuit, the input loss is zero in ideal state but bending input signals to output lines absorbs energy and declines the strength of internal signal due to having unavoidable resistance. The total number of gates used in circuits is considered one of the worthycost factor that controls performance of digital circuits [10].

2.2.4 Quantum Cost:
Every reversible circuit point’s unique singular unitary matrix which can be accomplished with one or more 2x2 and 4x4 unitary matrices which are also compatible to 1x1 and 2x2 basic primitives in Quantum Computing. Alternatively, the n-dimensional quantum primitive is identically formed of 2n x 2n dimensional unitary matrix. The total numbers of 2x2 quantum primitives are used to realize any reversible circuit is called Quantum Cost [11].

2.2.5 Garbage Output and Ancilla Input:
Unlike classical computing, reversible circuit requires extra output lines to map all input the states uniquely, called garbage output [12]. On the other hand, one or more input line(s) get saturated in constant level (i.e. 0 or 1) to perform specific operations is called Ancilla Input [13]. According to above definitions, the realization of 2-input EX-OR operation requires only one 2x2 reversible Feynman gate and the quantum cost of Feynman gate is 1 (single 2x2 quantum XOR gate is able to realize CNOT operation), the number of garbage output is 1 and finally, the number of ancilla input is zero.

2.3 Review on Reversible PLAs
In 2006, author of [14] has proposed the Reversible architecture of PLAs that was similar to classical PLA design [3] where AND plane consists of vertical complement and non-complement input lines and horizontal products lines spread over EX-OR plane. Toffoli gates were used to perform AND operation in AND plane whereas Feynman performed EX-OR operation in EX-OR plane. Additionally, Feynman gates were also used in AND plane for avoiding fan-out(s). The improved design of [14] was proposed in [15] brought prominent modification in the basic architecture of classical reversible PLA circuits by using only single line for each input literal in AND plane. Ref. [15] used MUX and Feynman gates to realize improved design of reversible PLAs.
where AND plane also performed copy operation by using Feynman gates as the similar way in [14]. Both papers had used multiple output functions $F$ (i.e. Eqn. 1) as a sample to represent their proposed designs and minimization methodologies.

$$F = \begin{cases} 
    f_1 = ab' \oplus ab'c \\
    f_2 = ac \oplus a'b'c \\
    f_3 = ab' \oplus ab'c \oplus bc' \\
    f_4 = ab \\
    f_5 = ab' \oplus ac \oplus bc' 
\end{cases}$$  \hspace{0.5cm} (1)

2.4 Motivation of this Research Work

Fundamentally, classical architecture of PLAs [3] was implemented by placing configurable switches at cross-point. These switches copy input signal multiple times increases fan-outs (which is restricted in Reversible Computing). The simplicity of AND, OR and NOT logic has been promoted by novel researchers to design such architecture of Programmable Logic Devices (PLD) in classical digital circuit [3]. But in reversible computing, the operability of basic bidirectional components is unavoidable when designing logic circuit such as Reversible Programmable Logic Arrays. In both [14] and [15], reversible PLAs focused the ideal zero energy dissipation due to use of large number of CNOT gates to recover fan-out(s) increases number of ancilla bits and garbage outputs. The idea of proposed research work comes through the reusability of garbage outputs as the input to next operational unit(s) that reduces the number of ancilla input at the same time.

III. PROPOSED REVERSIBLE GATES AND PL ARRAY

In this section, two new reversible primitives (Universal MUX and Universal NOR gates) have been introduced followed by the demonstration of logical units placement in AND plane as well as the ordering principle of products generation.

3.1 New Reversible Gates and Operational Templates

3.1.1 Reversible Universal MUX Gate (UMG):

The input and output vectors of Universal MUX gate can be written as $(a, b, c)$ and $(p = a, q = b \oplus c, r = ab' \oplus a'c)$ respectively. The equivalent quantum representation of UMG is shown in Fig. 2.

![UMG Diagram](image)

Fig. 2: Reversible Universal Multiplexer gate (UMG): (a) Block diagram of UMG and (b) Quantum realization of UMG; Truth table of UMG maps uniquely all the input states to output states

3.1.2 Reversible Universal NOR Gate (UNG):

In Boolean logic, NOR gate is a universal primitives can interpret the functionalities of all basic gates (AND, OR, NOT). Similarly, the input and output vectors of proposed 3x3 Universal NOR gate which perform NOR operation can be written as $(a, b, c)$ and $(p = a, q = b, r = (a + b) \oplus c)$ respectively (shown in Fig. 3).

