Programmable Floating Gate FPAA Switches Are Not Dead Weight

Christopher M. Twigg*, Jordan D. Gray§, and Paul E. Hasler†
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332–0250
Email: *ctwigg@ece.gatech.edu, §jgray@ece.gatech.edu, †phasler@ece.gatech.edu

***Abstract*— In most reconfigurable systems, such as FPAAs and FPGAs, the switch element and its associated memory cell is a necessary overhead for performing computation with the active components. However, floating gate based switches in large-scale FPAAs can be programmed anywhere between simple "off" and "on" connections, which allows these programmable conductances to be used as computational elements within synthesized circuits. This decreases the notion of switches as computational dead weight and increases the potential computational area efficiency within FPAAs.**

I. Reconfigurable Systems and Switches

Reconfigurable devices, such as FPGAs and FPAAs, enable the rapid synthesis and development of complex systems without long fabrication cycles. However, the flexibility of these systems generally requires a significant area overhead for the numerous switches and the memory cells that control them, which are used to interconnect the computational components. For digital devices, such as FPGAs, this area overhead can easily consume 60% to 90% of the chip [1], [2]. FPAAs, such as the RASP 2.7 depicted in Fig. 1, have similar or sometimes smaller routing areas. Although these switches are required to provide reconfigurability, they are computational dead weight. However, the floating gate transistor switches of the large-scale FPAA discussed in [3] provide a way to utilize some of this dead weight.

The key technology enabling reconfigurability in these large-scale FPAAs is the floating gate transistor. Several designs implement reconfigurability through programmable circuit topologies, such as the G_M cells of [4], which avoid switch overhead but generally incur other area penalties in the form of redundant computational components. Reconfigurability is commonly achieved using a transmission gate switch, as depicted in Fig. 2a. A memory cell, such as an SRAM, controls the transmission gate which connects a row and column within the routing network. Since the memory cell is digital, the switch can only operate between two states, "on" and "off". However, the floating gate transistor, depicted in Fig. 2b, provides a combined switch and memory cell capable of any arbitrary state between "on" and "off". Using these additional states, many useful circuits can be synthesized within the switch fabric.

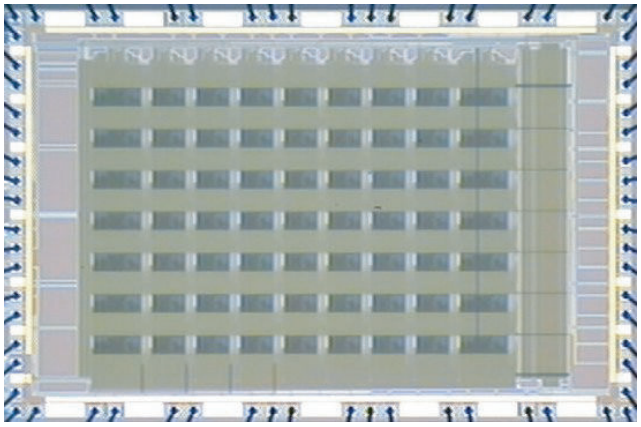

Fig. 1. RASP 2.7 die photograph. The window-like rectangles are the computational elements, and the areas in between are composed of routing switches.

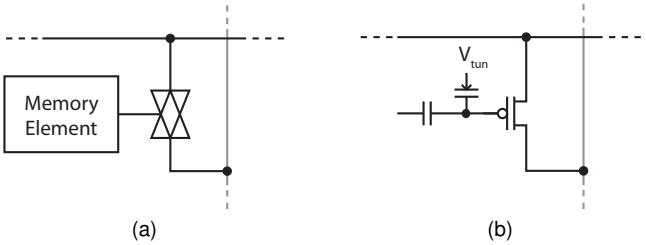

Fig. 2. FPAA switching elements.

II. Floating Gate Transistor Switches

The floating gate switch is simply a pFET with capacitive gate coupling, as shown in Fig. 3. A double poly capacitor is used to couple a control voltage, V_C, to the floating gate, and a MOS capacitor is used as a tunneling junction. With no DC path to a fixed potential, charge can be stored on the floating node. The current flowing through the floating gate pFET is then controlled by the voltages coupled onto the floating gate and the amount of charge stored on the floating node. By modifying this charge, the transistor can be programmed over a wide range of values for a given set of terminal voltages.

