River PLAs: A Regular Circuit Structure

Fan Mo and Robert K. Brayton
Dept. of EECS, University of California, Berkeley
{fanmo,brayton}@eecs.berkeley.edu

ABSTRACT
A regular circuit structure called a River PLA and its re-configurable version, Glacier PLA, are presented. River PLAs provide greater regularity than circuits implemented with standard-cells. Conventional optimization stages such as technology mapping, placement and routing are eliminated. These two features make the River PLA a highly predictable structure. Glacier PLAs can be an alternative to FPGAs, but with a simpler and more efficient design methodology.

Categories and Subject Descriptors
B.6.3 [**Logic Design**]: Design Aids — Automatic synthesis.

General Terms
Algorithms.

Keywords
Programmable Logic Array, River routing.

1. INTRODUCTION

In modern VLSI design, regular circuit structures are becoming preferable for two reasons. First, from layout to mask, manufacturers transform the geometric patterns to achieve better exposure and imaging quality [8], and for this, regular structures are preferable, because they have fewer layout patterns. The second more important reason is that regular structures make the prediction of circuit characteristics easier. Standard-cells have a somewhat regular structure, however its regularity only lies in the row arrangement. Routing results in numerous layout patterns. The synthesis, technology mapping, placement and routing flow associated with the standard-cell designs has the so-called timing-closure problem, the main cause of which is lack of predictability at early synthesis stages.

Programmable Logic Arrays (PLAs) are a regular structure. The design methodology for PLAs is quite mature [7]; it is a sum-of-products (SOP) minimization, which is a basis for modern logic synthesis. The result of a SOP minimization can be mapped directly to a PLA. Technology mapping, required in standard-cell designs, is not necessary; neither are placement and routing for a single-PLA implementation. Another advantage of the PLA is that its design flow is library free. With the rapid shrinking of feature sizes, design methodologies based on library cells suffer from a scaling problem; changes in design rules may require a complete re-characterization of the library cells, which is very costly.

However, single-PLA structures are relatively weak in expressing logic functions. Thus multi-level logic has become the mainstream synthesis technique [6]. In multi-level logic minimization, the entire circuit is represented as a network of nodes where each node is a SOP logic function (a Boolean network). A common approach is to optimize both the entire network as well as each node, and then transform the circuit to a network of library cells via technology mapping. A natural step is to build a network of PLAs (NPLA) from the minimized network without technology mapping [10]. Thus some desirable features of single-PLAs such as technology-mapping free synthesis are preserved. However, NPLAs require placement and routing and the placement of PLAs is not at the gate but at the block level. So far, block level placement is not as well developed as gate level placement. Also, even though each PLA in the NPLA is regular, global regularity is low because of placement and routing (this can also require additional metal layers [10]). In fact, global regularity of the NPLAs may be even worse than for standard-cells, which have a row structure at least.

To eliminate global irregularity of NPLAs caused by placement and routing, a multiple-PLA structure called River PLA (RPLA) is proposed. Given a Boolean network, the nodes are clustered into a stack of PLAs. In each PLA, the outputs and some of the input signals are ordered and fed to the next PLA via river routing. The advantages of RPLAs are:

1. Since the RPLA is a stacked structure and the river routing is quite regular, the RPLA has both local and global regularity.
2. The output buffer and the input buffer of the next PLA in the stack are merged, reducing area and improving speed.
3. The RPLA needs only two metal layers.
4. The design methodology for RPLAs maintains the advantages of the synthesis for NPLAs, but no placement and routing are necessary.
5. Since RPLAs have dedicated buffers, which only drive a known number of loads, and river routing has fixed shapes and involves short distances, the synthesizer is immune to buffering problems. The computation of the total delay becomes a simple summation of the delays of all the PLAs.

Thus, the RPLA is a good alternative to standard-cells. For future Deep Sub-Micron (DSM) designs in which regularity may be a key issue, the regularity of RPLAs should be an advantage.

A programmable version of RPLAs is called a Glacier PLA (GPLA), targeting re-configurable applications, where a block of hardware is embedded in a system-on-a-chip, and the functionality of the block is re-configured whenever the application changes. GPLAs, unlike Field Programmable Logic Array (FPGA), have fixed wiring; thus they need no routing. Although FPGAs have a regular array structure, the topology of a routed net can be irregular, which is hard to predict during synthesis. The granularity of GPLAs is coarser than FPGAs, so to implement a chain of logic functions, the buffers involved in the GPLAs are

DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.

less than in FPGAs. One may argue that the finer granularity of FPGAs leads to higher area utilization. Although true, a portion of the logic resources in FPGAs are wasted for either routability or performance considerations. In contrast, GPLAs waste area only because no appropriate logic can fill in. The design methodology for the FPGAs is similar to that for the standard-cells, except the technology mapping targets a LUT structure. Therefore, they also have problems with buffering and routing prediction.

