Exploring Error Correction Circuits on RISC-V based Systems for Space Applications

Nazim Altar Koca*†, Chip Hong Chang†, Anh Tuan Do* and Vishnu P. Nambiar*
* Institute of Microelectronics, A*STAR (Agency for Science, Technology and Research), Singapore
† School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

***Abstract*—RISC-V systems are becoming increasingly adopted in space applications. SRAM (Static Random-Access Memory) data memory is a critical component that occupies a large portion of the processor peripheral system. SRAM is vulnerable to single-event upsets (SEUs). Existing studies mainly considered Hamming error correction codes (ECCs) for memory protection in RISC-V processor. In this paper, we explore different ECCs as well as the triple modular redundancy (TMR) as solutions to mitigate SEU effects on SRAMs for RISC-V system. To overcome the area and power overhead of TMR with higher fault tolerance than ECCs, we propose a dual modular redundancy (DMR) with ECC memory protection scheme. We conduct a comprehensive error analysis to evaluate different fault-tolerant designs under various radiation attack scenarios, utilizing real-world data to devise the fault-injection campaign. An efficient scheduling for the self-refresh operation of SRAMs is proposed to prevent the error accumulation. The proposed DMR with ECC design reduces the power and area overhead of TMR by 28% and 11% respectively and improve the error resilience of the SRAM significantly compared with Hsiao and Hamming ECC schemes.**

I. Introduction

In recent years, there has been an increase in the adoption of RISC-V architecture [1] in space applications [2]. Characterized by its modular and open-source design, RISC-V comprises a collection of standard instruction sets, with a base set providing basic functionality, and optional extensions to suit different applications. The modularity of RISC-V allows system designers to select only the extensions required for their specific task, thereby optimizing the processor's performance and power consumption. This makes it a viable solution for space-grade processor systems that require adaptability and reliability. The surge of RISC-V in space applications has stimulated research interest into improving fault-tolerance in harsh radiation environments.

A critical challenge that processor systems confront in space applications is their susceptibility to single-event upsets (SEUs) [3], which are becoming increasingly more probable as transistor technology shrinks [4]. SEUs are transient, radiation-induced faults that can be inflicted upon electronic components, including memory devices and processor registers. When a high-energy particle strikes a sensitive region within a semiconductor device, it can deposit enough energy to cause a temporary change in state, resulting in bit flips or data corruption. SEUs are highly unpredictable and can occur at irregular intervals, posing a severe threat to system memory data integrity, and leading to corrupted processor instructions.

Static Random-Access Memories (SRAMs) are the most susceptible to SEU effects due to their manufacturing technology and large footprint [5]. There are two types of mitigation schemes commonly employed; space and information redundancy. Space redundancy replicates the memory units to perform the same operation on the same inputs and the outputs are determined by majority voting. Information redundancy adds extra parity bits to the data which allows the use of error correction codes (ECC) to restore information. There are pros and cons to both techniques due to the trade-off between fault tolerance and hardware cost. Unfortunately, there is a lack of analysis between these schemes when used in RISC-V memory systems for protection against SEU in a realistic radiation environment in published prior works.

In recent years, several fault-tolerant RISC-V systems have been proposed in previous works. In [6] and [7], the authors protected various memory elements with the Hamming Single-Error-Correction (SEC) scheme. However, system data memory protection, which is the largest and most vulnerable memory unit, was omitted in these two designs. In [8] and [9], data memory was included in the solutions of RISC-V systems. The SRAM structures were protected with Hamming Single-Error-Correction and Double-Error-Detection (SEC-DED). All these recent fault-tolerant RISC-V systems considered only Hamming ECC, which is inadequate for high bit error rate space applications.

In this paper, we evaluate four different fault-tolerant designs for memory structures; Hamming ECC, Hsiao ECC, Dual Modular Redundancy (DMR) with ECC and Triple Modular Redundancy (TMR), where DMR with ECC is a new design proposed in this paper. We synthesis these designs in 40nm CMOS technology node to make a fair comparison between their area, delay and power consumption. Our evaluation shows that Hsiao ECC can reduce the latency of Hamming ECC by 8.5%. Our proposed DMR with ECC design can reduce the power and area overheads of TMR by 28% and 11%, respectively. Furthermore, an efficient interrupt driven self-refresh scheme is proposed to enhance the system reliability against prolonged irradiation. Finally, we compare our results with related works on implementations of RISC-V memory protection that were reported in the literature.