Fowler-Nordheim tunneling is used to globally erase the floating gate pFET switch array, which can be viewed as an increase in the effective threshold voltage. Hot electron injection is then used to accurately program devices [5], which decreases the effective threshold voltage by adding electrons to the floating node. These two processes allow the floating

1-4244-0921-7/07 $25.00 © 2007 IEEE.

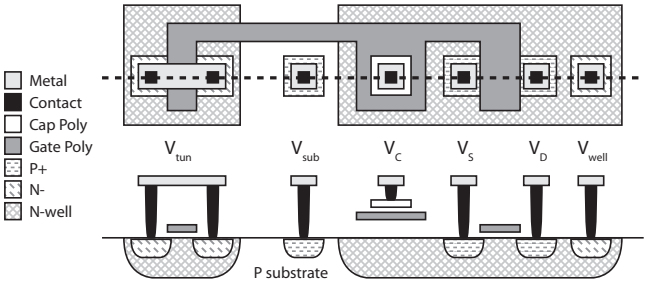

Fig. 3. Floating gate pFET layout.

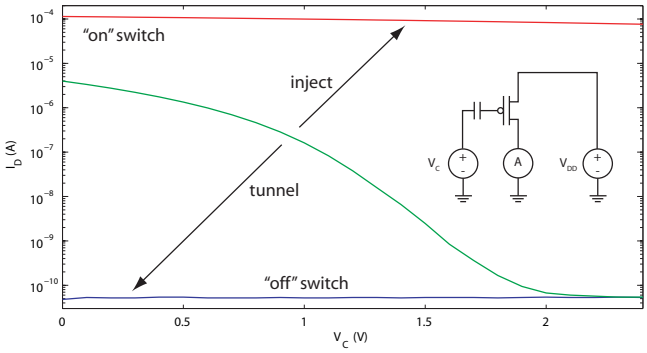

Fig. 4. Injecting and tunneling floating gate switches.

gate pFET to be programmed anywhere between the "off" and "on" states, as depicted in the V_C sweeps of Fig. 4. In the "off" state, the floating gate pFET conducts negligibly for any control voltage. The "on" state is characterized by a switch that conducts well for any control voltage. However, programming the switch to conductance levels between "off" and "on" is the key to using these devices for computation and biasing.

III. Switch Routing References

The limited number of input/output (I/O) lines on an FPAA can easily be consumed by the various references and biases that synthesized circuits generally require. As the number of biases increases, the complexity of the synthesized circuit quickly becomes constrained by the number of inputs that can be passed to it and the number of outputs generated by the circuit. This situation can be slightly alleviated by including programmable bias structures within the computational analog blocks (CABs), but this increases the area of each individual CAB and decreases the total number of CABs that can fit on the chip, which reduces the computational area efficiency of the chip. However, floating gate switches provide a means of implementing some or all of these references and biases within the switch fabric, which saves space and I/O lines.

A. Current Reference/Source

Fig. 5 shows an example current source synthesized within the switch fabric. In the first case, M_1 is programmed to the desired current, and M_2 is programmed as an "on" switch. As seen in Fig. 5, the current source is strongly dependent upon the drain voltage. This dependence is partially due to the Early

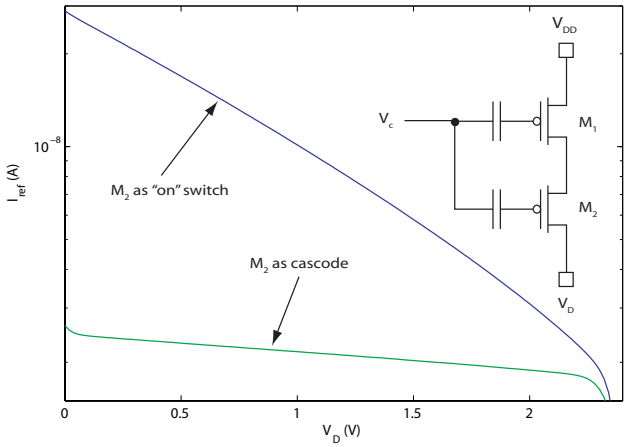

Fig. 5. Current reference/source.

