Power Proportional Adder Design for Internet of Things in a 65 nm Process

Adrian Wheeldon, Jordan Morris, Danil Sokolov, Alex Yakovlev
μSystems Group, Newcastle University
{a.r.wheeldon2, j.morris, danil.sokolov, alex.yakovlev}@newcastle.ac.uk

***Abstract*—In this paper we present a self-timed, power proportional, 32-bit ripple-carry adder design using a state-of-the art cell library. The cell library implements a new transistor sizing strategy for subthreshold in a commercial 65 nm low power process. Simulation results show improvements in performance and energy per cycle when compared to a fixed-period design. The adder has applications in the internet of things where systems are required to operate over a wide range of conditions and with varying power supplies.**

I. Introduction

Nowadays with the proliferation of internet of things (IoT) devices, we see the utilization of low power, 32-bit microprocessors in many different operating environments: from hand-held battery-powered devices; to small, wide sensor network nodes powered by energy harvesters; and mains-powered smart-home devices which are always on. This research marks the beginning of a study on the design of microprocessors for such applications by looking at an adder—a key computational component.

An important factor in the IoT is the reduction of the functional supply voltage ($V_{\mathrm{DD,min}}$). For systems powered by energy harvesters, maximum power point tracking (MPPT) is commonly implemented to extract what little power may be available [1], [2]. The nature of MPPT can lead to a very low V_{DD} supplied to the circuit. In many cases it is preferable for the circuit to continue operation under this condition, albeit with reduced performance. If the circuit did not function at this V_{DD}, the energy might simply be wasted, or stored in an inefficient battery system.

Dual-rail circuits provide robustness which can help achieve a lower $V_{\mathrm{DD,min}}$. In addition, it is possible to detect when the circuit has completed its computation by using a self-timed approach [3]. Using a weakly-indicating circuit (such as one based on the NCL-X design methodology [4]) with completion detection, we can implement a design which is *early propagative*. Such a design allows flexible performance which is self-adapting to the V_{DD}, and therefore the power availability when coupled with an energy harvester. We call this a *power proportional* design.

A potentially useful side-effect of a power proportional design comes from its ability to adapt to the V_{DD}. In systems where we have high power availability, but lower energy availability (such as battery-powered systems), we can artificially lower the V_{DD}, simulating a power-sparse condition. Operating over a long period of time in this condition leads to low energy usage overall.

In such battery-powered applications, we are interested in operating at the system's minimum energy point (MEP)—the point at which the system consumes the minimum amount of energy per computation. Operating at the MEP will lead to maximized battery life. The amount of energy per computation at the MEP (E_{min}) occurs where the leakage energy of the circuit is equal to its dynamic energy ($E_{\mathrm{leak}} = E_{\mathrm{dyn}}$). E_{min} can be characterised as a function of supply voltage. As a result, we define $V_{\mathrm{DD,MEP}}$ as the supply voltage for which E_{min} is achieved. We are interested specifically in operation where $V_{\mathrm{DD}} < V_{\mathrm{TH}}$, since this is where the MEP tends to lie for complex CMOS designs [5], [6].

In the contrary situation where we have high power and high energy availability (such as in a mains-powered device), we can supply a higher V_{DD}. The power proportional properties of the system will allow it to operate with higher performance, albeit at the expense of greater energy consumption.

Here we have discussed how a power proportional design can operate under three different usage scenarios with no extra design effort: energy harvesting, battery-powered and mains-powered. It is for this reason that we present a 32-bit adder design based on the power proportionality principle.

Section II will introduce the cell library upon which this work is based. Section III will introduce the power proportional adder design and the traditional design against which it will be compared. Section IV shows the results of the designs when benchmarked in a simulation environment. Section V summarises the work carried out and makes suggestions for further work in the area of power proportional design.