II. Fault-Tolerant Architectures

To mitigate the effects of the soft errors on memory blocks, four solutions are studied in this section; Hamming ECC, Hsiao ECC, TMR, and DMR with ECC.

A. Error Correcting Codes (ECCs)

Hamming code is a widely used ECC technique known for its simplicity and effectiveness in memory protection [10]. It is a systematic linear block code that adds redundancy to the

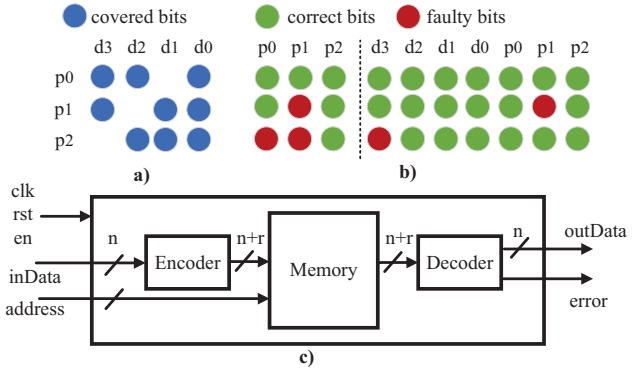


Fig. 1. a) ECC parity bit generation, b) error detection and location when there is a parity mismatch, c) block diagram of SRAM with ECC protection.

original data to enable error detection and correction. Hamming ECC consists of two processes, encoding and decoding. The encoding process with Hamming ECC involves adding the appropriate parity bits to the original data. The parity bits are first calculated by applying an XOR function to the data bits they cover, as illustrated by the (7,4) Hamming in Fig. 1(a). The parity bits are appended to the data bits to form an expanded data word (7-bit in this case). The decoding process with Hamming ECC is designed to identify and correct errors within the received expanded data. The parity bits are first calculated from the received data by applying the same formulation as in the encoding process. Then, the calculated parity bits are compared with the received parity bits at the specific location of the received data. If a difference is detected (as in the second and third row of Fig. 1(b)), the position of the erroneous bit is located and corrected. If multiple errors are detected and cannot be corrected, an error flag is set. Implementation details can be found in previous works [11].

Hsiao code [12] is founded on the same principle as Hamming code. It is an optimized version of Hamming by reconstructing the calculation of parity bits to use less logic levels during the decoding process. As a result, the critical path delay is reduced without increasing the area overhead. The top view of a memory structure equipped with the ECC logic (which are the same for both Hamming and Hsiao) is given in Fig. 1(c). The n-bit input data passes through the encoder logic before they are written into the memory. The output data of the encoder block has $(n+r)$ bits after adding the r redundant parity bits. Upon receiving the read command, the memory outputs an $(n+r)$-bit data to the decoder block. The decoder outputs an n-bit data and an error signal to inform the processor if errors are detected on the memory.

B. Triple Modular Redundancy (TMR)

Fig. 2(a) shows the overall structure of TMR. There are three identical SRAM memories connected to the same input data and address busses. Upon receiving the read command, the three SRAM outputs go through a majority voting circuit to determine the $outData$. The voting mechanism compares the outputs of the three modules as illustrated in Fig. 2(b). An error signal is flagged by the error detector logic if any of the inputs is different from the others. TMR systems can correct

