Design Automation for Charge Recovery Logic

Yilmaz Ege Gonul[1], Leo Filippini[1], Junghoon Oh[2], Ragh Kuttappa[1], Scott Lerner[1], Mineo Kaneko[2], Baris Taskin[1]
[1]*Drexel University, Philadelphia, PA, USA,* [2]*Japan Advanced Institute of Science and Technology, Japan*
Email: {yeg26,bt62}@drexel.edu

***Abstract*—This paper introduces a novel design automation methodology for charge recovery logic (CRL). The proposed methodology combines a novel logic compression algorithm with automatic schematic generation to automate the design process of CRL, enabling power and performance simulations for a large number and variety of CRL circuits. As a measure of the effectiveness of the proposed design flow, automated implementations of CRL equivalents of the LGSynth'91 combinational benchmark circuits are compared with their CMOS counterparts. The results demonstrate a trade-off in power for area: Automatically generated CRL circuits dissipate 51.3% less power on average compared to CMOS equivalents, occupying 54.9% larger area.**

***Index Terms*—Low-power, Charge Recovery Logic, Adiabatic Logic, Electronic Design Automation**

I. Introduction

Charge recovery logic (CRL), also known as adiabatic logic [1], is a logic design approach primarily focused on reducing power consumption through energy recycling. CRL utilizes the power-clock, a periodic signal (usually a sine or trapezoidal wave) to supply both energy and timing to the gates. Depending on the logic family, the power-clock can contain one or more signals, which are commonly called the phases of the power-clock. While CRL was initially used for low-power low-speed applications, previous studies [2] show that CRL can be competitive in the GHz range as well.

Several custom implementations of logic circuits utilizing CRL technology such as [3], [4] can be found in literature. Previous work was also done on the automatic generation of CRL adders and multipliers [5], high-level synthesis for CRL [6], and synthesis of reversible two-level adiabatic circuits [7], [8]. Another work [9] proposed building a CRL family compatible with the traditional standard-cell methodologies, but such approaches do not exploit distinct features of traditional CRL families (e.g. ECRL [10] or PAL2N [11]) such as fixed-delay, differential I/O, and the capability to support large fan-in. This work proposes a novel design automation methodology fully tailored to CRL circuits, leveraging the unique characteristics of CRL.

II. ECRL and CMOS Pipelines

In the remainder of the paper, a four-phase CRL family known as efficient charge recovery logic (ECRL) [10] is used. The design methodologies presented here can be extended to various CRL families with any number of phases. A generic ECRL gate implementing a boolean function F and its complement F^{-1} is illustrated in Figure 1(a). The operation

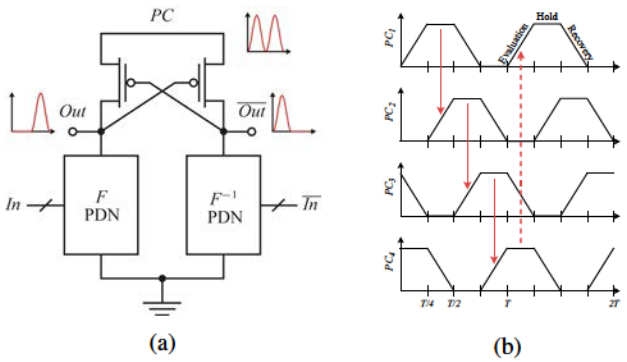


Fig. 1: (a) An ECRL gate implementing a boolean function F and its complement F^{-1} (b) Power clock of a 4-phase logic family.

of the ECRL gate is through the two cross-coupled PMOS transistors that amplify the voltage difference between the two complementary pull-down networks (PDNs). The pull-up activity is through the PMOS transistors, and this allows ECRL to accommodate larger fan-ins than static CMOS [1]. The inputs need to be stable only while the circuit is computing the output, enabling 4-phase operability and leading to the power-clock configuration shown in Figure 1(b). In the 4-phase clocking scheme, consecutive phases of the power-clock have a $-90°$ difference. This translates to the condition where a CRL gate powered by PC_1 can only drive gates powered by PC_2, and driven only by gates powered by PC_4.

