A High Performance Radiation-Hard Field Programmable Analog Array

Ji Luo, Joseph B. Bernstein, J. Ari Tuchman, Hu Huang, Kuan-Jung Chung
University of Maryland at College Park
2100 Marie Mount Hall, College Park, MD 20742
Phone: (301) 405-0357, fax: (301) 314-9601
Email: {jluo, joey, huanghu, kjchung}@eng.umd.edu, ari@technoventures.com,
Anthony L. Wilson
Mission Research Corporation, 5001 Indian School Road NE, Albuquerque NM 87110
E-mail: awilson@mrcmicroe.com

Abstract – While interest, availability and use of FPAs have grown, FPAs still have not achieved the same success as FPGAs in the digital domain. This results from several factors, including the lack of CAD tools, small circuit density, small bandwidth and layout dependent noise figures. These factors are all related to each other, making the design of a high performance FPA a multi-dimensional problem. A critical reason behind these difficulties is the non-ideal programming technology, which contributes a large portion of parasitics into the sensitive analog system. This paper presents a high performance, radiation hard Laser Field Programmable Analog Array (LFPA) using LaserLink’s MakeLink™ technology. Because of its extremely low resistance, negligible parasitic capacitance and full compatibility with commercial CMOS process, MakeLink™ can fully reduce these FPA design concerns and offer a breakthrough capability in analog array performance. The proposed LFPA consists of a 4 x 4 array of Configurable Analog Blocks (CABs) surrounded by abundant interconnect resources. There are 16 PAD groups around the chip, and 8 tracks per X/Y channel. Each CAB has 4 input and 4 output pins with internal circuit operating in fully differential mode. With appropriate programming, the LFPA can provide an accurate, low-cost and rapid-prototyping analog ASIC solution.

1. Introductions and Motivation

High-density digital integrated circuits have dominated the semiconductor market, but analog circuits as the interfacing blocks still play an essential role in today’s complex, high performance systems. After all, the real world is analog. Analog ICs find broad applications including analog filters, oscillators, modulators, pulse shaping circuitry, A/D, D/A converters, telecommunication systems, closed-loop control in industrial and integrated signal processing circuitry for various sensors. “For every dollar spent on microprocessors, another $1.50 is required to create an interface to the rest of the system” [1].

An important advantage of digital systems is their immunity to noise. More importantly, the clearly defined hierarchical decomposition of digital systems with well-accepted performance measures makes digital ICs very ease of design over analog systems. Many CAD compatible digital system design methodologies have been developed, which lead to short design cycles and low cost. On the contrary, the analog design process still features a lot of intuitive and manual design approaches. So there is a substantial need for developing high performance, reconfigurable analog ICs suitable for CAD methodologies in order to improve design efficiency. This has been the motivation for research in the area of Field Programmable Analog Arrays [3].

In general, an FPA is a monolithic collection of configurable analog building blocks (i.e., CABs), a programmable routing network used for passing signals between CABs, and a block of memory (for SRAM based FPA) storing configuration data [4]. Alternatively, the circuit topologies may be defined by other methods such as antifuse technologies. There are several FPAA designs in the academic literature and some commercial chips exist in the industry. However, a general purpose FPA suitable for high frequency applications does not yet appear. This can be due to (1) previous designs are based on bandwidth limited switched-capacitor techniques; (2) most FPAs use MOS transistor or antifuse based programming switches; and (3) a lack of appropriate CAD tools. In addition to introducing radiation softness, those switches can bring a large amount of parasitics (Fig. 1) into the sensitive analog circuit thus severely degrading circuit performance.

This paper presents a high performance, radiation hard Laser Field Programmable Analog Array using LaserLink’s MakeLink™ technology. Because of its extremely low resistance, negligible parasitic capacitance and full compatibility with commercial CMOS processes, MakeLink™ switches can offer a breakthrough capability in analog array performance.
The organization of the paper is as follows: section 2 compares currently available programming technologies and introduces MakeLink™ and its advantages. Section 3 describes the internal CAB circuitry. Section 4 presents the LFPAA architecture and the routing problem. At last, a brief conclusion is given and some future work is proposed.