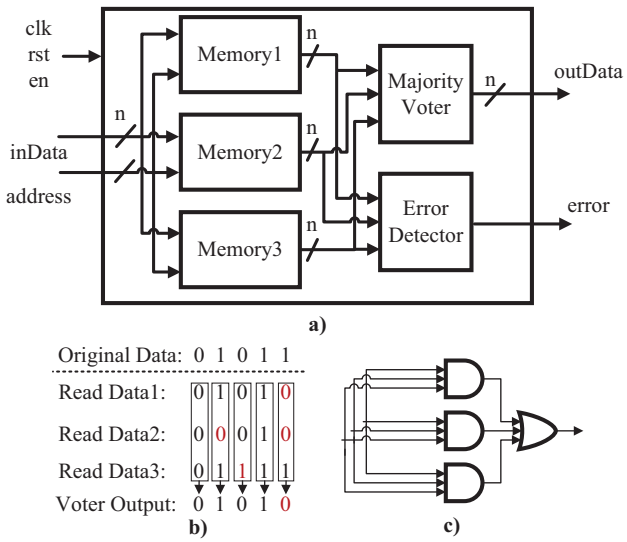


Fig. 2. a) Block diagram of SRAM with TMR protection, b) an example of TMR logic on 5-bit data, c) 3-input majority voter circuit.

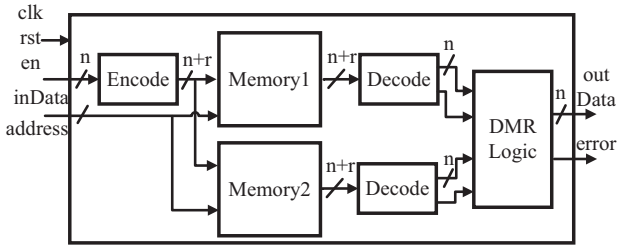


Fig. 3. DMR Logic with ECC for SRAM Memory.

single-bit and multi-bit errors as long as the errors do not occur simultaneously in two or more replicas. The system considers the output that has a majority vote (i.e., at least two out of three modules) as the valid result. If the errors occur at the same location of different SRAM outputs, an incorrect result will be inferred by the voter (example shown in Fig. 2(b)). The circuit structure of the 3-input majority voter is given in Fig. 2(c). TMR systems incurs a high area overhead by triplicating the memory block. However, it is much more resilient to transient faults and has much shorter critical path than ECC logic, which are desirable features in real-time applications.

C. Dual Modular Redundancy (DMR) with ECC

ECC is a compact solution but it is more vulnerable to multi-bit errors. On the other hand, TMR is a suitable choice for applications in harsh conditions but it has high area overhead. To strike a middle ground, we propose DMR with ECC as an alternative solution that is more area- and power-efficient than TMR, while being more error-tolerant than just ECC. The main problem with DMR is it is not possible to know which data is correct if the two outputs are different. We resolve this problem by combining ECC logic with DMR structure to determine which memory output is corrupted when the data is inflicted by soft errors. Fig. 3 shows the block diagram of the proposed DMR with ECC design. The same encoder block is used to encode the parity bits into the data. The same encoded data is stored at two different SRAMs. Each SRAM has its

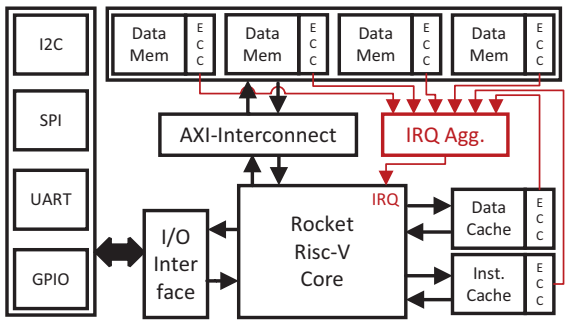


Fig. 4. System-level Overview with ECC Memory Protection.

own decoder block. After the decoder block, the n-bit data and the corresponding error signals are fed into the DMR logic block. The latter selects the decoder output data according to the error signals. If there is no error on the first memory output, it is directed to the $outData$. Otherwise, the second memory error output is checked. If there is no error, the second memory data output is selected. If both memory outputs are erroneous, then an $error$ flag is raised to inform the CPU. This structure improves the error resilience of the system significantly compared to ECC only solution and reduces the hardware overhead substantially compared to TMR. However, it has a longer delay as both the ECC decoder block and the DMR logic block fall on the same critical path.

