Multi Fluxon Storage and its Implications for Microprocessor Design

Naveen K Katam, Haipeng Zha, M Pedram and M Annavaram
Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, USA, 90089
E-mail: {nkatam, hzha, pedram, annavara}@usc.edu

Abstract. One of the main challenges for single flux quantum (SFQ) technology to successfully implement a microprocessor is to have a compact and robust on-chip memory that can be used for register files and cache memory. In this context, we designed high capacity destructive readout (HC-DRO) cells. The design is based on our insight that each cell can store more than one SFQ pulse with the same area footprint of a regular DRO cell. In our current design, we demonstrate a single HC-DRO cell that can store up to three SFQ pulses thereby enabling us to store the equivalent of two bits of memory in a single cell. We have designed a register file (RF) architecture for a microprocessor and used the HC-DRO cells to reduce the Josephson junction count required for its implementation. The RF architecture presented in this work can also be used for implementing cache memories. We also demonstrate the benefits of doubling the storage density with HC-DRO cells by designing a 2-bit branch predictor circuit block for a 5-stage pipeline microprocessor that dramatically improves the cycles per instruction metric of an SFQ based microprocessor. All the design details are presented in the paper by verifying the design concepts through JSIM simulations and Verilog simulations.

1. Introduction
As CMOS technology scaling slows, the search for post-CMOS technologies has intensified. There is an increasing demand for energy-efficient computing from applications such as high-performance simulations, machine learning algorithms, that require beyond petaflops of computing. Superconducting computing based on the Josephson Effect has the potential to meet these demands. Josephson junctions (JJs) switch quickly (1 ps) and dissipate little switching energy (10−19 Joules) at low temperatures. Likharev et al. [1] introduced Rapid Single Flux Quantum (RSFQ) technology, which uses quantized voltage pulses in digital data generation, reproduction, amplification, memorization, and processing. They demonstrated that RSFQ circuits are functional at operating frequency of up to 770 GHz. Many follow on techniques further reduce the power and energy consumption of SFQ logic circuit realizations [2].

One of the impediments for the broader adoption of SFQ designs in microprocessors is the on-chip memory capacity. SFQ memories are essentially built as flip-flop-like designs and hence memory density is quite low compared to SRAM. In current generation microprocessors, on-chip memory has a wide range of uses that go beyond cache storage. Large physical register file, branch predictor history tables, and prefetching buffers are performance-critical structures that rely on large memory availability.
SFQ provides two different reliable memory designs currently: a destructive readout (DRO) cell and a non-destructive readout (NDRO) cell. Since SFQ cells are based on magnetic flux pulses once a memory cell is read, by default the pulse is released into the following block and the memory cell is reset. Additional effort must be expended to restore the pulse once it is read. As such, NDRO cells are usually 3X larger than DRO cells.

Each DRO or NDRO cell in current designs stores a single pulse. The presence or absence of a pulse is treated as single-bit binary information. In this work, we propose a mechanism to double the density of DRO memory cell by allowing each cell to store up to three SFQ pulses without increasing the footprint of the DRO cell. This design requires a new read-out circuit to decode the two-bit value stored in the SFQ cell. Similarly, a new write circuit design is needed to encode the two-bit value in the cell. We first present dual-bit storage architecture, along with the read and write circuit design. We call this architecture HC-DRO (high capacity-DRO). As a demonstration of how HC-DRO can be incorporated in current microprocessor designs, we design a 2-bit branch predictor that uses a single column of HC-DRO cells to store the branch history. We then design a register file architecture that uses the HC-DRO cells to store the content. The goal of these two demonstrations is to show how the HC-DRO can be integrated into a traditional pipelined microprocessor design. Hence, the emphasis is on architecture integration rather than claiming the obvious performance advantages of higher density memory.

The primary contributions of this paper are as follows:

- We present the HC-DRO cell design and show how the cell data can be encoded and decoded using a flow-clocking scheme [3].
- We present a register file design that uses HC-DRO cells. We show how data must be encoded into 2-bit chunks for storage in HC-DRO cells. We also show the decoding process to read the compact HC-DRO representation into a wider data format.
- We present a 2-bit branch predictor that uses HC-DRO cells for storing the branch history. We present the read operation that decodes the 2-bit counter value stored in a single HC-DRO cell. We also present the encoding process that updates the counter value.
- Both the branch prediction counters and register files must preserve their content after a read operation. Since our design uses HC-DRO cells that destroy the content after each read we present a unique approach to re-store the value after each read. We use a single shared NDRO cell that is shared across an entire column of HC-DRO cells to enable a low-cost approach to preserve the HC-DRO cell contents.