Fig. 1 Intrinsic capacitance of a MOSFET switch

### 2. Programming Technologies

To date, FPAAs have been programmed with SRAM and antifuse technologies. A SRAM-based FPAA is programmed by loading the configuration from an external source. Each switch (usually a MOS pass transistor or transmission gate) is controlled by a memory cell. All known FPAAs have used SRAM-based interconnect. The big advantage of SRAM technology is its reprogrammability. The disadvantages are volatility, large area cost, high serial resistance (>1KΩ), large capacitance associate with MOSFET terminals, and low radiation tolerance.

Actel’s PLICE (programmable low-impedance circuit element) antifuse, Metal-to-Metal antifuse and Quicklogic’s Vialink antifuse are also well-known programming technologies (Fig. 2). They have relatively smaller resistance and occupy less area, but they still suffer from large parasitics and long-term reliability issue. All of them are not fully compatible to standard CMOS processes. Due to their nature, they are unable to carry large current density and therefore cannot be used in analog circuits (Table 1). Ideally we wish the programmable switches had the properties of a metal wire. MakeLink™ is the most promising technology approaching this goal.

The structure of laser MakeLink™ is schematically illustrated in Fig. 3. (7)

![Fig. 3. Schematics of laser-induced vertical metal link, (a) top view and (b) cross section A-A’](image)

The principle of link formation employs the contrast of material properties between the metal and the surrounding dielectrics SiO₂/Si₃N₄. The IR laser beam passing through the square hole of the upper metal (M2) frame is impinged on the lower metal (M1) line with negligible loss of energy in the covering dielectrics. The laser energy is absorbed on the surface of M1 to be resulting in a sharp metal temperature increase. Due to the extremely low thermal conductivity and light absorbency of the dielectrics, the dielectric temperature is not changed so much. In the mean time, metal expansion fractures the surrounding dielectrics along the stress concentration paths and molten metal fills in the crack. At an optimal laser energy and spot size, dielectric cracks can be controlled to initiate from the upper corners of the M1 line and terminate near the inside lower corners of the M2 frame without propagating to the outside of the structure or fracturing the top dielectric passivation. Fig. 4 (7) shows a FIB cross-section view of a vertical laser MakeLink™.

MakeLink™ can also be formed laterally for even smaller size. Advantages of Laser MakeLink™ in the application of programmable devices include: (1) extremely low electrical resistance per link: 2-3 orders smaller than that of an Actel antifuse or a MOSFET switch; (2) high reliability and tolerant to high current density; (3) area efficient: it is a pure metallic link and does not occupy any silicon area; (4) fully compatible with commercial CMOS processes; and (5) radiation hard: because no active devices are involved. A
A comparison between MakeLink™ with other antifuse technologies is listed in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>MakeLink</th>
<th>Actel M2M</th>
<th>Actel ONO</th>
<th>QuickLogic</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_{nominal}</td>
<td>0.5 - 5Ω</td>
<td>~25Ω</td>
<td>100-500Ω</td>
<td>~ 80Ω</td>
</tr>
<tr>
<td>C_{nominal}</td>
<td>~ 0</td>
<td>2.9 fF</td>
<td>7.7 fF</td>
<td>1 fF</td>
</tr>
<tr>
<td>EM Robustness</td>
<td>Strong 2-D metal sheet</td>
<td>Weak 1-D metal filament</td>
<td>Weak 1-D metal filament</td>
<td></td>
</tr>
<tr>
<td>Radiation tolerance</td>
<td>Radiation tolerant</td>
<td>Radiation tolerant</td>
<td>Radiation tolerant</td>
<td></td>
</tr>
<tr>
<td>CMOS Compatibility</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Geometry Size</td>
<td>4X</td>
<td>2.5X</td>
<td>2.5X</td>
<td>1X</td>
</tr>
<tr>
<td>Current density</td>
<td>~1 mA</td>
<td>~ μA</td>
<td>&lt; mA</td>
<td>~ μA</td>
</tr>
<tr>
<td>Applications</td>
<td>Analog, digital and Mixed-signal Circuit</td>
<td>Digital Circuit</td>
<td>Digital Circuit</td>
<td>Digital Circuit</td>
</tr>
<tr>
<td>Leakage Current</td>
<td>None</td>
<td>3 nA</td>
<td>10 nA</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1. Comparison of different antifuse technologies [5, 6, 8] (0.35um CMOS process)

3. Configurable Analog Block Circuitry

The functionality that an FPAA can implement is largely determined by the CAB configuration. A good CAB circuitry provides more flexibility and also helps achieve high performance for the instantiated system.