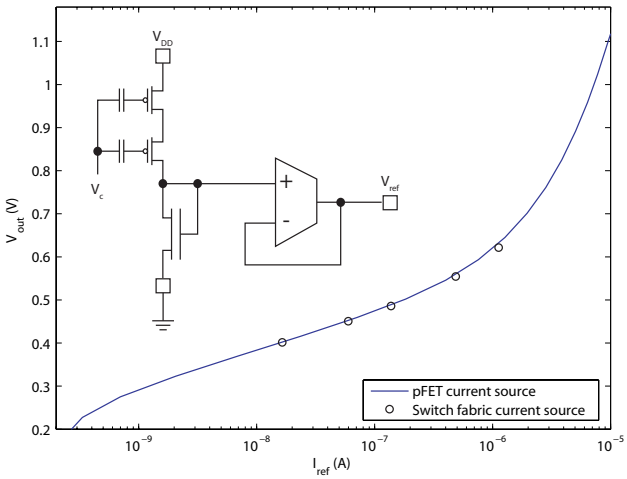

Fig. 6. Programmable voltage reference example.

effect, but it is also caused by the drain voltage coupling into the floating node via the drain overlap capacitance. The drain, source, and tunneling voltages all capacitively couple into the floating node just like the control voltage. In this case, the drain coupling ratio is the drain overlap capacitance divided by the total capacitance at the floating node. To increase switch density, the coupling and tunneling capacitors are kept fairly small, which makes the drain coupling ratio significant.

A common topology for reducing drain dependence is a cascode, which could simply be a properly biased transistor connected to the drain of the current source transistor. In the normal case, this would require another I/O or biasing structure, but in the floating gate switch routing, another switch will do. For this case, M_2 is programmed as a cascode transistor that is biased based upon the coupling voltages and the charge programmed on the floating node. As seen in Fig. 5, this significantly reduces the drain dependence of the current source. If needed, additional routing switches can also be used for multiple cascode transistors at a cost of headroom.

B. Voltage Reference

On-chip voltage references can also be synthesized using routing switches as part of the circuit topology. Fig. 6 depicts an example voltage reference generated using the cascoded current source and a diode connected nFET from a CAB. When driving purely capacitive loads, this circuit is sufficient, but an OTA from one of the CABs can also be used to buffer the voltage for other situations. To demonstrate the programmable range of this circuit, a pFET from one of the CABs was initially used as the current source. Select currents were then programmed within the switch fabric to demonstrate conformance to this curve. Although, this topology only covers part of the supply range, topology modifications can be made to generate any voltage within the supply rails.

IV. Switch Fabric Circuits

The switch fabric can also be used for more complex circuits beyond simple references. Relatively simple circuits may only utilize the reference circuits as part of the computation path, such as programmable envelope detectors. However, larger, more complex circuits, such as diffusors and vector-matrix multiplier (VMMs), can utilize a far greater percentage of the routing fabric as more direct computational elements.

A. Envelope Detector

A common function in analog signal processing is envelope detection. The CAB designs for the RASP 1.5, 2.5, and 2.7 FPAAs [6] contained dedicated envelope detectors for this function, but they are generally not fully utilized and just consume area. One option would be to reduce the number of these specialized devices by only including them in select CABs. However, this limits the number of channels that can be processed simultaneously to some predetermined value. Another option is to synthesize the envelope detectors using the available CAB components and switch fabric elements, as illustrated by the minimum detector depicted in Fig. 7.

The minimum detector uses a switch fabric current source to set the decay rate. An OTA, a pFET, and a drawn capacitor from a CAB are used to track the falling edge of the input. Since these CAB components are generally useful for many other circuit topologies, they are not underutilized when implementing circuits other than the envelope detector. By using the programmable switch fabric current source instead of a dedicated current source, CAB space is also saved for other analog components. As seen in Fig. 7, the decay rate can be significantly adjusted by programming the current source. Fig. 8 shows the response of the minimum detector to sinusoids of frequencies between 100 Hz and 800 Hz when programmed to a single decay rate.

B. Vector-Matrix Multiplier

The VMM and similar processing circuits, such as diffusors, are capable of utilizing large sections of the switch fabric for computation. As depicted by the 2x2 differential VMM structure of Fig. 9, the VMM is composed almost entirely of programmed switch elements. The only CAB component

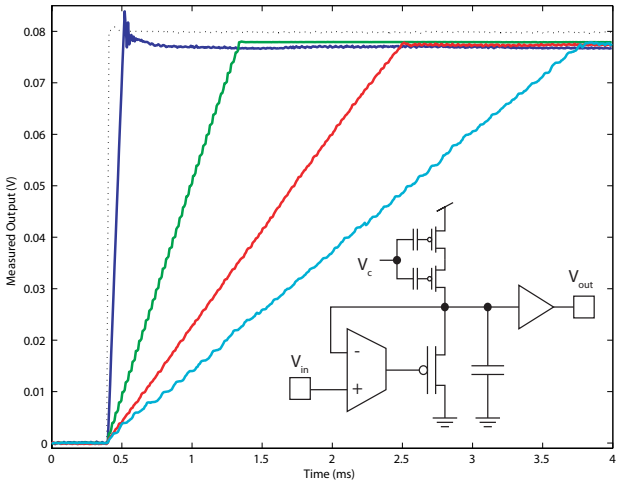

Fig. 7. Minimum envelope detector using switch fabric current source for adjustable decay rates.