2. Context of Microprocessor Design
In this section, we provide a brief overview of the architecture of the 5-stage microprocessor, which is our initial target to integrate the HC-DRO design. We use the notion of a pipeline stage as it is used in CMOS designs for simplicity of explanation. In SFQ designs, each such pipeline stage is in turn pipelined at a gate-level. Since SFQ based microprocessor designs are in their infancy we decided to integrate our design within a simpler in-order 5-stage pipeline that uses a 2-bit branch predictor [4]. Our design is illustrated in the Figure 1.

The first stage is the fetch stage which reads an instruction using the program counter (PC) register as the address pointer. Modern microprocessors access a branch predictor using a hash of the PC register to speculatively load the next PC. Note that at the time of fetch it is not even known whether an instruction is a branch instruction or otherwise. The instruction type determination is made only after decode. Note that even in-order microprocessors require a branch predictor to fetch, and decode from the predicted path. Hence, waiting until the decode to access branch predictor is impractical since even CMOS-based microprocessors may require a few cycles before an instruction is decoded, and SFQ based designs may require even more cycles to reach decode stage due to gate-level pipelining.
The 2-bit branch predictor design consists of 2-bits of storage per each entry. The 2-bit storage acts as a saturating up-down counter. The branch prediction entry is incremented if the prediction is true or else the entry is decremented. In our processor design, every cycle the branch predictor and a branch target buffer are accessed concurrently using the current PC. If the branch is predicted not-taken the PC is simply incremented by 4 (all our instructions are 4-bytes wide). If the branch is predicted taken the next PC is fetched from a branch target buffer which fills the next PC.

Instructions from the predicted PC are speculatively fetched and decoded, and even the register operands may be speculatively read. But no execution is permitted until the predicted path is resolved. The branch predictor is read once for every instruction fetched and the counter is updated when a branch instruction is resolved. The counter value must be preserved after a read, so the counter value may be incremented or decremented later.

Once an instruction completes the decode it may access the register file to read the data during the execution stage. Any load/store operations may access memory in the memory stage. All other instructions write back the computed results back to the register file. The register read process must preserve the data after the data is read since the register file is based on HC-DRO (not NDRO) cells.

3. DRO cell with multi fluxon storage

We now describe our HC-DRO design that stores multiple fluxons in a single superconducting loop.

3.1. Design and functionality

Destructive read-out (DRO) cell [1] of single flux quantum (SFQ) technology is one of the most important building blocks for superconducting circuits which can be used as a memory element for storing the SFQ pulses. It is also used as a buffer cell for synchronizing the signals [5] in a circuit. DRO cell is also known sometimes as an RS-Flipflop or D-Flipflop. Fig. 2(a) shows the schematic of a regular DRO cell that receives an SFQ pulse at input \( D \) and stores it in the superconducting loop \( J1-L2-J2 \) if it does not already have a fluxon (SFQ pulse) stored in it. If the \( J1-L1-J2 \) loop already has a fluxon stored in it, the incoming pulse is dissipated through the buffer junction \( J0 \). The stored fluxon is read by input \( CLK \) which resets the superconducting loop subsequently resulting in an SFQ pulse at the output \( Q \). Each cell read is destructive since

Figure 1. A simple representation of traditional five stage pipeline CPU. Compenents modified for HC-DRO usage are shown in 3D-boxes.
Figure 2. (a) Schematic of a DRO-Cell. Parameters for (i) C2DRO: L1 $\sim 6$ pH, L2 $\sim 20$ pH, L3 $\sim 4$ pH, J1 $\sim 115$ $\mu$A, J2 $\sim 111$ $\mu$A, J3 $\sim 80$ $\mu$A; (ii) C3DRO: L1 $\sim 6$ pH, L2 $\sim 28$ pH, L3 $\sim 4$ pH, J1 $\sim 115$ $\mu$A, J2 $\sim 85$ $\mu$A, J3 $\sim 80$ $\mu$A. J0 is removed for HC-DRO cells (b) HC-Write cell: encoder circuit to convert binary format bits $[B1 \ B0]$ into a stream of fluxons. S and M represent splitter and merger cells respectively. (c) HC-CLK cell: circuit producing three SFQ pulses by taking a single SFQ pulse (d) HC-Read cell: SFQ binary counter using T1-flipflops to convert SFQ pulses into binary format $[B1 \ B0]$ (e) JSIM simulation result of C3DRO cell

the loop is reset after each read. In the current design, each cell only stores at most a single flux, and hence each cell acts as a single bit storage.