II. Cell Library

The cell library in [7] is based on a commercial $65\,\mathrm{nm}$ low-power process. The library uses a novel, full diffusion sizing strategy with $100\,\mathrm{nm}$ transistor lengths. This allows for reduced propagation delay variability which is introduced by process variations when operating in the subthreshold and near-threshold regions. The cells exhibit a larger leakage power than cells designed using a traditional sizing strategy. However, they are proportionally faster for a given V_{DD}, leading to higher performance and a lower energy per cycle.

The library includes the following cells: NAND2, NOR2, INV, AOI22, OAI22. The range of available cells is limited since transistor stacks must be kept to a minimum to ensure

reliable operation in subthreshold. Each cell type is available in a one, two, three and four finger variant. Exceptions are the AOI and OAI cells which do not have the four finger variant due to cell dimension limitations. An increased number of fingers results in reduced propagation delay and increased leakage power. These cell variants give rise to a speed and leakage range for the synthesis tool to work with.

The cells also have stacked variants which are designed to reduce leakage power by pushing transistors into super-cutoff [7]. They are intended to provide stepping stones between regular- and low-V_{TH} cells for use in multi-V_{TH} designs. However, in this work, they have been tested in isolation to determine whether they can give improved energy efficiency. The AOI and OAI cells have an inherent stacked topology and therefore do not have an explicit stacked variant.

III. Adder Design

Since leakage power is increased in the target library compared to libraries with standard sizing, it is important to use circuit design techniques to allow the design to compete in terms of E_{min}. In silicon processes with reduced feature sizes such as the one used in this work, E_{leak} is inherently greater, requiring more effort in circuit design to keep the energy per computation low. For these reasons, a ripple-carry architecture is used for its small logic footprint. A smaller logic footprint leads to a lower E_{leak} from a circuits perspective.

For the 32-bit ripple-carry adder, in the worst case, the carry signal must travel through all 32 full adder blocks before the output is valid. However, due to the carry-propagate and -generate characteristics, it is possible for the output to be valid much sooner than the worst case—after a delay of $\log N$ carry stages on average [3]. We take advantage of this by using an early propagative design based on the NCL-X dual-rail design style [4], [8].

All circuits were synthesized using Synopsys Design Compiler with a combination of fingered variants from the regular-V_{TH} cell library. The tool successfully chose faster cells with more fingers in order to satisfy the critical path delay of 30 µs. Fewer-fingered variants were chosen elsewhere in order to minimize power dissipation.

We present the dual-rail full adder designs upon which the complete 32-bit adder is built from. We also implement a single-rail design using the target cell library in order to make a comparison with traditional designs used in the IoT today.

A. Single-rail

The single-rail design is derived from (1a) and (2a). We remove XORs in order to obtain (1b) and (2b) in terms of simple gates. These forms allow us to use $\left(\overline{\mathtt{a}}\mathtt{b} + \mathtt{a}\overline{\mathtt{b}}\right)$ as a common term between both $\mathtt{sum}$ and $\mathtt{cout}$. From these equations, we obtain the circuit in terms of positive gates as shown in Figure 1. The positive gates are implemented using negative gates followed by an inverter since there are no positive gates in the cell library.

We can simplify the circuit by using De Morgan's theorem to transform the AO gates into OAI gates with their inputs inverted. The result of the transformation is shown in Figure 2.

$$\mathtt{sum} = \mathtt{a} \oplus \mathtt{b} \oplus \mathtt{c} \tag{1a}$$

$$= \left(\overline{\mathtt{a}}\mathtt{b} + \mathtt{a}\overline{\mathtt{b}}\right)\overline{\mathtt{c}} + \left(\overline{\overline{\mathtt{a}}\mathtt{b} + \mathtt{a}\overline{\mathtt{b}}}\right)\mathtt{c} \tag{1b}$$

$$\mathtt{cout} = \mathtt{ab} + (\mathtt{a} \oplus \mathtt{b})\mathtt{c} \tag{2a}$$

$$= \mathtt{ab} + \left(\overline{\mathtt{a}}\mathtt{b} + \mathtt{a}\overline{\mathtt{b}}\right)\mathtt{c} \tag{2b}$$

