Hybrid Residue Generators for Increased Efficiency

Michael B. Sullivan, Earl E. Swartzlander, Jr.
University of Texas at Austin, Austin, TX.

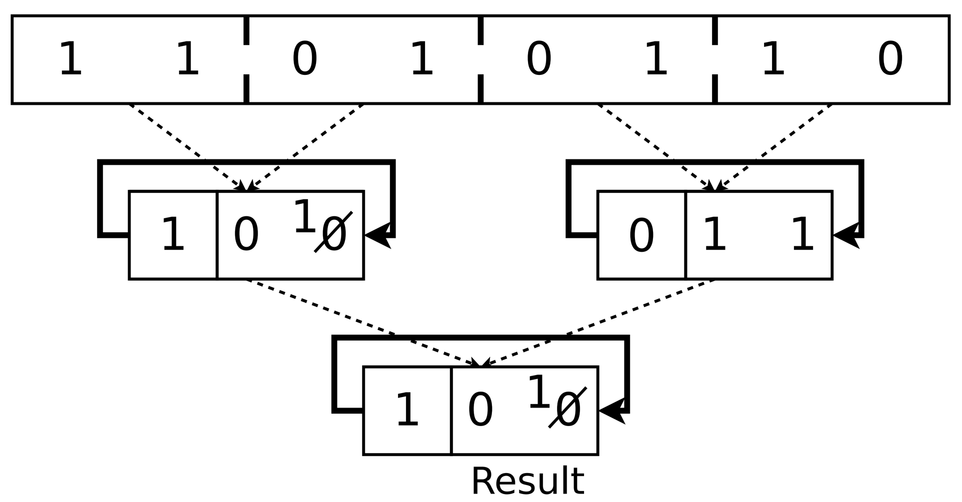
THE UNIVERSITY OF TEXAS AT AUSTIN

Introduction

Low-cost, single-cycle residue generators can be readily formed out of two's complement adders in two ways, which have area and delay tradeoffs. A residue generator using adder-incrementers for end-around-carry adders is small but slow, and a design using carry-select adders is fast, but large. It is shown that a hybrid combination of both approaches is more efficient than either.

Low-Cost Residue Codes

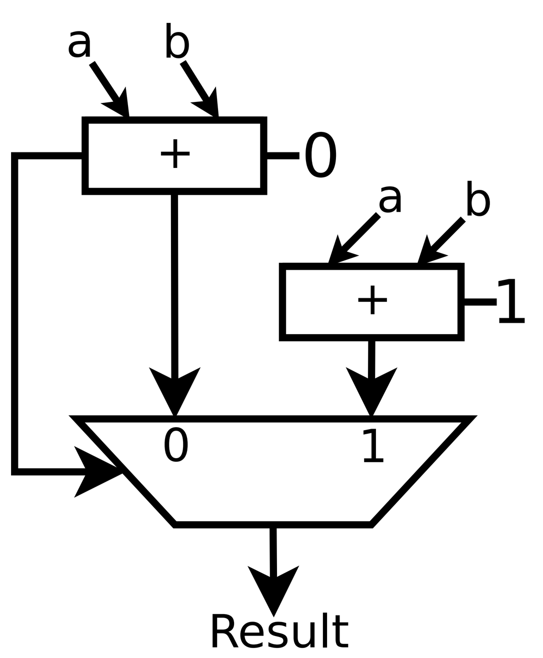
The low-cost residue code of a number number, X, $(|X|_{2^a-1})$ can be generated by the addition of non-overlapping a-bit slices of X under modulo-a arithmetic. This can be implmented as a tree of end-around-carry (EAC) adders, each of which performs a-bit, modulo-a addition.



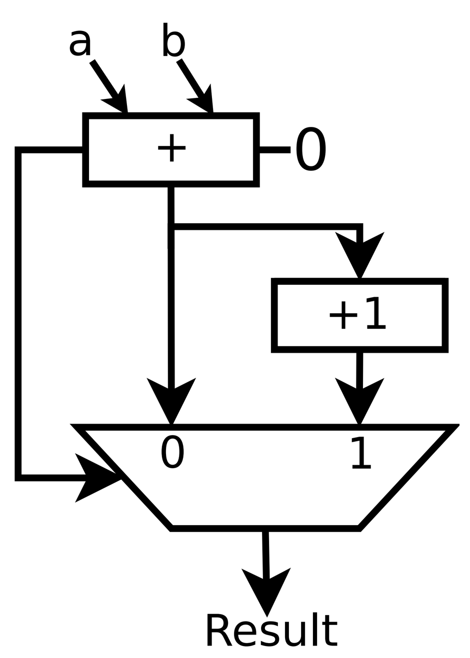
$1 \equiv |214|_3$ using a tree of EAC adders.

End-Around Carry Adders

EAC adders can be made out of commodity 2's complement parts in several ways, which have area, power and delay tradeoffs.