The primary innovation of our work is to enable the superconducting loop to store more than a single flux which we name as High-Capacity DRO (HC-DRO) cells. HC-DRO cell has the same circuit schematic as a DRO cell but the buffer junction J0 is removed. The area consumed by the HC-DRO cell will at most be equal to the area consumed by a regular DRO but can store more than one fluxon. In the rest of the paper, an HC-DRO that can store two fluxons and three fluxons will be called C2DRO and C3DRO, respectively.

Required modifications to go from a regular DRO to a high-capacity DRO are: (i) increase the inductance of L2 and modify the critical currents of J1 and J2 appropriately; (ii) Remove J0 as it prevents the cell from storing more than one pulse. The modifications in terms of the device parameters are given in the description of Fig. 2. As expected, the drawbacks of these designs are some reductions in margin and frequency of operation. But we contend that reduced frequency of operation is not a major impediment since the operational frequency is not limiting the design.

There are two challenges that must be dealt with for these HC-DRO cell designs. First, the write operation must encode 2-bits of information at once into a single cell by generating multiple fluxons. Second, on a read operation, multiple fluxons must be read out and transformed into an equivalent 2-bit value for processing.

**HC-Write operation:** C2DRO cell and C3DRO cell can have memory states 0,1,2 and
0, 1, 2, 3, respectively representing the number of fluxons they can store. For a write operation, the desired number of fluxons (in the form of SFQ pulses) need to be applied at input $D$ which will get stored in the J1–L2–J2 loop. However, writing a number of fluxons more than the HC-DRO capacity will lead to pushing out of the extra fluxon resulting in an SFQ pulse at output $Q$. For interfacing with a C3DRO cell in a regular circuit that uses the binary format to represent information, the circuit in Fig. 2(b) must be used to first convert the binary format data. Let us assume that an ALU computes an 8-bit value that is going to be written to a register. The 8-bit value must be first split into groups of two-bits (starting from LSB) and each two-bit output is stored in a single HC-DRO cell. Each two-bit value is used to generate a sequence of SFQ pulses whose count is equal to the number represented by the binary number. The HC-encoder encoder circuit splits the MSB of the two bits into two and merges these two pulses along with the LSB pulse with different delay values to make sure that they do not overlap. The splitter logic is shown as $S$ and the merge logic is shown as $M$ in the figure.

**HC-Read operation:** For reading high-capacity DRO cells without adding extra JJs and inductances to the regular DRO structures, every read operation will require applying multiple clock pulses at the input $CLK$ to extract the stored fluxons. C2DRO and C3DRO cells require two and three pulses respectively to be applied at input $CLK$ for a single read operation. The output will be a series of SFQ pulses indicating the stored number of fluxons. As such, the read operation essentially translates into generating multiple read clock signals successively to extract the fluxon stored in a loop which can be done by an HC-Read circuit shown in Fig. 2(c). The total number of fluxons extracted is counted to generate the 2-bit decoded value. The 2-bit decoded value can be generated with the traditional 2-bit binary counter [6] shown in Fig. 2(d).

### 3.2. Simulation result

We implemented the HC-DRO cell along with the associated HC-Write and HC-Read blocks. Fig. 2(e) shows the JSIM simulation waveforms of input pin $D$, clock pin $CLK$, output pin $Q$, and the current in inductance $L2$ showing the evidence of stored fluxons. Clocking of the cell is shown after storing 0, 1, 2, and 3 fluxons with three pulses used for reading. Note that the current flowing in inductance $L2$ increases as the number of input fluxons increases. For every reading event, the number of fluxons stored in the cell is read and results in SFQ pulses at output $Q$. When more than 3 pulses are given as input, it is also shown that the extra fluxon is pushed out of the cell which results in an SFQ pulse at the output. The C3DRO cell functions as usual after pushing out the extra pulse which is evidenced by following a read that gives three SFQ pulses at output $Q$.