Figure 2 shows two sequentially equivalent pipelines, in CMOS and in ECRL, intentionally designed to have equal latencies. Boolean function f in Figure 2(a) is implemented with a combinational circuit in static CMOS, and may comprise hundreds of logic gates. The squares of Figure 2(b) are single ECRL gates, each implementing a boolean function g_i where the composition of boolean functions g_1 through g_{12} is equivalent to the boolean function f. The data in the CMOS pipeline of Figure 2(a) propagates in one CLK period. The data in the ECRL pipeline in Figure 2(b) propagates through phases PC_1 to PC_4 in one power-clock period. If the power-clock and CLK have the same frequency, then the number of logic stages in the ECRL pipeline must match the number of phases of the power-clock to have the same latency of the CMOS equivalent. Equivalent latency enables CMOS integration of CRL implementations without affecting the system timing.

A one-to-one conversion of a CMOS circuit into CRL would significantly increase latency. For example, consider a pipeline

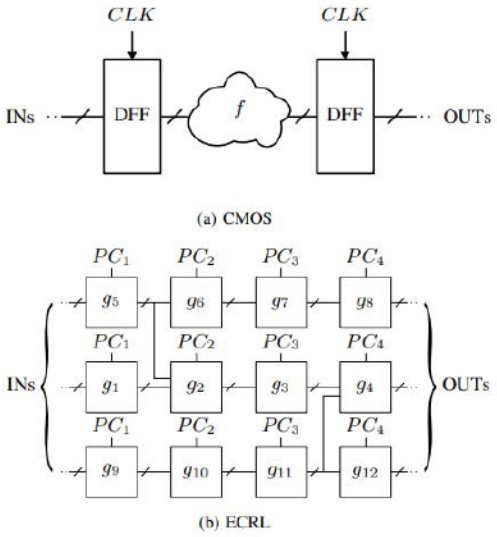


Fig. 2: Traditional CMOS and the equivalent ECRL pipelines.

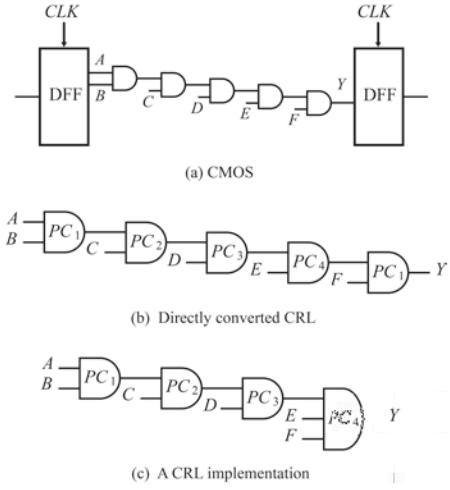


Fig. 3: The boolean function ABCDEF: (a) implemented with 2 input ANDs in CMOS, (b) directly converted from CMOS, (c) a latency-equivalent CRL implementation with a higher fan-in gate.

stage with a boolean function $Y = ABCDE$ as in Figure 3(a). The direct conversion from CMOS to ECRL leads to the circuit of Figure 3(b), which has a latency of 1.25 power-clock periods. The circuits of Figure 3(a) and Figure 3(b) are logically equivalent but have different latencies when CLK and the power-clock PC have the same frequency. The circuit of Figure 3(c) is a possible ECRL implementation of the boolean function $Y = ABCDEF$ that has the same latency as the circuit in Figure 3(a), with a higher fan-in gate. This example demonstrates the value of a CRL-specific automation flow.