The remainder of this paper is organized as follows. The circuit structure is described in Section 2, and area and delay formulations given in Section 3. The design methodologies are discussed in Section 4. Experimental results are presented and discussed in Section 5, and Section 6 concludes.

2. STRUCTURE Of RPLAs AND GPLAs

As shown in Figure 1, a RPLA consists of a stack of multiple-output PLAs. Since some of the nets feed more than one PLA in the stack, the output signals of a PLA may come from both the OR-plane (defined as OR-signals) and the AND-plane (defined as AND-through-signals). Those nets ending at the current PLA are called AND-only-signals. The AND-through-signals and the OR-signals of each PLA are ordered and connected to the next PLA in the stack via river routing. Metal layer one and two are assigned to the routing of the AND-though and OR signals respectively. The interconnections via river routing are local and hence short. The left sides of the PLAs are aligned. The primary inputs come in at the bottom. The primary outputs can be retrieved from the right side from anyone of the PLAs or the top. When they are derived at the right side, river routing (still on metal levels one and two) is used to bring the outputs of the PLA OR-planes to the right boundary.

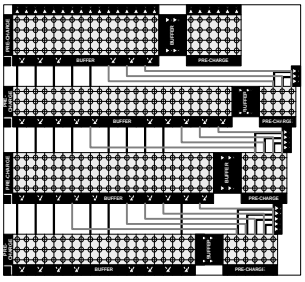

Figure 1. Circuit structure.

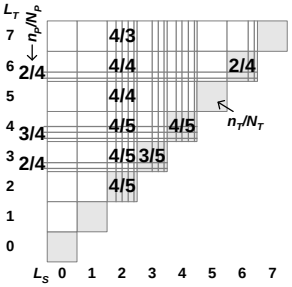

Figure 2. Provide/consume diagram.

A RPLA structure can be characterized by the following parameters: n_L is the number of the levels. $n_T(s,t)$ is a $(n_L+1)\times(n_L+1)$ matrix, the number of the nets starting at level s and reaching level t, where $s\le t$. The primary input nets have starting level of 0. $n_T(t,t)$ denotes the number of outputs of the PLA on level t. It is conceivable that $n_T(s,t)\ge n_T(s,t')$ for $t\ge t'$, because with increasing level, part of the nets coming from lower levels are no longer used by any nodes, while the nodes on the higher levels create their own outputs, which become the new nets. $n_I(t)$ is the number of inputs to the PLA on level t, which is equal to $\sum_{s=0}^{t-1} n_T(s,t)$. $n_O(t)$ is the number of outputs of the PLA on level t, which is exactly $n_T(t,t)$. $n_P(t)$ is the number of product terms on level t. For the GPLA, limits exist for each of the parameters. We use capital letters to define the limits: N_L is the limit for the number of levels, $N_T(s,t)$ is the limit for the number of the nets starting at level s and reaching level t, and $N_P(t)$ is the limit for the number of product terms on level t. It makes no sense to have a GPLA composed of PLAs with different width. Because of physical design issues, a column in the AND-plane (actually containing two bits, one for the positive signal, one for the complement) and a bit in the OR-plane may have different widths. To achieve a uniform PLA width, it is required that all

$$N_T(t,t) + r_{OA} \cdot \sum_{s=0}^{t-1} N_T(s,t) \qquad t = 1,2,..,N_L$$

are nearly the same, where r_{OA} is the width ratio of an OR-plane column to an AND-plane column. Normally this ratio is about 0.5. It is convenient to express the net and product term distribution by a provide/consume diagram as shown in Figure 2. This representation will be frequently used in the discussion of the GPLA synthesis algorithm. A valid GPLA implementation should satisfy $n_L \le N_L$, $n_P \le N_P$ and $n_T(s,t) \le N_T(s,t)$ for all $1 \le t \le n_L$ and $0 \le s < t$.

3. AREA AND DELAY FORMULATION

The area and delay of a RPLA are both explicitly expressed in terms of the PLA contents, so the objective function in the optimization can be evaluated with good accuracy. This contrasts with standard-cell design where the area and delay are hard to predict during technology independent logic optimization. The width, height and area of the PLA on level t are:

$$s_X(t) = W_{PA} + W_I \cdot n_I(t) + W_{BM} + W_O \cdot [n_O(t)+1]$$
$$s_Y(t) = W_{BI} + W_P \cdot (n_P(t)+1) + W_R \cdot [n_O(t)+1]$$

The number of horizontal wires in the river routing region is exactly $n_O(t)$, as will be shown in Section 4.1.3. The "+1" terms in the formulae account for the tracking lines, if a dynamic PLA structure is adopted. All the constants are defined in Table 1.