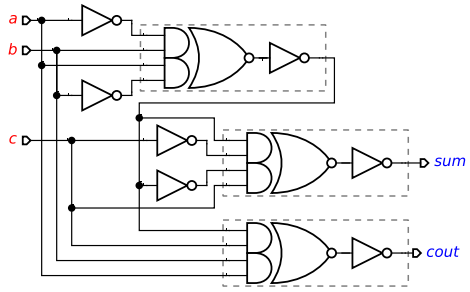

Figure 1. The positive-gate implementation of the full adder.

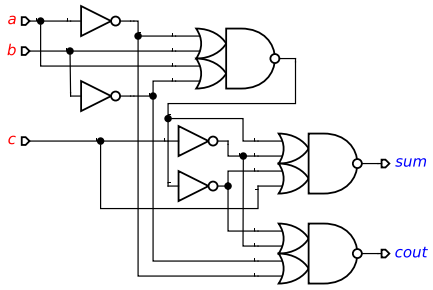

Figure 2. The final circuit for the single-rail full adder.

B. Dual-rail

Equations for the dual-rail design are derived from those of the single-rail design. Starting from (1b) and (2b), all of the inputs and outputs are substituted with their positive dual-rail counterpart, eg. $\mathtt{a} \rightarrow \mathtt{a}_1$, resulting in $\mathtt{a}_1$, $\mathtt{b}_1$, $\mathtt{c}_1$, $\mathtt{sum}_1$ and $\mathtt{cout}_1$. Secondly, all inverted inputs are replaced by the corresponding negative rail input. For example, $\overline{\mathtt{a}_1}$ is replaced by $\mathtt{a}_0$. After noting that $\overline{\mathtt{a}}\mathtt{b} + \mathtt{a}\overline{\mathtt{b}} = \overline{\mathtt{a}}\overline{\mathtt{b}} + \mathtt{ab}$, the result is (3) and (4).

Negative rails are introduced in order to output the complement of the positive rail as per the NCL-X design methodology [4]. To arrive at the equations for the negative rail, every input and output is replaced by the complement rail giving rise to (5) and (6).

$$\mathtt{sum}_1 = (\mathtt{a}_1\mathtt{b}_0 + \mathtt{a}_0\mathtt{b}_1)\mathtt{c}_0 + (\mathtt{a}_1\mathtt{b}_1 + \mathtt{a}_0\mathtt{b}_0)\mathtt{c}_1 \tag{3}$$

$$\mathtt{cout}_1 = \mathtt{a}_1\mathtt{b}_1 + (\mathtt{a}_1 + \mathtt{b}_1)\mathtt{c}_1 \tag{4}$$

$$\mathtt{sum}_0 = (\mathtt{a}_0\mathtt{b}_1 + \mathtt{a}_1\mathtt{b}_0)\mathtt{c}_1 + (\mathtt{a}_0\mathtt{b}_0 + \mathtt{a}_1\mathtt{b}_1)\mathtt{c}_0 \tag{5}$$

$$\mathtt{cout}_0 = \mathtt{a}_0\mathtt{b}_0 + (\mathtt{a}_0 + \mathtt{b}_0)\mathtt{c}_0 \tag{6}$$

In dual-rail encoding, two wires are used to represent a codeword. For a single bit x, the dual-rail encoding consists of the positive and negative rails $\{x_1, x_0\}$. $x = 0$ is encoded as $\{0, 1\}$, and $x = 1$ is encoded as $\{1, 0\}$. $\{0, 0\}$ is used as an empty state, or *spacer*, which separates codewords temporally to allow them to be distinguished from one another. This behaviour can be seen in Figure 3 where the `done` signal acknowledges the receipt of the codeword. Care must be taken to correctly implement spacers in the design, otherwise data hazards could occur [8].