### 4. Register file design

In this section, we describe how HC-DRO cells form the basic building blocks of a higher density register file in an SFQ microprocessor. Though there were several attempts at demonstrating high-frequency SFQ microprocessors [7, 8, 9], due to the unique challenges faced by the superconducting electronics [10], the demonstrations could not compete with the CMOS counterparts. One of the challenges is to have compact register files and cache memories. Many of the current SFQ-specific register file (RF) designs [11, 12] are bit-serial designs. These designs chain several DRO cells into a shift register (typically 32 DRO cells to represent a 32-bit shift register) and the bits are written to/read from this chain. Such a design leads to a significant increase in access latency. More critically, these designs are not compatible with a register file designed with our innovative HC-DRO cell. The reason is that two bits must be stored into a single HC-DRO cell and bit-serial operation is not well suited for such compact storage.

In this section, we present an RF design without bulky cells and with simple bit-parallel read and bit-parallel write operation. This design forms the foundation for designing a register file.
4.1. Single memory cell for the Register File

Fig. 3 presents one simple memory cell design for a parallel register file design. The memory cell in the figure in our context is an HC-DRO design that can store two bits. To write the data into a desired row of the RF, an AND gate is added to the memory cell with its inputs as Address decoder output (AD) and input-bits to be written (WB). Clocking of the AND gate is done by Write CLK (WC) signal which makes sure that the data is written to the desired decoded row when a write signal is given.

We present a more area-efficient parallel register file design that reduces the JJ count. The need for Write CLK (WC) distribution can be eliminated by using the Dynamic AND (DAND) gate instead of clocked AND gate in Fig. 3(a). Design and the implementation of the DAND gate can be found in [13]. Practical implementation of a single memory cell in the RF can be seen in 3(b) with Write address decoder output (WAD) used as one of the inputs for DAND gate Read address decoder output (RAD) as Read clock.

Using the newly designed parallel multi-bit access mechanism we build a microprocessor register file using HC-DRO cells. We now describe how the register file is accessed (write and read operations).

4.2. Register file with HC-DRO Cells

Design of 8-bit register using the RF Cell from 3(b) is shown in Fig. 4. This design can have read and write ports one each and both read and write operations can be done in parallel. One of the important blocks for any memory structures is a decoder to decode the given address to access the associated memory cell(s) in the memory. For the RF design, any decoder [14, 15] structure that has minimum count of JJs can be used. Below, register file design is described in terms read and write functionality.
4.2.1. Register file Write operation

WB7 to WB0 in Fig. 4 represent the 8 bits that would be written into register file which are sent through HC-WRITE cells to convert them into a stream of pulses. These converted pulses will be placed on write channels, WC3 to WC0. We only need half of the write channels since we convert two bits of data in binary format into one stream of pulses. The write channels supply the information to be stored in all the rows of RF. A write decoder output will go to each of the rows and only one of the WADs is activated at a time based on address loaded into the decoder which selects the single row of the RF. Because of the DAND operation as shown in Fig. 3(b), the bits placed on write channels will be written into the row of memory cells that are activated and the other rows will lose the information after the hold time of DAND gate. Note that the DAND inputs will be synchronized to arrive in a stipulated time frame.

4.2.2. Register file Read operation

Similar to the WAD, one of the read address decoder outputs (RAD) is split and given to each cell in the attached row based on the address loaded into the read decoder. Since we need to the read the data in the C3DRO cell, three pulses will be generated by using a circuit similar to the one shown in Fig 2(c) and be given to the select input of the decoder which will result in three pulses at the activated RAD instead of one. The activated RAD will clock all the cells in a row and their output will be placed on the read channels (RC3 to RC0) from which the read bits are decoded using the HC-READ circuit shown in Fig. 2(d). One HC-READ circuit per channel will convert the serial pulses into binary format and it will be taken as output to give as input to other parts of the microprocessor.

It is important to note that only a single HC-WRITE and HC-READ circuit is necessary for a column of RF cell. For instance, in a 32 X 8-bit register RF only four HC-WRITE and four HC-READ circuits are needed across all 32 register files. As such the HC-WRITE and HC-READ circuits costs are amortized across the entire register file.