Table 1. Definitions of PLA parameters

W_{PA}	Width of pre-charging circuit of the AND plane
W_I	Width of a pair of complementary input columns
W_{BM}	Width of the intermediate buffer
W_O	Width of an OR plane column
W_{BI}	Width of the input switch
W_P	Width of a product line
W_R	Width of a wire in the river routing area

The height of a RPLA is the sum of the heights of the PLAs, and the width is the largest width:

$$S_X = \max_{t=1}^{n_L} s_X(t), \qquad S_Y = \sum_{t=1}^{n_L} s_Y(t),$$

The non-uniformity of the PLA widths results in white space on the right side of those PLAs whose widths are smaller than the RPLA width. The GPLA has the same area formulation, except everything in the formulae is constant. Since buffers isolate the PLAs in the stack, the delay computation is just a summation of the delays of all the PLAs. In delay analysis, accuracy depends only on the delay model, because given a synthesized PLA, there is nothing unknown such as fanout number etc. [1]. We use a simple delay model, assuming the delay of a transistor or a buffer is a constant delay plus a variable part linearly proportional to the number of loads. For one product line in the AND-plane, the delay is the time the first transistor on the line starts to turn on, plus its turn-on transition time. The first transistor being turned on is the one with the minimum number of transistors in the corresponding AND column:

$$dA_p = d_{BI} + d_{BI}\left|n_P C_{poly} + \min_i\left(n_{TRi} C_{GS}\right)\right|,$$

where d_{BI} and d_{BI} are the constant and coefficient of the linear delay model for the input switch, $n_P C_{poly}$ forms the load of an AND column, n_{TRi} is the number of the transistors on the i-th AND column (only those AND columns on which there is a transistor connecting to the product line p are considered), and C_{GS} is the transistor gate capacitance. The turn-on transition time depends on the load plus the number of transistors on the product line:

$$dP_p = d_{TR} + d_{TR}\left(n_{TRp} C_{DS} + C_{BM}\right),$$

where d_{TR} and d_{TR} are the constant and coefficient of the linear delay model for the turn-on transistor, n_{TRp} is the number of transistors in the product line p, C_{DS} the transistor load capacitance, and C_{BM} the capacitance provided by the intermediate buffer. The delay of the input switch plus the AND-plane is:

$$d_{AND} = \max_p\left(dA_p + dP\right).$$

Similar results hold for d_{OR}. The delay computation for GPLAs is the same.

4. DESIGN METHODOLOGY

4.1. Synthesis of RPLAs

4.1.1. Multi-level logic minimization.

A multi-level logic minimizer is called to generate an initial Boolean network.

4.1.2. Node level-placement.

Node level-placement is possible because some nodes have flexibility in their levels. Suppose the level of node d is $l(d)$. If $l_{HFI}(d)+1<l(d)$ or $l(d)<l_{LFO}(d)-1$, where $l_{HFI}(d)$ is the highest fanin level and $l_{LFO}(d)$ is the lowest fanout level, then the node is said to be down-movable or up-movable. In this case, the level of a node d can be chosen within the range of $l_{HFI}(d)$ and $l_{LFO}(d)$. Notice that the flexibilities of different nodes may be correlated, because of the fanin/fanout relations. It has the good effect of decreasing the number of possible combinations, and thus shrinking the solution space to be searched. Simulated annealing is suitable for this, since the solution space is reduced due to the level dependencies, and the evaluation of area and delay is straightforward. The objective function is chosen as a weighted sum of the area and delay. In the area and delay computation mentioned above, every variable can be trivially derived from the configuration, except n_P, the number of product terms. The exact value of n_P requires a SOP minimization. One could call the SOP minimizer each time the cost is to be computed to get the exact product number; this might not be slow, because the PLA in a RPLA is usually not big. A faster but more conservative way is to use ns_P, the sum of the product numbers of all the nodes in the cluster, which is an upper bound of the number of product terms of the final multiple-output PLA. A trade-off is that when ns_P exceeds a threshold, we do not call the SOP minimizer but simply use the value ns_P. Otherwise a SOP minimization is performed. After the level of the flexible nodes is determined, the nodes on the same level are assigned to a multiple-output PLA and further optimized with a SOP minimizer.

4.1.3. Net ordering.

Each net is given two variables describing its starting level and ending level, denoted by $s(n)$ and $e(n)$. The starting level is the level of the PLA driving the net, and the ending level of the last PLA using this net as an input. The net ordering algorithm can be explained via the example shown in Figure 3. First, order the nets in the descending order of $e(n)$. For the nets with the same $e(n)$, order them in ascending order of $s(n)$. This results in Figure 3(a). There is no restriction on the nets with the same $s(n)$ and the same $e(n)$, and they will always go side by side. Then, starting from left to right, put each net in the leftmost slot that is vacant. Figures 3(b) to (d) illustrate the process. Applying this algorithm will not generate any twisting in the routing layers, meaning that river routing with two metal layers is feasible. Net ordering done in this example leads to a RPLA layout shown in Figure 1.