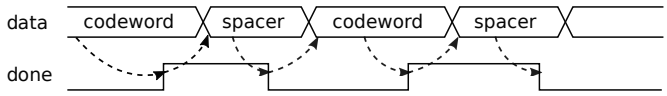

Figure 3. Role of the spacer in the dual-rail circuit.

Figure 4 shows the initial implementation of the dual-rail full adder. The circuit is made up of positive gates as shown by the groupings. These must be implemented as negative gates with a following inverter due to the cells available in the library. If we classify layers of logic separated by positive gates, when logic zero is applied to all primary inputs, all layers will produce a zero since there are no inversions (thinking in terms of positive gates). In this case, we say that the circuit uses all-zeroes spacers—the spacer is $\{0, 0\}$ and the $\{1, 1\}$ state is not allowed.

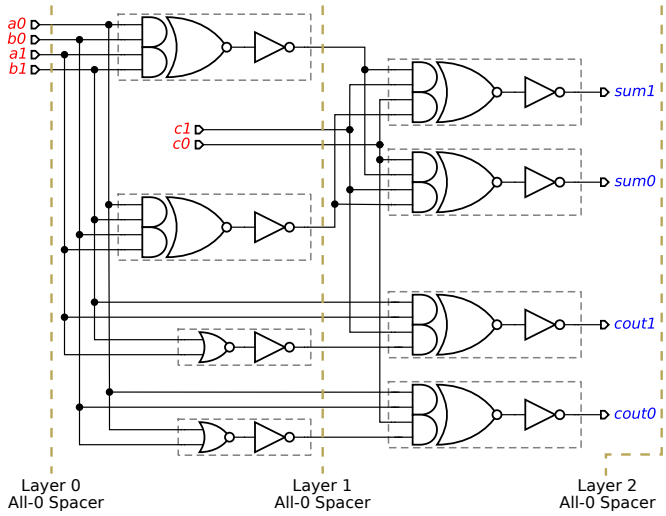

Figure 4. The initial dual-rail full adder design using positive gates.

To improve the circuit, we can transform the AO gates into OAI gates with inverted inputs like in Section III-A. The result is shown in Figure 5 where we have removed some of the double inversions and consequently reduced the number of inverters from eight to six. In the new circuit, for an all-zeroes spacer at layer 0, layer 1 produces an all-ones spacer due to the inverting logic. We now have *spacer inversion* in layer 1 and we use the $\{1, 1\}$ state as a spacer and forbid the $\{0, 0\}$ state. Following on, layer 2 has another spacer inversion and therefore the primary outputs produce all-zeroes spacers. We term this alternation of spacers between logic layers an *alternating spacer* protocol.

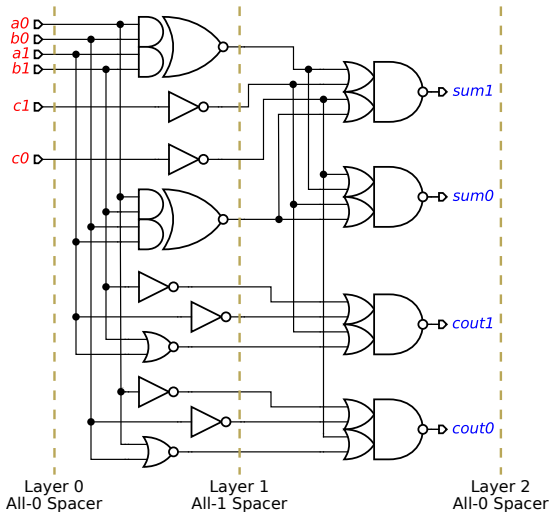

Figure 5. Logic for the dual-rail full adder using an alternating spacer.