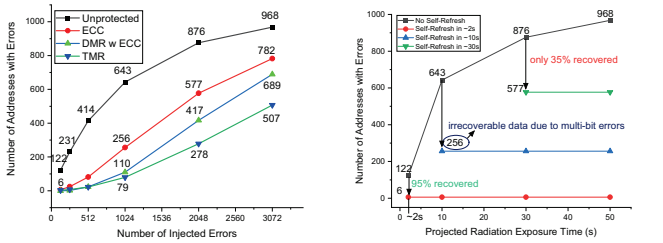
D. Fault-tolerant RISC-V System

Fig. 4 depicts the overall RISC-V system with SRAM memory protection. The system includes a 32-bit RISC-V core, data and instruction caches, input-output interfaces (e.g., GPIO, UART, etc.) and a main data memory connected to the core with AXI interface. All the SRAM memories in the system including the main memory and caches can be protected by any of the aforementioned fault-tolerant schemes. While the address and data busses are directly connected to the core, the error signals of the SRAM protection circuits are routed to the interrupt aggregator (IRQ Agg.) to generate the interrupt signal to the core. Upon detecting the soft errors, the CPU performs a self-refresh to prevent error accumulation as explained in the next section.

III. Experimental Results

A. Fault Injection Analysis

This section evaluates the resilience of the aforementioned techniques in the face of single-event upsets (SEUs). Previous works tested their designs under weak error rate assumptions. However, when creating a behavioural model to simulate SEUs in SRAM, it is important to have a more realistic baseline. According to [13], a proton radiation source with a minimum energy of 20 MeV should be considered for soft error test. In [5], the authors exposed their SRAM chip to accelerated proton radiation with an energy level of 39.38 MeV. We have taken this experiment as our reference as it provides a realistic test case. According to the results of [5], the bit-flips caused by the particles are uniformly distributed across the SRAM,



(a) (b)

Fig. 5. Total number of addresses with errors after a) different level of protection and fault injection b) different self-refresh schedule on ECC.

but the distribution of the number of error bits for a given address is relatively Gaussian. While the number of errors increases almost linearly with the duration of the radiation (time), it increases exponentially with the energy level of the particles. We model the soft error based on these error probability distributions at 39.38 MeV proton radiation energy, and develop an error injection module to emulate the bit flips induced by SEU when reading the data from the SRAM.

Fig. 5(a) summarises the results of the error injection analysis. We present 6 error cases (128, 256, 512, 1024, 2048 and 3072 injected bit errors) and the fault-tolerance of the designs under these error cases. We simulated a 4KB SRAM which stores 1024 words as in [5]. The 3072-Error case causes 968 corrupted data on unprotected memory (the last sample point on the black line of the figure), which approximates a 50s, 39.38 MeV radiation exposure [5]. The number annotated on each sample point of the figure indicate the number of addresses with uncorrectable errors after the error correction logic. Fig. 5(a) shows that ECC has the worst error-tolerance among all protected schemes. DMR with ECC performs much better than ECC-only design with comparable error-tolerance to TMR up to 1024-Error level. At high error injection rates, the advantage of TMR becomes apparent.

When exposed to high and prolonged irradiation, all the protection methods fail to recover at least half of the errors. This is due to the accumulation of soft errors, which can be prevented by applying a self-refresh operation periodically [9] [5]. Self-refresh means reading all the data in the memory and writing it back after applying the error correction logic. However, this operation is time consuming and can interrupt the processor's schedule if it is applied continuously. Therefore, we propose an efficient scheduling of self-refresh operation.