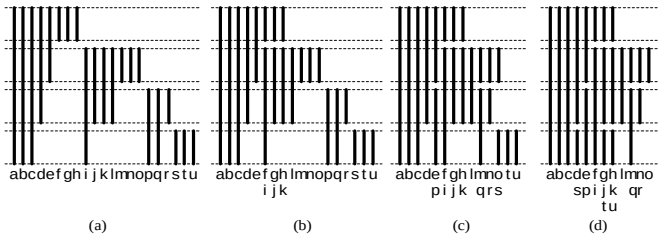

Figure 3. Net ordering example.

This simple scheme causes a little inaccuracy in the computation of PLA widths. Notice that in the bottom PLA in Figure 3(d), there is a vacant slot at the position "l". This makes the real AND column number of the bottom PLA larger than the number of its inputs, which is used to calculate the PLA size as described in Section 3. As long as such PLA is not the widest in the RPLA stack, all the computations remain the same. Experiments have demonstrated that the case where the widest PLA has empty columns is rare.

4.2. Synthesis of GPLAs

The synthesis of GPLAs starts with a multi-level logic minimization with level control. It is followed by an iteration of node level-placement and methods to fix different violations. The iteration either completes with a valid solution found, or quits after a given number of trials and switches to a larger GPLA configuration. If successful, a post optimization is performed to improve speed. In the following context, we use the term "N_P violation" to define the violation of $n_P>N_P$, "N_A violation" to define $n_T(s,t)>N_T(s,t)$ where $s<t$, and "N_O violation" to define $n_T(t,t)>N_T(t,t)$.

4.2.1. Multi-level logic minimization.

This step is similar to that in RPLA synthesis, except the number of levels is controlled to be no greater than N_L during the minimization.

4.2.2. Node level-placement.

As in the RPLA design flow, simulated annealing is used to place the nodes on levels. However, the cost function is the total number of violations. The following Subsections describe the methods for fixing violations remaining after node level-placement.

4.2.3. Net shifting.

Net shifting provides "detours" for nets causing N_A violations. Starting from the bottom left of the provide/consume diagram, we handle N_A violations one by one. The following pseudo code describes how a N_A violation of $e_V=(n_T(s_V,t_V)-N_T(s_V,t_V))$ excessive signals at (s_V,t_V) is handled by net shifting:

```
for (t=t_V−1; t>s_V, t−−) {
    sn=min(N_T(t,t)-n_T(t,t), N_P(t)-n_P(t));   //no. of spare lines.
    if (sn≤0) continue; else dn=min(sn, e_V); //no. of detours.
```

$n_T(s_V,tt)$−=dn for all tt=t+1 to n_L;
$n_T(t,t)$+=dn; $n_P(t)$+=dn; $n_T(t,tt)$+=dn for all tt=t+1 to n_L;
if (e_V==0) return success;
}
return failure;

An example is shown in Figure 4, where the N_A violations at (s=2,t=5) and (s=2, t=7) are eliminated through net shifting.

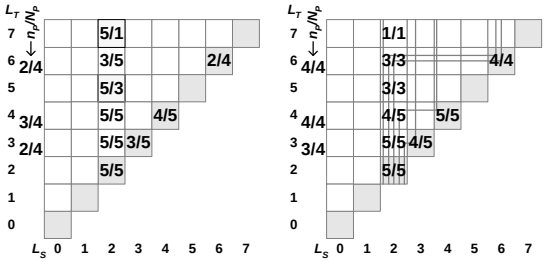

Figure 4. An example of net shifting.

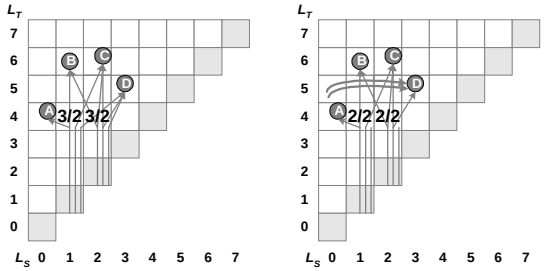

Figure 5. An example of node collapsing.

4.2.4. Node decomposition.

The objective of this operation is to reduce the number of product terms on a level with the N_P violation. The decomposition operation factors a node and makes its divisors new nodes in the network. The new node may not have trouble in finding a place in the stack. Area minimized multi-level logic synthesis aims at minimizing the total number of literals, which is a good estimate of the total number of transistors if the circuit is to be implemented with gates. The decomposition does not change the total number of literals, so the initial synthesis step may not generate this decomposition. However, in the synthesis of GPLAs, the first goal is to get a valid implementation for the given configuration. Using more nodes or more literals should not be a problem as long as the mapping is valid. To choose the best node to be decomposed, we simply examine the nodes on the level with the highest N_P violations and select the one having the most product terms. Since decomposition of a node will generate new nodes and the new nodes may affect other levels, the node decomposition is only performed on one node at a time.