Figure 6 shows a further optimized design. Here, `sum1` and `sum0` use all-zeroes spacers, whereas `cout1` and `cout0` use all-ones spacers. To achieve this, the gates directly preceding the $\texttt{cout}_1$ and $\texttt{cout}_0$ primary outputs have been moved into a new layer of logic by the introduction of inverters at their inputs. Consequently, $\texttt{c}_1$ and $\texttt{c}_0$ are moved into layer 1 so that their spacers match $\texttt{cout}_1$ and $\texttt{cout}_0$ from the previous block when the full adders are chained. This optimization removes a further two inverters from the design, reducing the amount of logic and therefore lessening the effect of leakage power on the MEP.

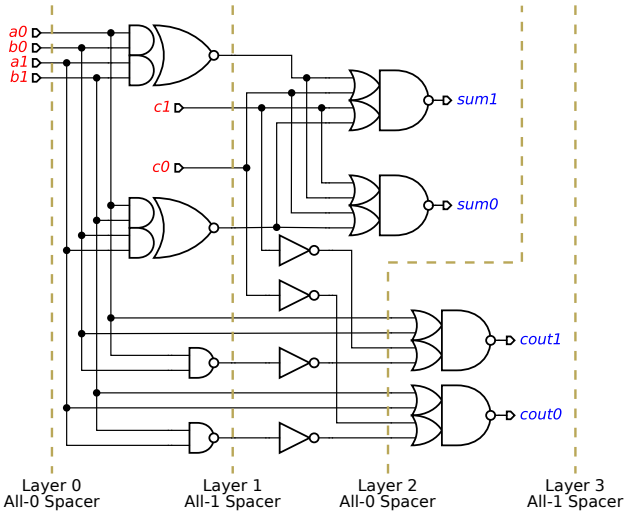

Figure 6. The final optimized logic for the dual-rail full adder.

A completion detection circuit is used in the self-timed dual-rail design to generate a `done` signal which indicates when either the positive- or negative-rail is high for each sum output. When all sum outputs have only one rail asserted, we know

that the computation is complete. This allows the circuit to be early-propagative, gives rise to power proportionality, and minimizes latency. We have implemented completion detection using a cascade of NAND-/NOR-gates as shown in Figure 7. This implementations makes the assumption that adequate time is available for all logic to reset to spacer before a new operand is applied to the primary inputs. This assumption is reasonable in IoT applications where activity factor is low. In order to make the design speed independent (SI) [3] and remove this assumption (eg. in applications with a higher activity factor), a completion detection tree based on a C-element would be required.

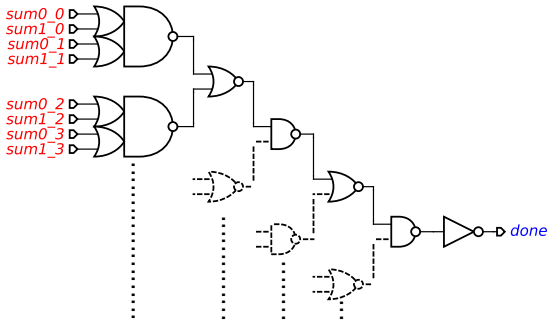

Figure 7. The synthesized completion detection tree. Only a section of the regular tree structure is shown. `sum1_0` refers to bit 0 of $\texttt{sum}_1$.

IV. Results

In order to benchmark the designs, 131 072 randomized 32-bit integers were added together in a digital simulation environment. The designs were synthesized using Synopsys Design Compiler for $V_{\mathrm{DD}} = 0.25\,\mathrm{V}$ and then tested in the range $0.25\,\mathrm{V}$ to $0.6\,\mathrm{V}$. The dual-rail design was tested with a fixed clock period, and also in its self-timed form using the `done` signal from the completion detection circuit to define the period. The single-rail design was tested only with a fixed period clock.