III. PROPOSED DESIGN METHODOLOGY FOR CRL

The proposed design automation methodology introduces two stages to the design flow, utilizing features of CRL such as large fan-in capability and differential I/O, as well

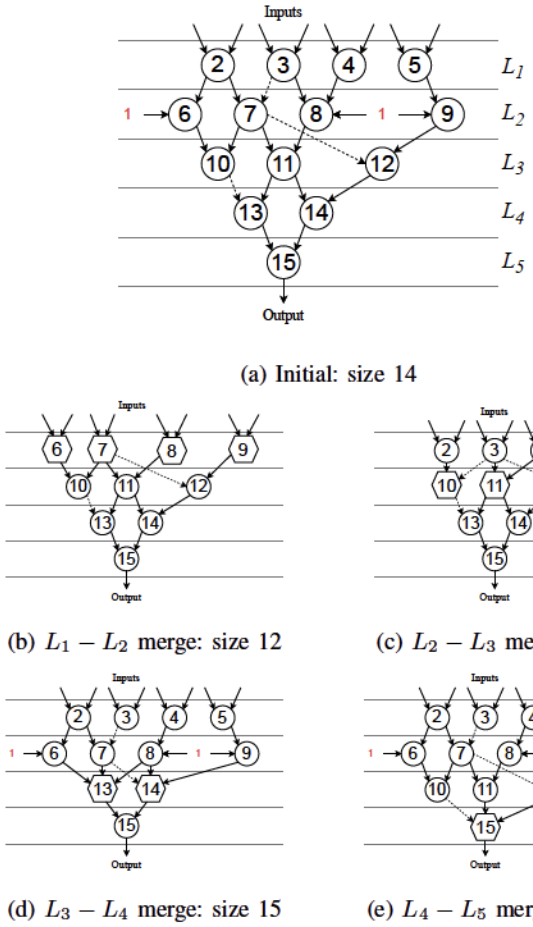


Fig. 4: Example of an AIG divided into logic depth levels. Circles are AND operations, hexagons represent clusters, and dashed edges indicate that the output of a node is negated.

as satisfying the choice of equal latency with CMOS. The first stage is a novel custom-cell based mapping approach that uses a logic compression algorithm. An and-inverter-graph (AIG) [12] representation of a given logic is compressed down to a target logic depth that matches the power-clock phases and achieves the same latency as that of a CMOS design with the same clock period. The second stage is a schematic generation process that maps the compressed logic inside the clusters to PDNs, creating ECRL gates.

A. Logic Compression Algorithm

The input for the proposed compression algorithm is an AIG representation of the input logic. Logic cones in the input logic are minimized and balanced using the logic synthesis tool ABC [13] before the conversion to AIG. This is done to minimize the node count of the AIG, hence the overhead. Figure 4(a) shows an example of an AIG in which each node represents an AND operation and has been assigned a logic depth level. Note that nodes 6, 8 and 9 are different from other nodes with their logic 1 inputs, which make these nodes behave as buffers. The logic in Figure 4(a), if implemented as an ECRL circuit using a power-clock with period T, would

0 0 0 0 σ

σ

0

0 0

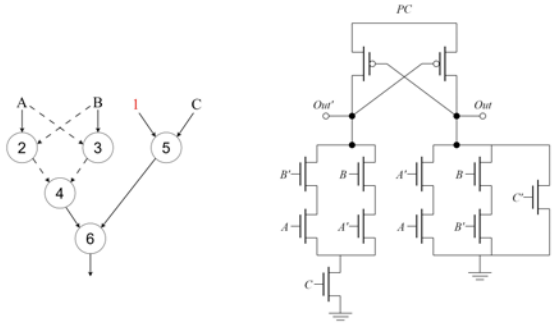
0

0

0

	1	1
	1	1
	1	1
	1	1

	−
h	
h	
σ	h h
σ	σ
σ	σ
σ	σ h
σ	
σ	σ
σ	h
σ	
σ 0	
σ	h
σ	
	σ h
σ	
σ	
σ	σ
σ	σ
σ	
σ	σ
σ	



(a) AIG of $(A \oplus B) \cdot C$ (b) $(A \oplus B) \cdot C$ schematic

Fig. 5: The AIG and the ECRL gate implementing the $(A \oplus B) \cdot C$ boolean function.