4.2.5. Node collapsing.

The node collapsing operation deals with the N_A violations. N_A violations mean that the current and higher levels use excessive nets from the lower levels. A remedy is to decrease the need, of the current and higher levels, for nets coming from lower levels. The collapse operation removes intermediate nets from the inputs of a node and replaces them by primary inputs. This operation can be supported by reserving sufficient vertical lines for the primary input nets. We choose the node using the most unique intermediate nets crossing the violation level. The number of unique nets is the saving achieved by collapsing the node. So the algorithm checks all nodes on or above the level with the highest N_A violations to identify a node with the maximum number of unique nets crossing the violation level and collapses it. An example is shown in Figure 5, where node D is collapsed because it used two unique nets passing through level 4.

4.2.6. Node elimination.

N_O violations are handled via node elimination. This operation eliminates a node and collapses its function into all its fanout nodes. Eliminating a node immediately reduces $n_T(t,t)$ by one. Node elimination effectively makes a copy of the current node function in all its fanout nodes, so the implementation of the functions in the fanout nodes may cause unexpected growth in the product terms. Thus the node to be eliminated should be chosen carefully. The first criterion is the one with the least fanouts. If a tie occurs, choose the one with the least product terms. Further ties can be resolved by choosing the one with the least input nets.

4.2.7. Post optimization for performance.

Post optimization tries to improve the performance, because the previous steps focus on seeking a valid rather than a fast implementation. Post optimization is a greedy approach, accepting any node displacement that preserves validity but reduces total delay. The algorithm terminates when no further improvement can be found.

5. EXPERIMENTAL RESULTS

In the experiment, three structures are compared: standard-cells with three-layer (3 metal layers) over-the-cell-routing (SC), four-layer NPLAs (NPLA) and two-layer RPLAs (RP). Each example starts with a *SIS* optimization on the initial Boolean network and creates a network with n_{L0} levels [2]. Then the number of levels is gradually reduced via partial node collapsing. For each number of levels, n_L, the circuit is technology mapped twice with *SIS*, one with area priority and one with delay priority[1], generating SC-a(n_L) and SC-d(n_L). We assume the SC designs have 100% area utilization and are 100% routable, so no placement and routing is performed. A clustering algorithm, similar to the one proposed in [10], is adopted to build the NPLAs; however there is no area or delay priority. The placement of the NPLAs is done by a force-directed macro-cell placer [4], producing NPLA(n_L). Two RPLAs are synthesized, one with 100% weight on area and one with 50% weight on area and 50% weight on delay, generating RP-a(n_L) and RP-d(n_L). A 0.35-micron technology is assumed for all circuits. The NPLA clustering and the RPLA mapping algorithms are written in JAVA; and all other procedures are within *SIS* package and *ESPRESSO* [3]. The programs ran on a Dec Alpha 8400 5/625 workstation. Sixteen combinational examples from the LGSynth 91 benchmark set [5] were tested. The gate counts of the examples range from 300 to 17K. The results are given in Table 2. To give a better view of the results, we normalize the area and delay data and plot them in Figure 6. For each example, the normalization is with reference to the data of the SC-a(n_{L0}). The layouts of all three types of implementations for alu2(n_{L0}=10) are shown in Figure 7; routing is done by the *Cadence Warp Router*. The experimental results show that in general, fewer levels mean faster speed but larger area. RPLAs (2 layers) needs on average 23% more area than SCs (3 layers), and NPLAs (4 layers) needs on average 31% more area. Note that RPLAs and NPLAs do not use 100% of the

[1] SIS command "map -m 0 –AF" is used for area-prior tech-mapping, and "map -n 1 –AFG" for delay-prior tech-mapping.

area. RPLAs have an average area utilization of 78%, and 76% for NPLAs. If the module is to be embedded in a big chip, other circuits may occupy this white space. In terms of the delay, on average RPLAs are 4% slower than SCs, and NPLAs 9% slower. Figure 6 shows that SCs have cleaner shapes for the area/delay trade-off curves, while NPLAs and RPLAs have more scattered distributions. Synthesis times of SCs are several times faster than for NPLAs and RPLAs. In the RPLA case, the synthesis time is exactly the design time. However, for SCs and NPLAs, additional time is needed to complete the placement and routing. In addition, the placement and routing may change the delay achieved by the synthesis. Thus it is possible that the final layouts do not meet the requirements, which may invoke another run of the design flow.

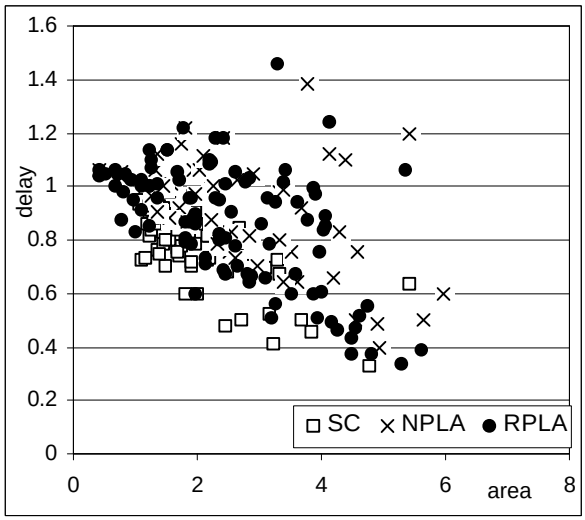

Figure 6. Normalized area/delay.