For the single-rail and dual-rail fixed-period implementations, the clock period was set to a value $10\,\%$ slower than the critical path reported by the synthesis tool. This is a reasonable overhead to account for mismatches between the simulated and manufactured design to ensure maximum yield. For the self-timed dual-rail design, the testbench used the `done` signal to supply new operands to the circuit as shown in Figure 8. There is a delay between the output becoming valid and `done` being asserted due to the completion detection tree.

The results produced in simulation were verified in order to ensure no timing violations occurred. The fixed-period dual-rail circuit takes two cycles per computation since it must reset to spacer after each result is produced. This is also true for the self-timed dual-rail design, however the spacer cycle tends to be much faster than the computation cycle as can be seen in Figure 8.

In the self-timed dual-rail implementation, the average computation time was calculated from the total runtime divided

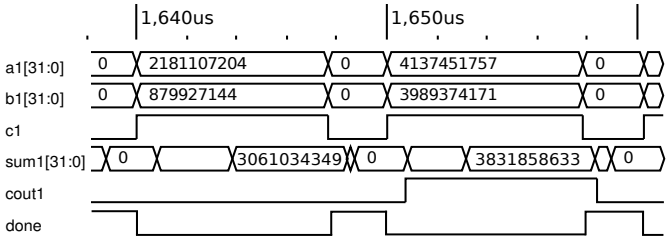

Figure 8. Waveforms from simulation showing the `done` signal reacting to the outputs becoming valid ($\texttt{sum}_1$, $\texttt{cout}_1$).

by the number of computations. Power figures were produced with Synopsys PrimeTime using the value change dump from digital simulation.

Figure 9 shows the energy per computation as a function of V_{DD}. This is calculated using the power-delay product and takes into account the time taken to reset to spacer. The self-timed dual-rail design achieves a lower energy per computation than the fixed-period dual-rail design throughout the tested V_{DD} range. This can be explained by the reduced computation time (see Figure 11) which results in a lower E_{leak}. Both dual-rail implementations consume considerably more energy than the single-rail design due to the increased amount of logic.

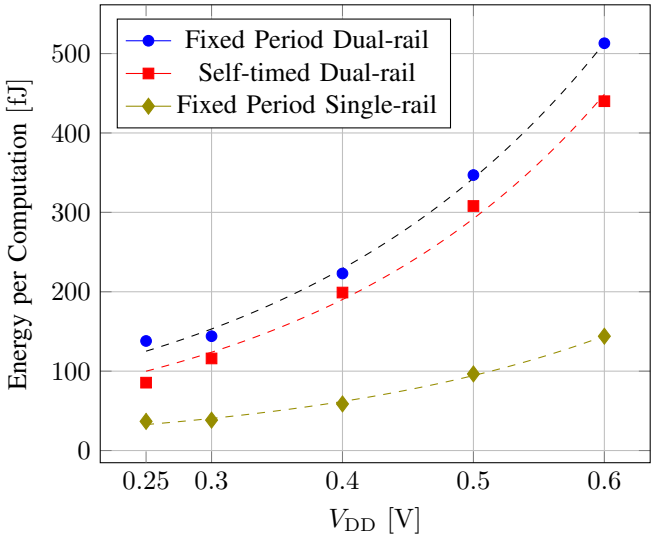

Figure 9. Energy per computation comparison of dual-rail and single-rail implementations.

It is not obvious from these results if $V_{\mathrm{DD,MEP}}$ has been reached. By plotting E_{leak} versus E_{dyn}, we can extrapolate where they will be equal, and thus predict $V_{\mathrm{DD,MEP}}$ and E_{min} for the self-timed dual-rail design. According to Figure 10, this occurs at $V_{\mathrm{DD,MEP}} = 0.09\,\mathrm{V}$ where $E_{\mathrm{leak}} = E_{\mathrm{dyn}} = 2.54\,\mathrm{pJ}$. Since $E_{\mathrm{min}} = E_{\mathrm{leak}} + E_{\mathrm{dyn}}$, this leads to a predicted $E_{\mathrm{min}} = 5.08\,\mathrm{pJ}$. This is the theoretical E_{min} for the self-timed dual-rail implementation assuming that the cells function correctly at this V_{DD}.