(a) Carry-Select (Fast)

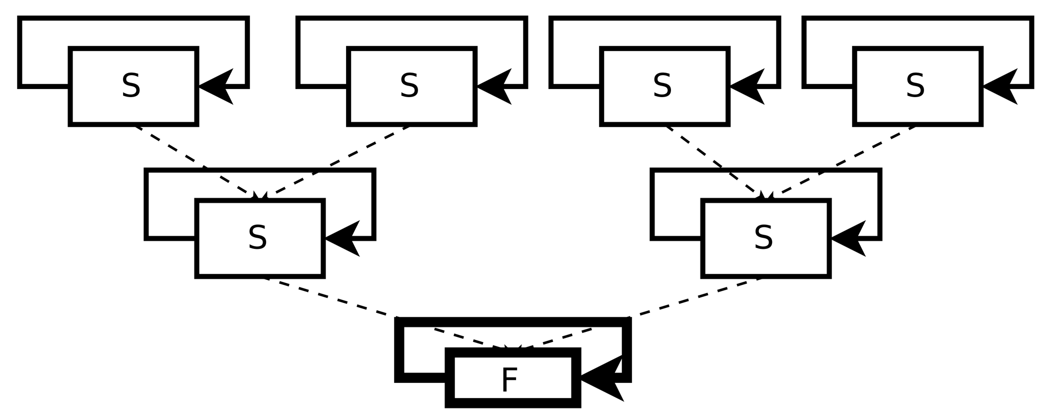


(b) Add-Increment (Small)

Two EAC designs using two's complement adders.

Hybrid Residue Generation

The hybrid residue generator (HRG) is a low-cost residue generator that can combine multiple architecturally distinct EAC adders in order to increase the total implementation efficiency.

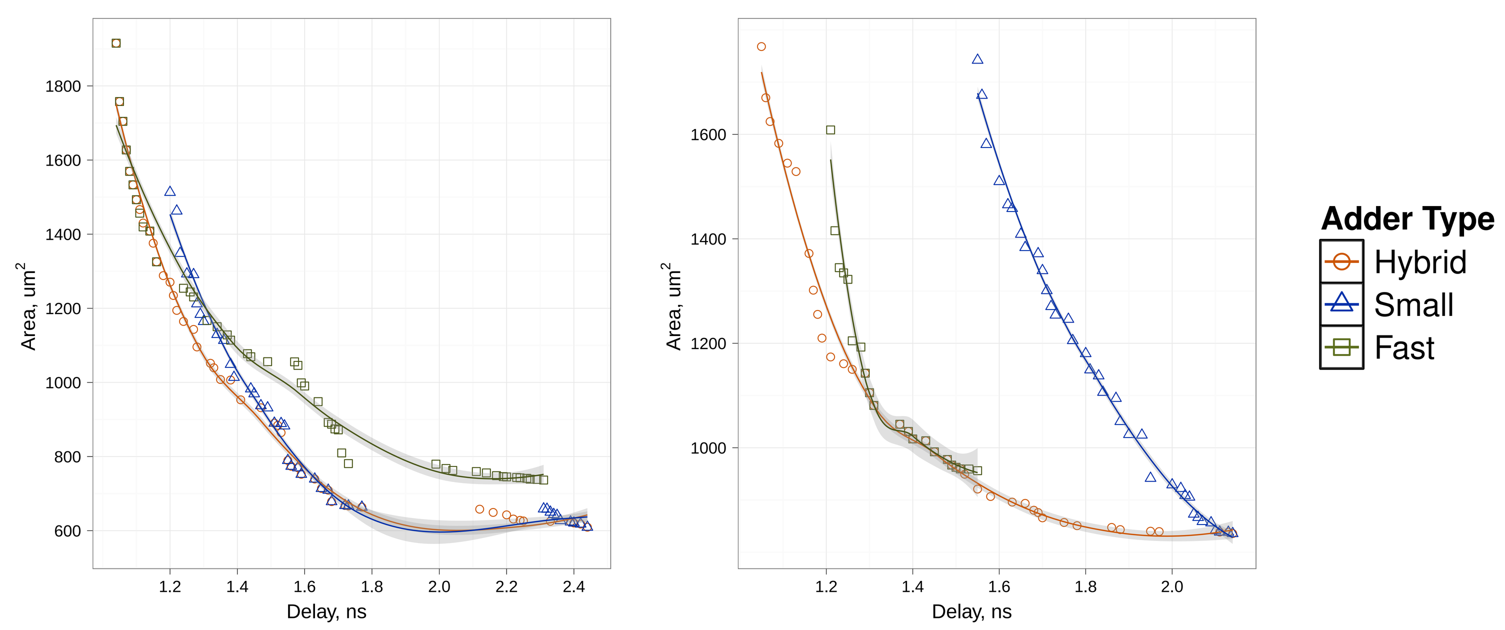


An HRG; Small adders (S) are used to increase efficiency, while the fast adder (F) preserves timing.

This HRG is appropriate for a design where a "small" residue generator tree cannot quite satisfy the timing constraints.

Behavior

Intellegent HRG formation tends to increase efficiency, particularly in regions where neither constituent adder design dominates. In some cases, the HRG can increase decrease the critical speed of operation (given synthesis area limits).