Fig. 5(b) shows the statistics that motivate our proposed interrupt driven self-refresh scheduling scheme. If the processor waits for long before applying the self-refresh, most of the data will be corrupted. In the case of 30s radiation attack, 577 out of a total of 1024 data are uncorrected. However, if the processor performs the refresh earlier, for example, for the 2s radiation attack case, the uncorrected data rate can be kept below 1% (6 out of 1024). Two actions can be taken to improve the error rate and efficiency trade-off. If we can estimate or track the error profile, it is possible to schedule the self-refresh operation after a specific elapse time (1s for our case) of an

TABLE I
SYNTHESIS RESULTS OF PROPOSED DESIGNS ON 40NM AT 100 MHZ

Design	Total Area (μm^2)	Power (mW)	Critical Path Delay (ns)
Unprotected	205941	15.577	1.759
ECC - Hsiao	273067	16.863	2.775
ECC - Hamming	273029	16.895	3.031
DMR with ECC	546190	33.501	3.288
TMR	617975	46.330	2.074

error detected in the memory. All four protected designs have error output signals to alert the processor via the interrupt line. After receiving the interrupt, the processor can apply self-refreshing in the interrupt handler by initiating a timer interrupt. Otherwise, an error threshold can be incorporated in the interrupt handler to initiate self-refreshing by counting the number of errors until a pre-determined value such as 1% of the total addresses. Self-refreshing can be done in software with a burst read and write command, or an extra module can be added in the SRAM wrapper to do it completely in hardware.

B. Area, Power and Critical Path Delay

We synthesized our designs written in Verilog on Cadence Genus tools using UMC 40 nm technology cell library at 100 MHz. Table I summarizes the synthesis results. The unprotected design corresponds to a 64KB SRAM memory block without any protection. As shown in the table, Hsiao and Hamming have similar area and power consumption but the critical delay path of Hsiao is 8.5% shorter than that of Hamming. For larger bitwidth data, Hsiao is more area efficient according to [14]. TMR design has $3\times$ higher areas over the unprotected design as expected. TMR has the highest frequency among the fault-tolerant designs. DMR with ECC reduces the area overhead to $2.6\times$ over the unprotected design at the expense of a longer critical path delay. It also reduces the power consumption of TMR by 28%.

C. Comparison with Related Works

We integrated our fault-tolerant memory designs with Rocket RISC-V core [15] which has 256KB SRAM data memory. Table II summarizes different memory protection solutions on RISC-V systems. While [9] applies the protection on all memory arrays in the system, only core registers in [7] and instruction memory (IM) in [6] are protected. Other works only considered Hamming ECC which is less error-resilience compared to TMR and DMR-ECC, and slower than Hsiao ECC. Unlike our evaluation, which is based on more stringent soft errors probabilistic model from realistic accelerated proton radiation energy experiments, the fault-tolerant designs of these three works were tested with more lenient fault injection rate. The reliability of these methods ranked according to their fault-tolerance performance in Fig. 5(a) are TMR, DMR-ECC and followed by all ECC solutions.

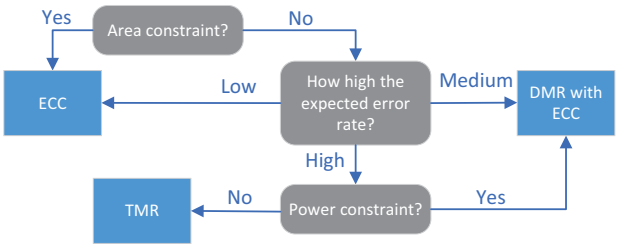


Fig. 6. Recommendation Flowchart on Memory Protection Methods.

TABLE II
COMPARISON WITH RISC-V MEMORY PROTECTION DESIGNS

Design /Technology	Protection Type /Target Memory	Hardware Overhead	Fault Injection	Reliability
DTIS'20 [7] /FPGA ZC7020	Hamming ECC /Core Registers	20% more FF 15% more LUT	Random 1-fault /10-fault	Weak
ARCS'20 [9] /GF 22nm	Hamming ECC /All Memory	6% more Logic 40% more Mem	Only 1-bit /2-bit faults	Weak
DFT'22 [6] /FPGA A7-100T	Hamming ECC /Instruction Mem	3% more FF 11% more LUT	1,2,3-bit faults	Weak
Proposed I /UMC 40nm	Hsiao ECC /Main Data Mem	14% more Area	Random Multi-bit	Weak
Proposed II /UMC 40nm	DMR with ECC /Main Data Mem	73% more Area	Random Multi-bit	Medium
Proposed III /UMC 40nm	TMR /Main Data Mem	89% more Area	Random Multi-bit	Strong

We also listed the system level area overheads over the unprotected design in the third column of Table II. As can be seen, the Hsiao ECC design (proposed I) introduces 14% area overhead similar to Hamming ECC. The area overheads due to DMR with ECC (proposed II) and TMR (proposed III) are 73% and 89%, respectively. Based on these results, a simple flowchart is provided in Fig. 6 to guide the designers in selecting the appropriate fault-tolerance memory design based on the error-tolerance performance and hardware cost.