Three GPLA configurations are considered, named G1K, G5K and G10K respectively. The three FPGAs used for comparison are the *Xilinx* XC4000XL series, 4002, 4020, and 4062, which have similar numbers of programmable bits as G1K, G5K and G10K respectively. Some important parameters are listed in Table 3 [9]. The same LGSynth 91 benchmark examples are tested. The GPLA synthesis algorithm is written in JAVA, except the multi-level logic minimization uses *SIS* [2] and the SOP minimization uses *ESPRESSO* [3]. The FPGAs are synthesized by *Synplicity Synplify Pro* 6.2.4. Due to the lack of a technology library for the programmable devices, we do not have area and delay information available in the experiment. Only a qualitative measurement of "whether the circuit can be implemented in the configuration" is given. The results are shown in Table 4. For the FPGA implementations, some of the failures are caused by insufficient I/Os, not insufficient logic resources. If I/O limit is not an issue, the results show that GPLAs and FPGAs have similar powers for implementing the benchmark circuits.

Table 3. Configuration parameters.

	GPLA			*Xilinx* FPGA		
	G1K	G5K	G10K	4002	4020	4062
max.no.primary input	85	185	384	64	224	384
max.no.primary output	214	540	1132			
max. no. levels	12	18	24			
no. programmable bits	54k	456k	1.41M	61k	522k	1.43M

Table 4. Comparison of GPLA and FPGA

Example	# PI	# PO	G1K	G5K	G10K	4002	4020	4062
apex7	49	37	●	●	●	○	●	●
alu2	10	6	●	●	●		●	●
C1355	41	32	●	●	●	○	●	●
apex6	135	99		●	●		○	●
alu4	14	8		●	●		●	●
x3	135	99		●	●		○	●
C2670	234	139			●		○	●
k2	45	43		●	●		●	●
C3540	50	22		●	●		●	●
dalu	75	16		●	●		●	●
i8	133	81		●	●		●	●
i10	257	224			●		○	○
C5315	178	123		●	●		○	●
C6288	32	32			●		●	●
C7552	206	107			●		○	●
des	256	245			●		○	○

(1) ● means a successful mapping; and ○ means that the logic can be fully mapped, but the I/O ports are insufficient.

6. CONCLUSIONS

RPLAs and their design methodology are described. The high regularity of the RPLAs is attractive in DSM integrated circuit design, since it greatly relieves the burden of analyzing huge amount of layout patterns. More important, is that RPLAs have a simple design methodology; conventional design flow steps of technology mapping, placement and routing are eliminated. Thus there is no outer loop in the design for timing closure and hence the design cycle is shortened. Experimental results show that, compared to other structures, RPLAs provide similar delays. The RPLAs implemented with 2 metal layers consume on average 23% more area than standard-cells with 3 layers, however on average 22% of the area overhead is white space, which might be utilized by other circuits. RPLAs offer a good alternative to the standard-cell structure, providing higher circuit regularity and easy design methodology. The GPLA is a re-configurable version of the RPLA. The contents of the PLAs are programmable, but all PLA sizes and interconnections are fixed. Experimental results show that GPLAs can be a viable alternative to the FPGA structure. Extension to sequential circuits is easy for RPLAs and GPLAs. The space under the river routing wires can be used to build flip-flops. One more metal layer (third layer) can be used to route feedback wires from higher level PLAs to lower level PLAs, and these can also be completed via river routing.

7. ACKNOWLEDGEMENTS

This work was supported by GSRC (grant from MARCO/DARPA 98DT-660, MDA972-99-1-0001).

8. REFERENCES

[1] C. Papachristou and A. Pandya, "A design scheme for PLA-based control tables with reduced area and time-delay cost", IEEE Transactions on CAD, vol. 9, no. 5, May 1990, 453-472.