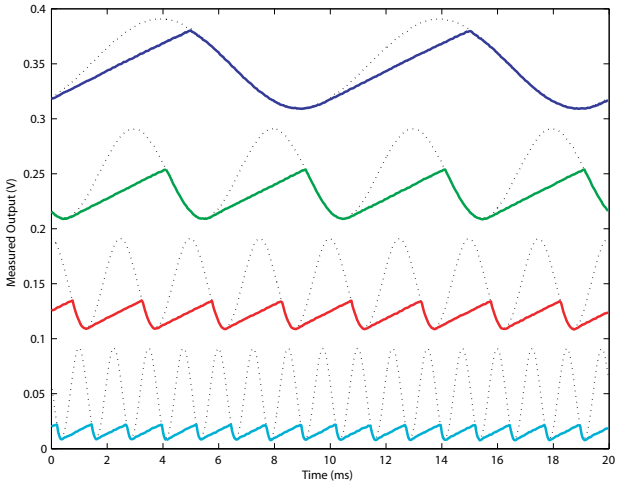

Fig. 8. Minimum detector response to various frequencies. The decay rate can be adjusted to respond appropriately for the frequency of interest.

is the OTA used to buffer the source voltage of the input switch element. The individual multiplier weights are set by programming a charge difference between the input and output switch elements. Multiplier outputs are then tied together for current summation to perform the final computation. A two quadrant multiplier is constructed using a second single-ended multiplier to provide positive and negative inputs. In a similar fashion, a four quadrant multiplier can be constructed by duplicating the output stage of the two quadrant multiplier to provide positive and negative weights.

For a four quadrant structure, the CAB component utilization is dependent upon the number of inputs (#OTAs = 2*#inputs). The switch fabric utilization is dependent upon both the inputs and outputs (#switch elements = 2*#inputs*[2*#outputs + 1]). Since there is no CAB component cost when increasing the number of outputs, additional processing can be added with little impact on CAB component utilization. Using this technique, arbitrarily sized VMMs can be constructed. This

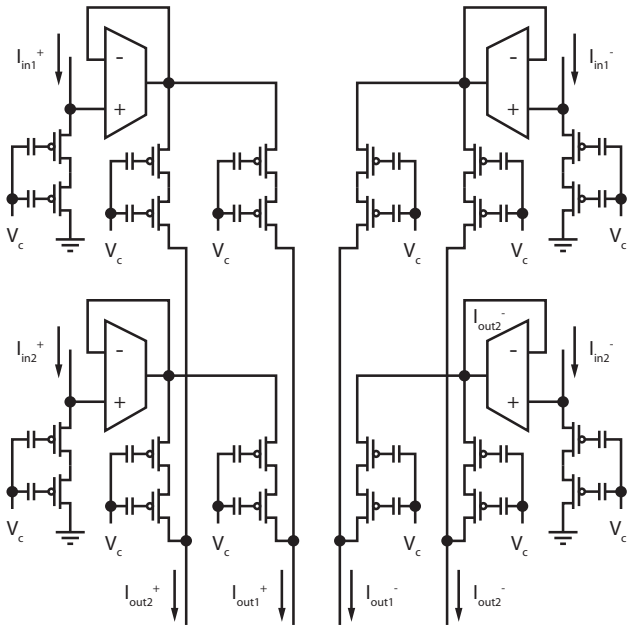

Fig. 9. Differential 2x2 VMM structure utilizing programmable switch fabric elements (two floating gate switches in series).