![UNG Diagram](image)

Fig. 3: Reversible Universal NOR gate (UNG): (a) Block diagram of UNG and (b) Quantum cost is 5; UNG maps input and output states uniquely
3.1.3 Operational Procedures of Proposed Gates:

Proposed UMG and UNG gates are used to perform AND operation of two literals (or a literal and a product). The forms of logical unit(s) which are used in proposed reversible PLAs have been selected based on following facts:

- Best orientation of Input and/or Output line(s)
- Projected output(s)(product/sum) of plane(AND/OR)

UMG performs MUX operation by setting input a as selection line and others (bandc) as data. Proposed UMG is able to generate three min terms of two inputs (\(ab\), \(ab'\) and \(a'b\)) are represented through templates \(\alpha\), \(\beta\) and \(\gamma\) (swapping orientations are \(\alpha', \beta',\) and \(\gamma'\)) as shown in Fig. 4. UMG doesn’t erase the input value of any operational unit while performing AND operation and those unused outputs can be used as the primary inputs to another reversible gate. UNG covers the limitation of UMG and the operational template is symbolized using \(\pi\) (shown in Fig. 5).

Algorithm 1 shows the methodology of selecting template (oriented form of logical unit) to perform AND operation of inputs \(p\) and \(q\) depending on the value of swapFlag. The statement, \(\text{swapFlag} = 0\) indicates to perform AND operation by using \(\alpha', \beta'\) or \(\gamma'\) (otherwise \(\alpha, \beta, \gamma\) or \(\pi\)).

![Diagram of Proposed templates of Universal MUX gate (UMG) that are used in proposed Reversible PLAs design](image1)

![Diagram of Proposed templates of Universal NOR Gate (UNG) those are used in proposed Reversible PLAs design](image2)

Algorithm 1: \(\text{OpAND}(p,q,\text{swapFlag})\)

Templates \(\{\alpha, \beta, \gamma, \pi\}\) (for swapping \(\{\alpha', \beta', \gamma', \pi'\}\)) are used to AND \(\{p, q\}\) based on the value of \(\text{swapFlag}\).

**Start**
1. If \(p\) is a literal in complemented form Then
2. If \(q\) is complemented form Then
3. If \(\text{swapFlag} = 0\) Then \(\pi\) else \(\pi'\)
4. End If
5. Else
6. If \(\text{swapFlag} = 0\) Then \(\beta\) else \(\beta'\)
7. End If
8. End If
9. Else
10. If \(q\) is complemented form Then
11. If \(\text{swapFlag} = 0\) Then \(\alpha\) else \(\alpha'\)
12. End If
13. Else
14. If \(\text{swapFlag} = 0\) Then \(\gamma\) else \(\gamma'\)
15. End If

---

![Footer: www.ajer.org](image3)
On the other hand, the proposed EX-OR plane consists of only Feynman gates used to perform XORing products. Three templates of Feynman gates have been used in proposed design are symbolized through symbols $\Delta$, $\lambda$, and $\nabla$ performing NOT{$a, a'}$, EX-OR {$(a \oplus b)$} and COPY operation {$a, a$}, respectively (shown in Fig. 6).

Fig. 6: Templates of CNOT gate are used to design EX-OR plane

### 3.2 Reversible PLAs Grid and Primitives Placement

Reversible gates are more powerful performing multiple logic operations in a single cycle [10]. The orientation of input and product lines of proposed AND plane is pointed through solid lines (shown in Fig. 7a) where dotted lines indicate another pathway to swap input signals (shown in Fig. 7b) according to the following algorithm (Algorithm 2).

**Algorithm 2: SwapLiterals($L_i, L_j$)**

Exchanging input signals {$I_i, I_j$} in lines, {$L_i, L_j$}

Start
1. Set $a :=$ signal at input line, $L_i$
2. Set $b :=$ signal at input line, $L_j$
3. $L_i := b$ and $L_j := a$

End

Basically, swap operation of two literals be performed when the uses of any literal got ended for doing AND operation in AND plane. SwapLiterals ($L_i, L_j$) moves unused literals from left to right vertical tracks of AND Plane. Performing AND operation at any cross-point of two vertical lines binds single horizontal line to generate cumulative product and again, connecting another literals to cumulative product (if needed) to generate final product of AND plane.