[2] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis", Tech. Rep., UCB/ERL M92/41, Electronics Research Lab, University of California, Berkeley, May 1992

[3] R. Rudell and A. Sangiovanni-Vincentelli, "Multiple-valued minimization for PLA optimization", IEEE Transactions on CAD, vol. 6, Sep 1987, 727-750

[4] F. Mo, A. Tabbara and R.Brayton, “A Force-Directed Macro-Cell Placer”, Proceedings of ICCAD, Nov 2000
[5] http://www.cbl.ncsu.edu/pub/Benchmark_dirs/LGSynth91/
[6] R. Brayton, G. Hachtel and A. Sangiovanni-Vincentelli, “Multi-level logic synthesis’, Proc. of IEEE, vol. 78, Feb. 1990
[7] R. Brayton, G. Hachtel, C. McMullen and A. Sangiovanni-Vincentelli, “Logic minimization algorithms for VLSI synthesis”, Kluwer Academic Publishers, 1984
[8] “Silicon VLSI Technology”, Chapter 5, Lithography, edited by J.D. Plummer, M.D. Deal and P.B. Griffin, Prentice Hall, 2000
[9] Xilinx Datasheet, XC4000E and XC4000X series field programmable gate arrays, version 1.6, May 1999.
[10] S. Khatri, R. Brayton and A. Sangiovanni-Vincentelli, “Cross-talk immune VLSI design using a network of PLAs embedded in a regular layout fabric”, Proceedings of ICCAD, Nov 2000, 412-418