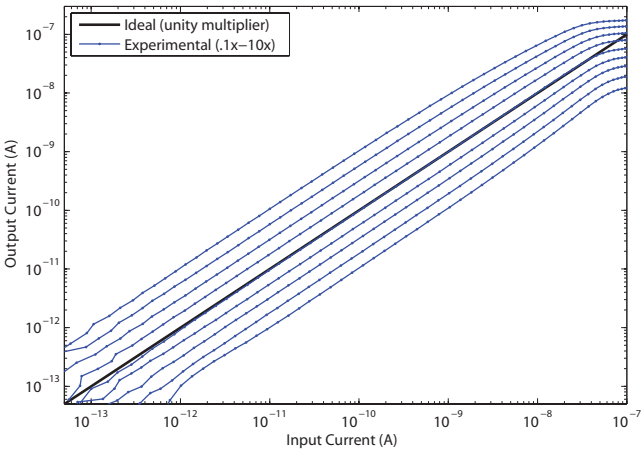

Fig. 10. Single quadrant multiplication with weights programmed between .1 and 10.

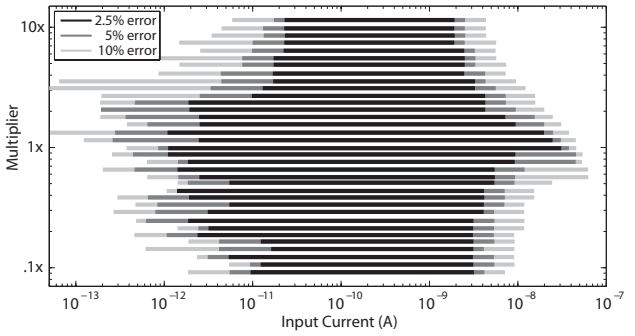

Fig. 11. Plot depicting a range of multiplier values that produce outputs within a $\pm$ 2.5%, 5%, and 10% error band.

eliminates the need to include presized versions of them as CAB components, which saves a significant amount of computational area for other components.

Data from a single multiplier is shown in Fig. 10. This multiplier was constructed with a single floating gate switch element as an input, an amplifier buffering the source voltage, and another switch element as an output. The weight of the multiplier was programmed over two orders of magnitude from .1 to 10. The curvature apparent at higher currents is a result of the transistors leaving the sub-threshold region of operation. The jagged profile at sub-picoamp currents results from limitations in the off-chip measurement equipment. For multiplier weights ranging from .5 to 1.5, a reasonable range for common signal processing tasks, the error was observed to be within $\pm 2.5\%$ over three decades of current.

The data of Fig. 10 is analyzed in Fig. 11 to illustrate the trade-off between accuracy and dynamic range in the multiplier element. The three error bands represent the range of currents over which a particular programmed multiplier results in an output that falls within the specified error range. As illustrated in Fig. 11, the input range is greatest for all error bands around a unity multiplier. Since the circuit is functioning as a current mirror in this range, the effect of offsets between the input and output transistors is minimized.

V. Conclusion

Programmable floating gate based switches have been demonstrated to provide computational power within the switch fabric of large-scale FPAAs. This increases the potential computation density within such devices far beyond what is capable with standard reconfiguration technologies found in most FPAAs and FPGAs. In simple cases, the switch fabric can be used to generate on-chip biases and references that save I/O pins and CAB space for other components. In extreme cases, large switch element circuits, such as diffusors and VMMs, can be implemented almost entirely within the switch fabric, thereby significantly increasing the computational area efficiency of a reconfigurable IC. The notion of computational dead weight does not apply to these switches.

References

[1] H. Schmit and V. Chandra, "FPGA switch block layout and evaluation," in *ACM Proceedings of the International Symposium on Field–Programmable Gate Arrays*, 2002, pp. 11–18.

[2] J. Rose, R. J. Francis, D. Lewis, and P. Chow, "Architecture of field–programmable gate arrays: the effect of logic block functionality on area efficiency," *IEEE Journal of Solid–State Circuits*, vol. 25, pp. 1217–1225, Oct 1990.

[3] C. M. Twigg and P. Hasler, "A large–scale reconfigurable analog signal processor (RASP) IC," in *IEEE Proceedings of the Custom Integrated Circuits Conference*, 2006, pp. 5–8.

[4] J. Becker and Y. Manoli, "A continuous–time field programmable analog array (FPAA) consisting of digitally reconfigurable G_M–cells," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, 2004, pp. 1092–1095.

[5] A. Bandyopadhyay, G. J. Serrano, and P. Hasler, "Programming analog computational memory elements to 0.2% accuracy over 3.5 decades using a predictive method," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, 2005, pp. 2148–2151.

[6] T. S. Hall, C. M. Twigg, J. D. Gray, P. Hasler, and D. V. Anderson, "Large–scale field–programmable analog arrays for analog signal processing," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 11, pp. 2298–2307, Nov. 2005.