IV. CONCLUSION

Area- and power-efficient fault-tolerant memory protection in processor based systems with adequate error reconciliation capability under harsh radiation environments are critical for space applications. This paper evaluated the reliability, hardware costs and performance of RISC-V system memory protection using Hamming ECC, Hsiao ECC, TMR and our proposed DMR with ECC schemes. Through a realistic fault injection model based on radiation measurements from previous works, the performance gap between TMR and ECC is found to be apparent, which outlines the need for a middle ground solution. Our proposed DMR with ECC fills this gap by striking a middle ground between error correction capability and hardware budget. It reduces the power and area overheads of TMR by 28% and 11%, respectively. We also propose an interrupt-driven self-refresh solution to efficiently resolve the error accumulation problem, which is applicable to all the investigated schemes.

REFERENCES

[1] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanović, "The risc-v instruction set manual, volume i: Base user-level isa," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2011-62, May 2011. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-62.html

[2] S. Di Mascio, A. Menicucci, G. Furano, C. Monteleone, and M. Ottavi, "The case for risc-v in space," in *Applications in Electronics Pervading Industry, Environment and Society*, S. Saponara and A. De Gloria, Eds. Cham: Springer International Publishing, 2019, pp. 319–325.

[3] F. Wang and V. D. Agrawal, "Single event upset: An embedded tutorial," in *21st International Conference on VLSI Design (VLSID 2008)*, 2008, pp. 429–434.

[4] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[5] M. S. M. Siddiqui, S. Ruchi, L. Van Le, T. Yoo, I.-J. Chang, and T. T.-H. Kim, "Sram radiation hardening through self-refresh operation and error correction," *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 2, pp. 468–474, 2020.

[6] E. B. Annink, G. Rauwerda, E. Hakkennes, A. Menicucci, S. D. Mascio, G. Furano, and M. Ottavi, "Preventing soft errors and hardware trojans in risc-v cores," in *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2022.

[7] D. A. Santos, L. M. Luza, C. A. Zeferino, L. Dilillo, and D. R. Melo, "A low-cost fault-tolerant risc-v processor for space systems," in *2020 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2020.

[8] S. Gupta, N. Gala, G. S. Madhusudan, and V. Kamakoti, "Shakti-f: A fault tolerant microprocessor architecture," in *2015 IEEE 24th Asian Test Symposium (ATS)*, 2015, pp. 163–168.

[9] A. Dörflinger, Y. Guan, S. Michalik, S. Michalik, J. Naghmouchi, and H. Michalik, "Ecc memory for fault tolerant risc-v processors," in *Architecture of Computing Systems – ARCS 2020: 33rd International Conference Proceedings*, 2020, p. 44–55.

[10] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[11] S. Tam, "Single error correction and double error detection," in *Xilinx Application Notes*, 2006.

[12] M. Y. Hsiao, "A class of optimal minimum odd-weight-column sec-ded codes," *IBM Journal of Research and Development*, vol. 14, no. 4, pp. 395–401, 1970.

[13] J. R. Schwank, M. R. Shaneyfelt, and P. E. Dodd, "Radiation hardness assurance testing of microelectronic devices and integrated circuits: Test guideline for proton and heavy ion single-event effects," *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 2101–2118, 2013.

[14] G. Tshagharyan, G. Harutyunyan, S. Shoukourian, and Y. Zorian, "Experimental study on hamming and hsiao codes in the context of embedded applications," in *2017 IEEE East-West Design & Test Symposium (EWDTS)*, 2017, pp. 1–4.

[15] K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, "The rocket chip generator," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, Apr 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html