Fig. 7: Reversible PLAs Grid Architecture of AND plane: (a) Primary inputs and generated products passed through vertical and horizontal lines, respectively and (b) Swapping inputs

Ordering products takes crucial role to reduce the cost of product generation by using garbage output ($s$). Also, the usability of different templates provides mining opportunities for optimizing the cost in physical layer. Resultant products contain a number of literals placed according to the order of input literals. For example, products ($P_s$) consist of literals {a, b and c} be produced in order, product(s) start with a followed by start with b followed by start with c. Algorithm 3 describes the methodology of placing products based on their usability. For example, product $abcd$ be generated before $abc$, $abd$ or $ab$ which are consisted of less number of literals.
Algorithm 3: OrderingProducts($I_v, P_v$)

Products, $P_v$, be ordered according to inputs, $I_v$.

Start
1. Set $P_Q := \emptyset$ [is used to store products]
2. Sort $P_v$ based on SizeOf($P_i$) in descending order
3. For $i = 1$ to totalLiterals
4. For $j = 1$ to totalProducts
5. If $I_i \in P_j$ and $P_j \notin P_Q$ then Add $P_j$ to $P_Q$
6. End If
7. End Loop
8. End Loop
9. Set $P_v := P_Q$
End

According to above algorithm (ALG. 3), the order of products consists of inputs $a, b, c, d$ and $e$ (only the non-complemented forms) can be graphed as shown in Fig. 8.

![Fig. 8](image)

Fig. 8: The order of products consist of literals (a, b, c, d, e) is: \{[abcde, abce, abde, abe], [abcd], [abc, abd], [ab], [acde, ace], [acd], [ac], [ade], [ad], [ae], [bcde, bce], [bcd], [bc], [bde], [bd], [be], [cde], [cd], [ce], [de]\}. Products in the same group (for example [abcde, abce, abde, abe]) are independent can be generated in any order.

IV. PROPOSED DESIGN OF AND AND OR PLANES

In this section, proposed design of AND plane based has been described followed by the realization of EX-OR plane.

4.1 Designing Reversible AND and EX-OR Planes

AND plane dominates the performance and cost factors of reversible PLAs where every AND operation rises all cost factors compared to simple EX-OR operation. Algorithm 4 presents the construction of proposed AND plane as well as the minimization of garbage outputs. The construction of AND plane includes, the ordering of products (Algorithm 3) followed by counting $\text{swapFlag}$ and then invoking $\text{OpAND}(p, q, \text{swapFlag})$ (Algorithm 1). Input lines exchange signals (as Algorithm 2) after finishing the generation of all mutual products. Finally, Queue($P_QG$) stores unused garbage products which are reused in the forward product when they get similar to unexplored products.

Algorithm 4: ConstructANDPlane($I_v, P_v$)

This function constructs AND plane by taking inputs of literal ($I_v$) and generates products ($P_v$) connecting multiple input lines ($I_v$) by using UMG and UNG gates.

Start
1. OrderingProducts($I_v, P_v$)
2. Set $P_Q := \emptyset$ and $\text{n\_dot} := 0$ [$P_Q$ stores garbage]
3. For $g = 1$ to totalLiterals
4. For $h = g + 1$ to totalLiterals
5. Set $\text{swapFlag} := 0$
6. For $i = 1$ to totalProducts
7. If $I_i \in P_j$ Then $\text{swapFlag} := \text{swapFlag} + 1$
8. End If
End
Theorem 1. Let, $n$ be the number of AND operations of $m$ output functions, and $t$ be the number of AND operations of garbage outputs $(P_{OG})$ which are identical to any product, then the minimum number of reversible gates to realize AND plane is $(n-t)$, the quantum cost is $5(n-t)$, and the number of ancilla inputs is $(n-t)$.

**Proof:** As performing every reversible AND operation needs a single UMG or UNG gate, results total number of gates to realize AND plane is $n$. But reusability of garbage reduces the number of acting AND operations to $(n-t)$. Similarly, the quantum cost of UMG or UNG is 5 sums up the total quantum cost of circuit is $5(n-t)$ and every reversible AND operation requires an ancilla bit summarizing total number of ancilla inputs to $(n-t)$.