Table 2. Comparison of SC-a, SC-d, NPLA, RPLA-a and RPLA-d

		gate count		area (1000 um^2)					delay (ns)					area utilization			synthesis time (minute)				
Circuit	#level	SC-a	SC-d	SC-a	SC-d	NPLA	RP-a	RP-d	SC-a	SC-d	NPLA	RP-a	RP-d	NPLA	RP-a	RP-d	SC-a	SC-d	NPLA	RP-a	RP-d
alu2	10	644	875	34.8	47.25	15	14.1	14.4	8	6.1	8.5	8.5	8.3	80%	79%	76%	0.5	0.5	1.9	2.4	2.6
	6	888	1091	48	58.95	46	29	33	7.4	5.9	8.5	8.4	8.2	73%	77%	73%	0.5	0.5	1.8	2.2	2.1
	5	972	1222	52.5	66	55	38.4	38.4	7.2	5.6	8.1	8.2	8	82%	78%	78%	0.5	0.5	1.6	2.2	2.2
alu4	16	1175	1522	63.45	82.2	49	42.6	46	15.6	15	16.5	16.6	16.3	74%	75%	75%	0.5	0.5	2.3	3.1	3
	10	1494	1716	80.7	92.7	80	50.9	62.3	12.9	12.3	15	15.3	14.8	80%	78%	73%	0.6	0.6	2.5	3	3
	7	1958	2219	105.8	119.9	109	69.5	78.8	12	11.2	14.3	14.2	13.3	80%	83%	74%	0.8	0.8	3	2.9	2.8
	5	3119	3333	168.5	180	162	120	148.8	11.5	11	12.9	12.2	12.8	68%	80%	82%	0.9	1	3	2.8	3
apex6	7	1180	1572	63.75	84.9	93	85.9	86.3	14	13.2	14	14.1	13.4	92%	90%	87%	0.8	0.8	2.3	3	3
	4	1380	1575	74.55	85.05	99.2	122.5	119.1	13.6	12.1	12.3	12.3	12.1	88%	90%	89%	1	1.1	2.4	3.4	3.5
apex7	8	350	430	18.9	23.25	14.8	9.5	12.9	6.4	5.8	6.6	6.7	6.4	90%	97%	82%	0.4	0.4	1.8	2	1.9
	5	425	522	22.95	28.2	25.5	14.7	18.8	5.2	4.5	5.8	5.6	5.3	75%	98%	85%	0.4	0.4	1.8	1.9	1.8
C1355	10	755	900	40.8	48.6	70.8	61.8	61.8	9.4	8.1	10.9	10.7	10.7	72%	61%	61%	0.5	0.5	2	2	2.2
	7	813	947	43.95	51.15	79.4	50.1	50.1	8.8	7.9	10	9.4	9.4	77%	75%	75%	0.6	0.5	2.2	2.8	2.5
	5	1113	1477	60.15	79.8	89.3	77.2	80.2	8.2	7.4	9.4	9	8.2	70%	80%	77%	0.6	0.6	2.3	3	2.9
C2670	12	1119	1516	60.45	81.9	159.4	158.4	205	17.4	16.1	17.8	18.4	17.7	80%	75%	71%	0.8	0.8	4	4.4	3.9
	9	1733	2213	93.6	119.6	200.4	170.8	218.1	17	15.9	17	18	16.4	72%	79%	82%	1	1.1	4.2	4	3.9
	6	2150	2450	116.1	132.3	259.4	182.5	240.3	16.8	14.2	14.5	15	13.2	71%	69%	72%	1.5	1.6	5	4.4	4.2
C3540	23	2044	2780	110.4	150.2	150	136.5	140.4	25.8	22.4	28.9	29.4	28.4	81%	76%	75%	0.8	0.9	4.2	4.8	3.9
	13	3575	5286	193.1	285.5	203	189.3	200.3	20	19.9	22.4	26.4	20.8	73%	77%	73%	1	1.2	3.3	4.9	4.2
	9	4397	6363	237.5	343.7	315	200.8	234.2	18.8	18	21	20.5	18.4	71%	77%	70%	1.8	2	3.8	4.4	4.4
	6	6819	11091	368.3	599	504	310.6	389.2	17.3	16.4	19.4	17.4	15.4	68%	72%	76%	2	2.4	4	4.4	4.3
C5315	15	2633	3133	142.2	169.2	290.4	312.4	236.7	29	27	30.8	31.4	30.5	77%	73%	71%	1	1	4.9	4.8	4.2
	8	3500	4027	189	217.5	420	378.3	439.1	25	26.1	20.3	20.3	19	73%	91%	83%	1.5	1.8	4.8	4.8	5.1
	6	4444	5247	240	283.4	500	450	508.2	22	25	22	22.7	19.6	67%	98%	88%	2.2	2.4	5.6	5.3	5.9
C6288	25	6086	7861	328.7	424.5	694	725.1	732.7	50	44	55.8	55.1	54.6	80%	65%	65%	5	5.3	8.4	9	10.4
	20	7763	8813	419.3	476	1238	1083	1129	47	46.1	69.2	72.9	53	78%	73%	77%	5	5.2	9	12.2	13
	14	11883	16388	641.7	885	1785	1356	1759	45.2	42.2	60	62.2	53.1	79%	82%	79%	5.8	5.8	10.2	12	10.5
C7552	18	3388	5055	183	273	609.9	692.3	730	50.4	40.2	40.2	43.9	30.5	80%	61%	67%	3	3.1	8.2	8.2	9
	14	5283	6119	285.3	330.5	770	710	780	40.2	30.3	33.1	30	23.2	80%	77%	69%	3.4	3.4	12	8	8.9
	9	6805	9180	367.5	495.8	830	760	820	30	25.4	25.4	25	19	75%	72%	77%	4	4.1	11	9.5	9.9
	6	8352	10894	451.1	588.3	902	820	880	24.3	20.6	20.1	22	18.8	72%	80%	82%	4.9	5	11.2	10	10.3
dalu	14	1327	1691	71.7	91.35	130	127.4	164.9	18	16.4	21.9	21.9	21.2	79%	77%	72%	1.2	1.3	4	3.2	3.4
	8	1638	2000	88.5	108	174.3	174	175	16.9	16	21.3	21.2	18.2	77%	77%	78%	1.4	1.4	4	3.3	3.9
	5	2222	2472	120	133.5	207.3	140.1	181.8	15.1	14.9	18.8	16.1	16.3	65%	94%	84%	1.5	1.5	3.9	3.8	3.8
des	8	5038	5569	272.1	300.8	999	887	1093	36	26.2	33	34	30.2	82%	84%	82%	2	2.3	4.5	5.9	5.1
	5	11116	16511	600.3	891.6	1620	1292	1527	26.4	26	21.4	20	14	81%	78%	81%	4.9	4.8	6	8.3	8.9
i8	7	1633	2186	88.2	118.1	204.5	216.3	216.7	14.8	14.2	12.3	12	10	88%	79%	78%	0.9	1	3.1	3.9	3.2
	5	1713	1997	92.55	107.9	220.5	187.7	173.8	14.1	13.7	11.2	10.8	8.9	79%	86%	86%	0.9	1.1	2.9	3.3	4
i10	22	3611	4833	195	261	450	459	470	43.7	43.2	34.2	35	30.2	77%	71%	77%	1.5	1.8	5.2	4.9	5.2
	13	7777	9194	420	496.5	702	555	620	35.6	32.5	28.2	28.2	22.3	80%	80%	82%	2.9	3.3	5.9	5.1	5.5
	9	11388	13250	615	715.5	957	770	890	22.9	21.9	21.1	22.3	20.5	72%	82%	85%	3.3	3.8	5.9	8.2	8.7
	6	13916	17222	751.5	930	1103	902	1029	19.9	14.3	22	22.4	14.8	74%	92%	88%	5	5.5	6	9	9.1
k2	14	1638	2261	88.5	122.1	200.8	203.9	207.1	27.5	20.6	27.5	26.3	26.1	80%	76%	75%	1	1.4	5.9	5	5.7
	8	1913	2783	103.4	150.3	230.2	232.3	254.3	20.2	15	20.1	21.4	18.2	79%	86%	82%	2	2	6.4	6.8	6.4
	4	4050	5675	218.7	306.5	299.4	250.8	288.4	18.8	14.8	17.7	18	15.4	73%	90%	85%	3.7	4.1	6	7.2	6.9
x3	8	1413	1683	76.35	90.9	150	94.9	143	14.8	14.2	14.4	15.8	14.2	71%	89%	76%	0.7	0.8	1.8	2	2.1
	4	1522	1788	82.2	96.6	169	138.3	150	12.4	11.6	13	12.8	12.7	62%	90%	90%	0.8	0.8	1.8	1.9	2