The comparison of RF structures using DRO cells, NDRO cells, C3DRO cells is shown in Table 1. In Table 2 we demonstrate the area benefits of using HC-DRO cells in a larger 8X8-bit
Table 1. Comparison among different memory cells in the context Register File design

| Memory Cell          | DRO  | NDRO | C3DRO |
|----------------------|------|------|-------|
| JJ count             | 4    | 9    | 3     |
| Storage capability   | 1 fluxon | 1 fluxon | Yes  |
| Can read more than once | No  | One read pulse | Yes |
| No. of pulses to read | One read pulse | Yes | Reset pulse distribution to all cells |
| Reset Required       | No   | Yes  | No    |
| Additional circuitry | Baseline |     | One HC-CLK cell and one HC-WRITE per write channel and HC-READ cell per read channel |

Table 2. Estimate of Josephson junction count of different components for 8X8-bit and 32X32-bit sizes of register file

| Register File size & Memory Cell Type | 8X8 RF | 32X32 RF |
|---------------------------------------|-------|----------|
|                                      | DRO   | NDRO     | C3DRO |
| 1-bit RF cells                        | 576   | 896      | 256   |
| (Memory cells + Dynamic ANDs)         | (256 + | (576 + | (96 + | (4096 + | (9216 + | (1536 + |
| RAD distribution                      | 168   | 168      | 72    | 2976    | 2976    | 1440   |
| WAD distribution                      | 168   | 168      | 72    | 2976    | 2976    | 1440   |
| Write channel                         | 168   | 168      | 72    | 2976    | 2976    | 1440   |
| Read channel                          | 280   | 280      | 150   | 4960    | 4960    | 2528   |
| Reset distribution                    | -     | 189      | -     | -       | 3069    | -      |
| Peripheral circuitry                  | -     | -       | 140   | -       | -       | 512    |
| Total JJ count                        | 1360  | 1869     | 762   | 23104   | 31293   | 11456  |

register file and a 32X32-bit register file. The JJ count of an 8X8-bit register file is about 56% of the traditional DRO RF. Also, note that as the size of the register file grows the overheads of HC-READ and HC-WRITE circuits are amortized. Hence, for a 32X32-bit register file, the HC-DRO design is less than 50% of the DRO register file design.

5. Branch Prediction

Any design of a microprocessor will have an instruction set specific to it and one of the important instructions is branch in different forms. Branch prediction is an important aspect of computer architecture to improve the operating latency of an instruction in a microprocessor [16]. A two-bit branch predictor is one of the most successful early branch predictors that were used to predict if a branch instruction is taken or not taken. The two-bit branch predictor can be represented as a finite-state-machine (FSM) with four states as shown in Fig. 5(a). The two-bit counter is used for representing different states as shown in the figure. In general, whenever a branch is not-taken the counter is decremented, and whenever the branch is taken the counter is incremented; these counters are saturating counters and hence no overflow or underflow. For predicting a branch instruction at a specific address, if the two-bit counter value corresponding to a hash of that address is either 0 or 1, the prediction will be not taken, and if it is 2 or 3, the prediction will be taken. In the section below, the design of a 2-bit branch predictor circuit block using C3DRO cells is explained.
5.1. Branch Predictor circuit block
The design of a two-bit branch predictor circuit system using four states of C3DRO cells for branch instructions is shown in Fig. 5(b). In this figure, we assume the branch predictor has 8 entries and each entry is accessed by a 3-bit hash of the PC register. Because of the pulse-based logic, note that a 3-to-8 decoder circuit can also be used as a 1-to-8 demultiplexer (1-to-8 DEMUX) circuit with three address bits loaded into it. In general, the branch predictor system will also have address mapping similar to the cache mapping [17], and the larger the count of branch predictor cells, the better the prediction system. The branch predictor circuit system design will be explained in terms of its functionality. Specifically, how to implement the FSM presented in 5(a) with the different number of fluxons stored in C3DRO cells as different states of FSM.