From multi-output function $F$ in Eqn. (1), the total number of AND operations $(n) = 7$, then the number of AND operation(s) in garbage which are similar to any product $(i)$ is 1. So, the number of gates $= (n-t) = 7 - 1 = 6$, quantum cost $= 30$ and total ancilla input $= 6$ (shown in Fig. 9).

![Fig. 9: Optimized version of reversible PLAs of multi-output function $F$ in Eqn. (1)](image)

Theorem 2. Let, $p$ be the number of products (consist of more than two literals) of $m$ output functions, $q$ be the number of garbage outputs which are identical to any products and $ndot$ be the number of cross-point then the number of garbage is $sp + totalLiterals - ndot - q$. 

9. End Loop
10. If swapFlag $>$ 0 Then
11. For $i = 1$ to totalProducts
12. If SizeOf($P_i$) $>$ 1 Then
13. If $P_i \in P_{OG}$ Then Remove $P_i$ from $P_{OG}$
14. Else
15. If $I_g, I_h \in P_i$ Then swapFlag $:= $ swapFlag - 1
16. Set pivotP $:= \text{OpAND}(I_g, I_h, \text{swapFlag})$
17. If SizeOf($P_i$) $>$ 2 Then
18. For $k = h + 1$ to totalLiterals
19. If $I_k \in P_i$ Then
20. Set $P_G := \text{OpAND}(\text{pivotP}, I_k)
21. \text{pivotP} := \text{OpAND}(\text{pivotP}, false)
22. End If
23. End Loop
24. Add $P_G$ to $P_{OG}$ [Add new garbage to $P_{OG}$]
25. End If
26. End If
27. End If
28. Else ndot $:= ndot + 1$ [Use via (.)]
29. End If
30. End Loop
31. Else SwapLiterals($L_G, L_h$) [No mutual products]
32. End If
33. End Loop
34. End Loop
End
Algorithm 5: Describe the construction of EX-OR plane by using Feynman gates where $\lambda$ connects product lines $P_i$ to corresponding function lines $F_j$ produces output signals and another identical copy of $P_i$.

**Algorithm 5:** Construct XOR Plane ($P_v, F_v$)

EX-OR plane generates the final output of multi-output function, $(F_v)$ consists of products $P_v$.

Start
1. Set $F_Q := \emptyset$ and $xdot := 0$
2. For $i = 1$ to total products
3. For $j = 1$ to total functions
4. If $P_i \in F_j$ Then
5. If $F_i \notin F_Q$ Then
6. $\text{FreqOf}(P_i) == 1$ Then
7. $xdot := xdot + 1$
8. Else
9. Use $\nabla$ [Keep a copy of $P_i$]
10. Set $\text{FreqOf}(P_i) := \text{FreqOf}(P_i) - 1$
11. End If
12. Add $P_i$ to $F_Q$
13. Else Use $\lambda$ [XORing $P_i$ to $F_j$ line]
14. End If
15. Else $\text{IF}_i \in F_j$ Then
16. Use $\Delta$ [Keep a copy of $P_i$]
17. End If
18. End Loop
19. End Loop
End

**Theorem 3.** Let $n$ be the number of EX-OR operations of $m$ output functions and $xdot$ be the number of cross-points, then the minimum number of Feynman gates to realize EX-OR plane is $n^2 + m - xdot$, total number of ancilla inputs is $m - xdot$.

According to the proposed algorithms (ALG. 4 & 5), the construction of multi-output function $F$ in Equation (1) is shown in Fig. 9, where garbage outputs are represented by lines ending with a box. Table 1 summarizes that the proposed design of reversible PLAs requires less number of gates, garbage outputs and ancilla inputs as well as minimum quantum cost compared to existing design [15].