A major choice when designing an FPAA is whether to operate in discrete-time or continuous-time. Discrete-time approaches, for example, switched-capacitor technique, are well suited to digital control and do not require the use of on-chip tuning circuitry for VLSI implementations of programmable components. However, such sampled-data techniques require that input signals be band-limited to at least one half of the sampling frequency and hence anti-aliasing and reconstruction filters must be used. This requirement significantly limits the bandwidth of discrete-time FPAA circuit implementations (usually less than 1/10 of the clock rate) [9]. Thus we prefer continuous-time circuit design for the LFPAA.

A CAB is usually composed of a few programmable capacitor arrays (PCA), programmable resistor arrays (PRA) (Fig. 5) and an op-amp-like analog unit. This unit is the core and should be a universal analog function block, i.e. we wish easy configurability to implement as many analog functions as possible. The candidate circuitries include traditional op amp [10], operational transconductance amplifier (OTA) [11] and second-generation current conveyor [12]. In our LFPAA, a four-terminal floating nullor (FTFN) combining both voltage and current operation mode capabilities is adopted [13]. It has been demonstrated that any active circuit can be realized by FTFN [14]. A fully balanced configuration is desired because 1) it has large output swing; 2) circuit is less susceptible to common-mode/coupling noise; and 3) there are no even-order nonlinearities thus less harmonic distortion [15]. The disadvantages are they require a CMFB circuit and consume more power. In our CAB design, a Fully Balanced Four Terminal Floating Nullor (FBFTFN) is used.
current source load and amplified. The low impedance Z terminal (due to compensation feedback) can be used for voltage-mode signal operation. On the other hand, the W terminal associated with high output impedance is suitable for providing output current signals.

Unlike the custom analog IC design, the routing resources in FPAs are fixed and limited. All connections have to be completed within the horizontal and vertical channels via Manhattan paths. The FPAA routing architecture not only affects routability but also has significant impact on the performance of the implemented circuit. Considering the FPAA circuit topology, we proposed an array-based architecture as shown in Fig. 8.

The LFPAA architecture contains a 4X4 CAB array. To accommodate for fully differential circuit operation, each CAB has 8 pins, 4 input pins on the left of the CAB and 4 output pins on the right of the CAB. Each CAB is surrounded by 4 connection boxes. There are 8 tracks per horizontal (X)/vertical (Y) channel. 25 switch boxes are uniformly distributed at the vertical and horizontal channel intersections. It has 16 I/O PAD groups (some reserved PADS are not shown in the figure), with 8 PADS per group. The left column and bottom row PADS are for input only; the top row and right column PADS are for output only. This “prototype” LFPAA architecture does not have segmentation but it can easily be modified, adding more CABS, PADS, tracks and/or segmentation scheme to develop into a more versatile structure.

4.2 LFPAA Router

Similar to FPGA, implementing an analog circuit on FPAA requires appropriate programming of thousands of switches. Obviously this has to be done with the aid
of a set of supporting CAD tools. Fig. 9 shows a simplified FPAA design flow. The end user only describes a targeted application at a high level of abstraction, for analog array, typically using a schematic entry with the IPmodule/CAM (configurable analog module) library provided by the FPAA manufacturer. Then CAD tools convert this high-level description into a programming file, which specifies the state of the switches in the FPAA.

![Fig. 9 LFPAA Design Flow](image)

Due to the inherent nature of analog system, the technology mapping phase and placement phase might have to be done manually and semi-manually. Thus, routing plays an essential role in FPAA design automation.