(a) 64-bit ($a = 4$)

(b) 128-bit ($a = 2$)

HRG behavior across different word lengths and moduli.

Evaluation Details

HRG efficiency improvements are investigated across a spectrum of word sizes, modulo widths, and delay budgets using gate-level area and power estimates. The Synopsys toolchain (area optimized, high mapping effort), DesignWare IP parallel prefix adder, and 45nm Nangate Open Cell Library are used.

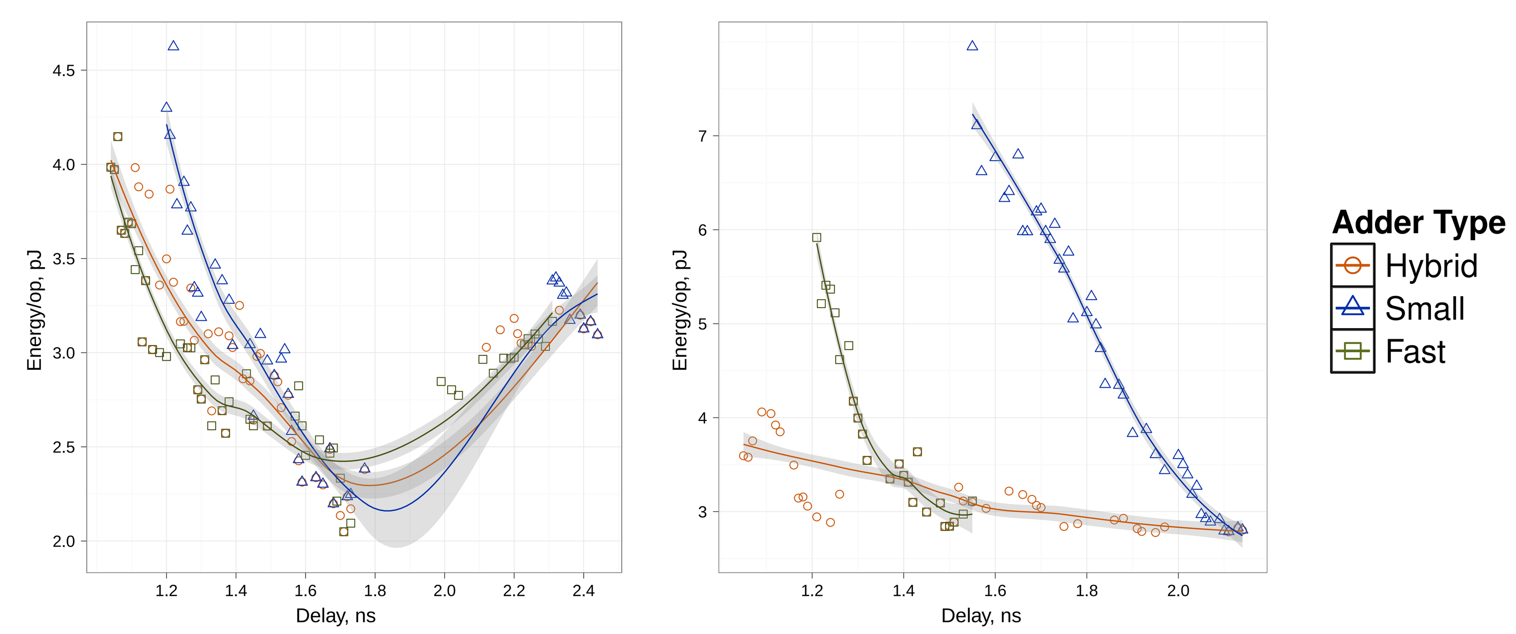
Results

Mean AT and AT^2 HRG savings.

Mod Width (a)	vs. Fast AT	vs. Fast AT^2	vs. Slow AT	vs. Slow AT^2
16-bit				
2	-0.151	-0.157	-0.096	-0.142
4	-0.098	-0.226	-0.034	-0.010
32-bit				
2	-0.098	-0.143	-0.097	-0.024
4	-0.091	-0.213	-0.038	-0.011
8	-0.153	-0.239	-0.012	-0.017
64-bit				
2	-0.113	-0.117	-0.126	-0.110
4	-0.105	-0.209	-0.048	-0.033
8	-0.154	-0.179	-0.005	-0.021
16	-0.216	-0.338	-0.002	-0.009
128-bit				
2	-0.065	-0.035	-0.247	-0.245
4	-0.127	-0.195	-0.046	-0.025
8	-0.173	-0.214	-0.023	-0.034
16	-0.227	-0.360	-0.003	-0.002
32	-0.283	-0.388	-0.008	-0.017

Energy Savings

The HRG has energy benefits, as well. Combined metrics (EDA) can be optimized, even with conicting area and energy trends.



(a) 64-bit ($a = 4$)

(b) 128-bit ($a = 2$)

Energy savings consistent with area improvements.

Mail: mbsullivan@mail.utexas.edu