<table>
<thead>
<tr>
<th>RPLAs Design</th>
<th>Total Gates (GA)</th>
<th>Garbage (GB)</th>
<th>Ancilla Input (AI)</th>
<th>Quantum Cost (QC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>13</td>
<td>5</td>
<td>7</td>
<td>37</td>
</tr>
<tr>
<td>Existing [15]</td>
<td>18</td>
<td>10</td>
<td>12</td>
<td>39</td>
</tr>
</tbody>
</table>

V. PERFORMANCE ANALYSIS

The realization of benchmark circuits analysis is based on proposed algorithms by using programming language Java (jdk1.7) on NetBeans IDE (8.0) in Windows 7 Workstation is presented in Table 2. All the experiment results are tested on Intel (R) Core(TM) i3 CPU @ 3.30GHz with 2GB RAM. Table 2 shows the experimental results for different benchmark functions and the comparison with the existing method [15] where the required number of gates, garbage outputs and ancilla inputs are minimized in a notable manner. Finally, the trade-off between quantum cost and other factors summarizes the better optimization of proposed design of reversible PLAs is presented.
Table 2: Experimental results using different benchmark functions

<table>
<thead>
<tr>
<th>RPLAs</th>
<th>Total Gates (GA)</th>
<th>Garbage (GB)</th>
<th>Ancilla Input (AI)</th>
<th>Quantum Cost (QC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing</td>
<td>Proposed</td>
<td>Existing</td>
<td>Proposed</td>
</tr>
<tr>
<td>5xp1</td>
<td>166</td>
<td>132</td>
<td>112</td>
<td>44</td>
</tr>
<tr>
<td>9sym</td>
<td>427</td>
<td>377</td>
<td>385</td>
<td>111</td>
</tr>
<tr>
<td>adr3</td>
<td>67</td>
<td>52</td>
<td>48</td>
<td>26</td>
</tr>
<tr>
<td>apex3</td>
<td>3998</td>
<td>3719</td>
<td>1799</td>
<td>520</td>
</tr>
<tr>
<td>b12</td>
<td>159</td>
<td>127</td>
<td>132</td>
<td>56</td>
</tr>
<tr>
<td>bw</td>
<td>305</td>
<td>281</td>
<td>64</td>
<td>30</td>
</tr>
<tr>
<td>cordic</td>
<td>12162</td>
<td>11389</td>
<td>10640</td>
<td>1573</td>
</tr>
<tr>
<td>duke2</td>
<td>941</td>
<td>874</td>
<td>667</td>
<td>157</td>
</tr>
<tr>
<td>e64</td>
<td>2170</td>
<td>2074</td>
<td>2148</td>
<td>121</td>
</tr>
<tr>
<td>inc4</td>
<td>16</td>
<td>11</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>inc5</td>
<td>23</td>
<td>16</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>misex1</td>
<td>88</td>
<td>74</td>
<td>51</td>
<td>25</td>
</tr>
<tr>
<td>misex2</td>
<td>199</td>
<td>165</td>
<td>176</td>
<td>79</td>
</tr>
<tr>
<td>pdc</td>
<td>3801</td>
<td>3451</td>
<td>3006</td>
<td>487</td>
</tr>
<tr>
<td>rd53</td>
<td>56</td>
<td>44</td>
<td>42</td>
<td>23</td>
</tr>
<tr>
<td>rd73</td>
<td>187</td>
<td>150</td>
<td>141</td>
<td>68</td>
</tr>
<tr>
<td>rd84</td>
<td>328</td>
<td>267</td>
<td>265</td>
<td>114</td>
</tr>
<tr>
<td>sasao</td>
<td>14</td>
<td>9</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td>sao2</td>
<td>284</td>
<td>257</td>
<td>236</td>
<td>60</td>
</tr>
<tr>
<td>t481</td>
<td>49</td>
<td>40</td>
<td>53</td>
<td>36</td>
</tr>
<tr>
<td>table3</td>
<td>2602</td>
<td>2439</td>
<td>1814</td>
<td>342</td>
</tr>
<tr>
<td>table5</td>
<td>2539</td>
<td>2384</td>
<td>1819</td>
<td>319</td>
</tr>
<tr>
<td>xor5</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>z5xp1</td>
<td>167</td>
<td>139</td>
<td>114</td>
<td>50</td>
</tr>
<tr>
<td>z9sym</td>
<td>427</td>
<td>377</td>
<td>385</td>
<td>111</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

Reversible computing is in high demands due to the increasing thirst for low power computation. The proposed design increases the reusability of garbage outputs which in turn enhances zero-energy dissipation which is the primary concern for reversible computing. This work has introduced a novel approach to design reversible PLAs by proposing reversible PLAs grid and algorithms to construct AND and EX-OR planes with minimized gates and other cost factors which lead towards the advancement of reversible PLAs which will promote the development of Reversible Field Programmable Logic Arrays in the near future[9].

References