For LFPAA, the coordinate system is defined in Fig. 7, from (0, 0) to (5, 5). The four corner positions, (0, 0), (0, 5), (5, 0), (5, 5), are blank areas. Each X or Y directed channel belongs to the pad or CAB right below it, or on the left to it, having the same coordinates. The legal routing connections are:

- **LHS pads can connect to all the tracks in channel y (0, 1); RHS pads can connect to all the tracks in channel y (4, 1).**
- **Bottom row pads can connect to all the tracks in channel x (1, 0); top row pads can connect to all the tracks in channel x (1, 4).**
- **Input CAB pins can connect to tracks in the channels immediately on the left, top and bottom of the CAB; output CAB pins can connect to tracks in the channels immediately on the right, top and bottom of the CAB.**
- **Tracks in the horizontal channel can connect to tracks in vertical channel if a switch is available at the intersection.**
- **Direct connections between CAB pins are not allowed; direct connections between PADS and CAB pins are not allowed.**
- **Dogleg is not allowed, i.e., CAB pin cannot be acted as intermediate vertex to route a net.**

Based on above definition, the LFPAA architecture is converted into a highly detailed routing resource graph (RRG), which contains all the connection and constraints information, by a RRG generator. Then the router performs routing for an input netlist on RRG.

The goal of routing is not only to complete all the required connections without congestion, but also to satisfy a set of performance constraints. In digital domain, the performance constraints are induced by RC delay which can be counted efficiently with the timing term in cost function, while the performance constraints (tolerable variation of gain, bandwidth, noise etc.) imposed on analog array are too abstract for the routing tools to handle directly, thus they must be converted to a set of routing constraints, or, interconnect parasitic constraints. Once the routing constraints are met, the performance constraints of the analog circuit should also be satisfied. The parasitics that are to be controlled during routing are metal wire resistance, switch resistance, metal wire to ground and metal-to-metal capacitance. Among them, parasitic capacitance is the major concern. Those parasitic effects are classified into bounding constraints and matching constraints and incorporated into a cost function. Cost function is used as the criteria to determine how the connections are made.

Our LFPAA router is based on the Pathfinder Negotiated Routing Algorithm [16, 17, 18]. We have finished a routability-driven router for LFPAA. Although it’s called routability-driven, the router in fact not only resolves the congestion but also tries to find the “shortest path”. In other words, the accumulated parasitics (especially the loading capacitance and serial resistance) are automatically kept to a near minimum value, along with the wave expansion process. Also, because of the extremely small parasitics associated with MakeLink™ switch, this router is sufficient for this relatively small-scaled FPAA.

5. Conclusion

While interest in, availability and use of FPAs has grown, FPAs still cannot achieve the same success as FPGAs have in digital domain. This is the result of several factors. A critical reason behind these difficulties is the non-ideal programming technology. This paper presents a high performance, radiation hard
Laser Field Programmable Analog Array using LaserLink’s MakeLink™ technology. Compared to the traditional SRAM based FPAA, LFPAA has the advantages of flexible circuit topology, area efficiency, radiation hardness and extremely small parasitics. Therefore it provides a promising high performance, low cost and fast-prototyping analog ASIC solution.

We will further investigate the analog performance constraints on analog circuits and their conversion to routing constraints. Also more work is required to develop large-scale FPAA architectures and function-rich analog IPModule libraries.

Fig. 10 LFPAA Routing Algorithm

References


RT( neti): a linked list storing current routing of net i
Sort the nets according to their classifications
while (overuse exists & max iteration not exceeded) {
  for (all.net) {
    rip-up existing RT( neti) and update p(n);
    init RT to source;
    for (all sinks of net i) {
      Initialize PQ to RT;
      while (no sink found) {
        dequeue PQ;
        for (all fanout vertices n of node m) {
          If (n not a pin/pad & un-reached)
            add n to PQ & update pathcost(n);
          else {
            (n is a sink)
            add it to a temp sink list;
        }
      }
      if ( more than one sinks found) {
        add those sinks and their parents to RT;
        update p(n) only if n is not contained in RT;
      }
      for (all vertices in path) {
        update p(n) only if n is not contained in RT;
        add n to RT(i);
      }
    }
  }
  Update h(n) for all n;
}