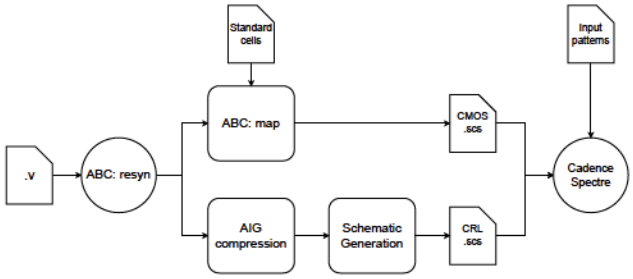


Fig. 6: The experimental flow.

periments are performed using a 65 nm technology node, for 76 combinational circuits in the LGSynth'91 benchmark suite [14]. Both ECRL and reference CMOS implementations are generated as Spectre netlists starting from minimized and balanced Verilog descriptions of the benchmark circuits. CMOS implementations are mapped using a standard cell library containing generic logic gates. The benchmark circuits are simulated at a frequency of 13.56 MHz, targeting the operating frequency of an RFID tag, as such frequencies are typical for charge recovery logic applications [15], [16]. The transistors used to implement both the CMOS and ECRL circuits are selected at the minimum size for consistency. The circuits in the benchmark suite are simulated with random input patterns with an activity factor of 10%. The performance metrics selected in the experiments are i) power dissipation reported from the Spectre simulations of the generated netlists and ii) area estimates represented by the transistor count of the generated netlists.

Power and area comparisons of the synthesized ECRL circuits with CMOS implementations are shown in Figure 7(a) and Figure 7(b) respectively. The figures are histograms of the normalized (ECRL/CMOS) power consumption and area. ECRL implementations consume an average of 51.3% less power compared to static CMOS implementations. On the other hand, ECRL implementations occupy 54.9% more area on average than their CMOS counterparts. 74 of the 76 combinational circuits in the benchmark remain within the curve with a standard deviation of $\sigma = 0.504$ for the area

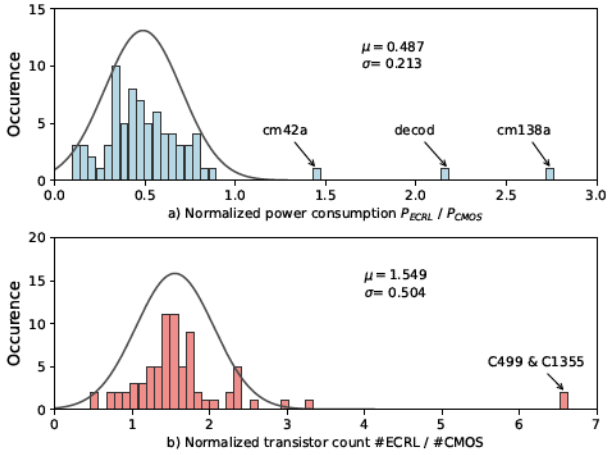


Fig. 7: Normalized (CRL/CMOS) power consumption, at the top, and normalized transistor count, at the bottom, for the combinational circuits of the LGSynth'91 Benchmark suite run at 13.56 Mhz with an activity factor $\alpha = 0.1$.

overhead. 73 of the 76 circuits in the benchmark consume less power than the CMOS implementations.