The average computation time for the designs can be

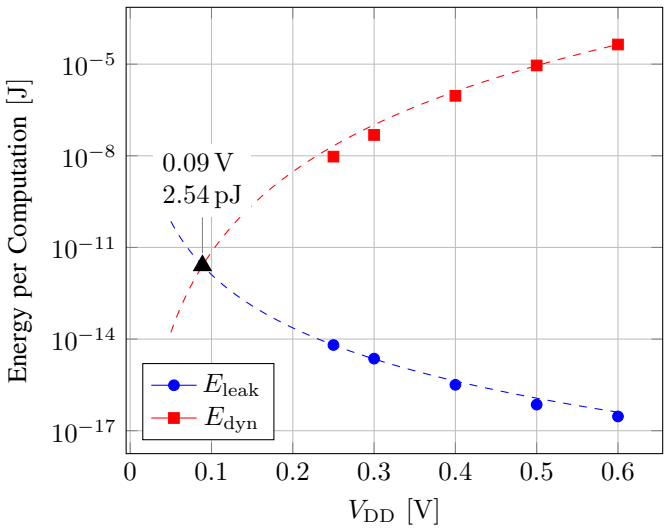

Figure 10. Leakage and dynamic energy for the self-timed dual-rail implementation.

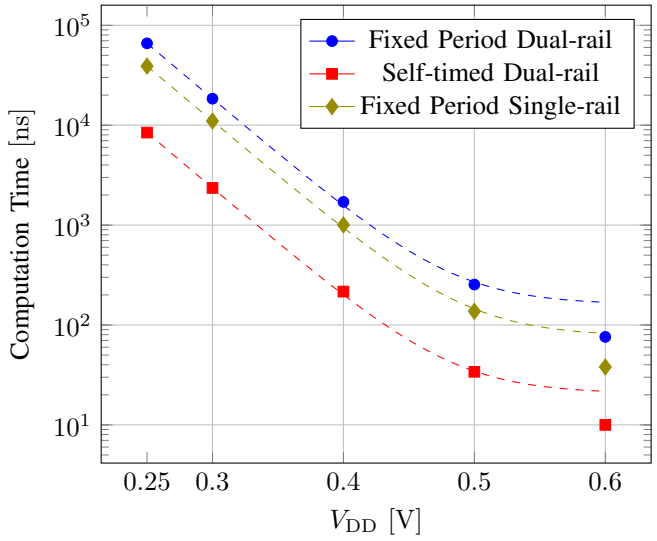

Figure 11. Computation time of dual-rail and single-rail implementations. The results include the reset-to-spacer time for the fixed-period and self-timed dual-rail implementations.

seen in Figure 11. The self-timed design shows a significant performance improvement compared to both dual-rail fixed, and single-rail designs. It also has the added advantage that the performance inherently adapts without the need for complex dynamic voltage and frequency scaling (DVFS).

Figure 12 shows the effects of using only stacked cells in the self-time dual-rail design. At 0.25 V, the leakage power is reduced from 757 pW to 613 pW. The trend continues for the rest of the voltage range; however, the energy per computation is increased. This is due to the vastly increased computation period, resulting in leakage power being consumed over a longer period of time. It should be noted however, that the activity factor in this benchmark is 100 %. The stacked cells