Circuits $C499$ and $C1355$ are the outliers in Figure 7(b) that have several times higher overhead compared to the rest of the benchmark suite. The outliers in the normalized power consumption in Figure 7(a) are circuits $cm42a$, $decod$ and $cm138a$. Their power consumptions are several times higher than the rest of the benchmark suite because the activity factor of their internal nodes are 5.3%, 5%, and 5.2% respectively, which are lower than the targeted 10%. Because of the power-clock, CRL gates switch their outputs every cycle whether the inputs change or not, giving static CMOS an advantage at very low activity factors [1]. As mentioned in Section III-B, for some of the circuits, a larger target logic depth is used to relax the amount of logic inside the custom cells. Circuits $alu4$, $i10$, $C3540$, and $C6288$ are synthesized with 8 stages of logic, having double the latency of the rest of the benchmark suite. Although these circuits become multi-cycled, both their power consumption and area overhead remain within the curve, showing that the advantages of ECRL are sustained.

V. Conclusions

This work proposes a novel methodology to implement combinational logic as ECRL circuits. The proposed design methodology is applied to the LGSynth'91 benchmark suite, and shows power savings of 51.3% with an area overhead of 54.9% compared to static CMOS implementations on average. Latency of CMOS to ECRL circuits is preserved, paving the use of synthesized ECRL circuits as macro blocks in an SoC integration with interoperable CMOS and ECRL blocks.

VI. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grants No. 1409014, 1816857.

REFERENCES

[1] P. Teichmann, "Adiabatic logic - future trend and system level perspective," *Springer*, vol. 7, 01 2012.

[2] V. S. Sathe, J. Chueh, and M. C. Papaefthymiou, "Energy-efficient ghz-class charge-recovery logic," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 38–47, 2007.

[3] L. Filippini and B. Taskin, "The adiabatically driven strongarm comparator," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 12, pp. 1957–1961, 2019.

[4] L. Filippini and B. Taskin, "A charge recovery logic system bus," in *2017 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, pp. 1–4, 2017.

[5] A. Blotti and R. Saletti, "Ultralow-power adiabatic circuit semi-custom design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 11, pp. 1248–1253, 2004.

[6] L. Varga, G. Hosszu, and F. Kovacs, "Two-level pipeline scheduling of adiabatic logic," in *2006 29th International Spring Seminar on Electronics Technology*, pp. 390–394, 2006.

[7] A. Zulehner, M. P. Frank, and R. Wille, "Design automation for adiabatic circuits," 2018.

[8] Y. Ushioda and M. Kaneko, "Hardware minimization of two-level adiabatic logic based on weighted maximum stable set problem," in *2022 IEEE 40th International Conference on Computer Design (ICCD)*, pp. 394–397, 2022.

[9] C. Lee, P. Hsieh, and C. Yang, "A standard-cell-design-flow compatible energy-recycling logic with 70% energy saving," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 1, pp. 70–79, 2016.

[10] Y. Moon and D. Jeong, "An efficient charge recovery logic circuit," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 4, pp. 514–522, 1996.

[11] F. Liu, "Pass-transistor adiabatic logic with nmos pull-down configuration," *Electronics Letters*, vol. 34, pp. 739–741(2), April 1998.

[12] A. Mishchenko, S. Chatterjee, and R. Brayton, "Dag-aware aig rewriting: a fresh look at combinational logic synthesis," in *2006 43rd ACM/IEEE Design Automation Conference*, pp. 532–535, 2006.

[13] B. L. Synthesis and V. Group, "Abc: A system for sequential synthesis and verification," Mar. 2018. [Online]. Available: https://people.eecs.berkeley.edu/alanmi/abc/.

[14] S. Y. Yang, "Logic synthesis and optimization benchmarks user guide version 3.0," 1991.

[15] M. Avital, H. Dagan, I. Levi, O. Keren, and A. Fish, "Dpa-secured quasi-adiabatic logic (sqal) for low-power passive rfid tags employing s-boxes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 149–156, 2015.

[16] S. D. Kumar, H. Thapliyal, and A. Mohammad, "Ee-spfal: A novel energy-efficient secure positive feedback adiabatic logic for dpa resistant rfid and smart card," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 2, pp. 281–293, 2019.