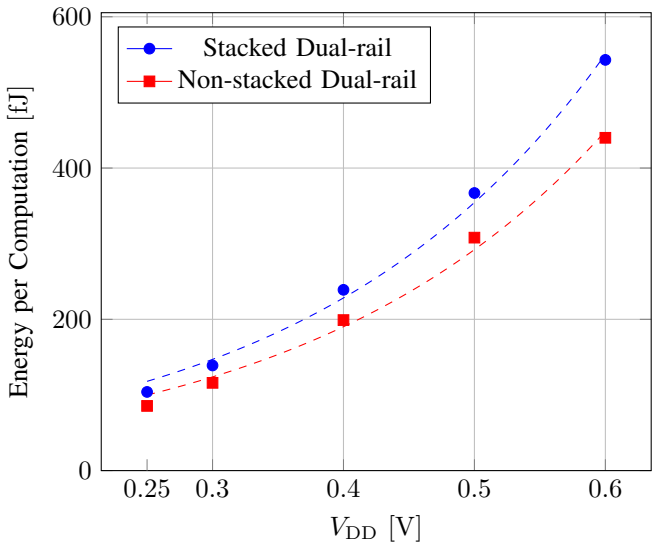

Figure 12. Energy per computation comparison of self-time dual-rail designs.

are predicted to give a benefit in situations where activity factor is low—ie. the circuit is idle for a long period of time—as is the case in many IoT applications.

V. Conclusions

In this work we have presented a self-timed dual-rail design for a 32-bit ripple carry adder suitable for IoT applications. We have compared its performance and energy consumption with a fixed-period dual-rail version, and with a traditional fixed-period single-rail equivalent. The self-timed dual-rail design shows improved performance over the fixed-period dual-rail design. It also benefits from inherent power proportional performance. In contrast, the fixed-period single-rail design would need to implement complex DVFS to achieve similar power proportionality.

We have also shown that the use of stacked-transistor cells does not lead to improved energy efficiency with this combination of design, cell library, and benchmark, since the increased propagation delay outweighs the power savings. In future work, we will explore how the MEP of the stacked-cell design changes depending on activity factor.

We will simulate the design using a lower V_{DD} in order to find the MEP of the self-timed dual-rail design experimentally and confirm if the design is functional at this V_{DD}. This will require additional analogue simulations and characterisation of the target cell library.

Furthermore, we plan to investigate completion detection using a C-element approach which will make the design speed independent. This will require the addition of a C-element to the cell library. Simulations will be run with variability in cell delays to verify the robustness of the design.

We plan to compare the design from this paper with that using a cell library with a traditional sizing strategy for subthreshold.

ACKNOWLEDGMENT

This work was supported by EPSRC and ARM.

REFERENCES

[1] X.-D. Do, S.-K. Han, and S.-G. Lee, "Optimization of piezoelectric energy harvesting systems by using a MPPT method," in *2014 IEEE 5th Int. Conf. Commun. Electron.*, Jul., pp. 309–312.

[2] A. Montecucco and A. R. Knox, "Maximum Power Point Tracking Converter Based on the Open-Circuit Voltage Method for Thermoelectric Generators," *IEEE Trans. Power Electron.*, vol. 30, no. 2, pp. 828–839, Feb. 2015.

[3] J. Sparsø and S. Furber, *Principles of Asynchronous Design: A Systems Perspective*. Kluwer Academic Publishers, 2001.

[4] A. Kondratyev and K. Lwin, "Design of asynchronous circuits using synchronous CAD tools," *IEEE Des. Test Comput.*, vol. 19, no. 4, pp. 107–117, Jul. 2002.

[5] A. Wang and A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 310–319, Jan. 2005.

[6] J. Myers *et al.*, "A subthreshold ARM cortex-M0+ subsystem in 65 nm CMOS for WSN applications with 14 Power Domains, 10T SRAM, and integrated voltage regulator," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 31–44, Jan. 2016.

[7] J. Morris *et al.*, "Unconventional Layout Techniques for a High Performance, Low Variability Subthreshold Standard Cell Library," in *Proc. ISVLSI 2017*, Bochum, Germany, Jul. 2017, p. TBA.

[8] D. Sokolov, "Automated synthesis of asynchronous circuits using direct mapping for control and data paths," Ph.D. dissertation, Newcastle University, Jan. 2006.