5.1.1. Reading the branch prediction value  The reading mechanism will be explained using the 8-BP cell system shown in Fig. 5(b). To read a cell in the block, the corresponding 3-bit hashed PC value will be given to the ADD port which will go to both the DEMUX cells in the circuit and then read pulses are given to READ port. The read pulses will be split and given to the SET pin of the NDRO and the READ pin of the corresponding C3DRO. The NDRO will be set and if the C3DRO has least one pulse stored in it, there will be output pulse(s) which will also be given to the READ pin of the NDRO along with those pulses appearing on OUT port. Since the NDRO is set, the C3DRO cell will be restored to the earlier value through the DEMUX cell as the OUT pin of the NDRO gives the same number of pulses as it receives at READ pin. If there is no pulse stored in the corresponding C3DRO cell, there will no OUT pulse and the NDRO will not be read to restore the C3DRO cell. Reading of the branch prediction value will always be done with two SFQ pulses because of the following reasons: (1) To read the C3DRO cell, we need three pulses and hence one pulse is not enough to read; (2) We do not require three pulses as the decision of the branch prediction will be taken when the cell is in states 2 or 3. Hence, knowing that the cell is in state 2 is enough to make the decision taken and no need to further check if the cell is in state 3. If the cell state turns out to be 0 or 1 after reading with two pulses, the decision will be not taken.
5.1.2. Incrementing the predication state  Whether the prediction is taken or not taken, if the branch should have been taken, the state of the cell needs to be incremented. Note that the maximum state is 3 and it cannot be incremented further. Once the branch decision turns out to be taken, a pulse will be sent to the TAKEN port and it will be split and sent to RESET pin of the NDRO which resets it and the SET pin of the concerned C3DRO cell. If the C3DRO’s state is 2 or smaller, its state will be incremented by one. If the C3DRO cell is in state 3, a pulse will be pushed out to through its OUT pin which will be sent to the READ pin of the NDRO that is reset and the OUT port which will be ignored as it is the incrementing operation. Since the NDRO is reset with a pulse from TAKEN, no unwanted pulse will be generated to create an infinite loop between a state 3 C3DRO cell and the set NDRO.

5.1.3. Decrementing the predication state  If the actual branch result turns out to be not taken with no regards to the prediction of a branch instruction, the state of the corresponding cell needs to be decremented. Note that the smallest state of the cell is 0 and it can not be decremented further. Once the branch decision turns out to be not taken, a pulse will be sent to the NOT_TAKEN port. It will be split and sent to the RESET pin of the NDRO which resets it and the READ pin of the corresponding C3DRO cell which also decrements it by one state if at least one fluxon is stored in it. This pulse will be split to the OUT and the READ pin of the NDRO. Note that the released OUT pulse from the C3DRO pulse will be ignored as described in section 5.1.2.

5.2. Results
5.2.1. Verilog simulation result A Verilog simulation netlist is built with the branch predictor system shown in Fig. 5(b). All the functions of the system with different combinations of inputs are verified in the simulation. Fig. 6 shows the results of verifying the functionality. The timing of the circuit is not discussed here but just the verification as the focus is on the design with
Figure 7. Relative execution time of different branch predictor compared to not having a branch predictor

high capacity DRO cells.

The functionality of the branch predictor block shown in Fig. 6(a) is described here: (1) Bits 101 are loaded as address bits (ADD) which selects the C3DRO at 101 position of the DEMUX; (2) Branch instruction relating to address 101 is taken four times consecutively so the TAKEN signal is given four times and since the capacity is only 3 pulses, the fourth pulse is pushed out showing a pulse on OUT; (3) Note that the both DEMUX cells are still selected at 101. To read the cell, two pulses are given at READ port which produced two pulses at OUT port; (4) These two pulses written back (restored) using the NDRO and DEMUX which is evidenced by reading the same cell again which produces two pulses at OUT port.

Fig. 6(b) shows the result of decrementing the state of the C3DRO cell at address 101 and reading the state of the cell. It is a continuation of Fig. 6(a) in time and the cell is in state 3 holding three fluxons. It can be seen that two pulses are given at the port NOT_TAKEN and each pulse releases a corresponding pulse at OUT port. After releasing the two pulses, the cell state is read by sending two pulses to READ port and it produces only one pulse at OUT port verifying the functionality of decrementing. Later to empty the remaining fluxon in the cell, one more pulse is given at NOT_TAKEN port, and then the cell state is read which produces no pulse at OUT port.

5.2.2. Architecture simulation result  A full core simulator is written in C++ and it is based on the traditional five-stage pipeline CPU and RISC-V ISA simulator Spike[18]. To show the advantage of a two-bit branch predictor, two different branch predictions are used whose results are given here: (1) no branch predictor, all instructions after branch instructions will be stalled until the branch instruction finishes executing; (2) 2-bit branch predictor, 128 2-bit branch predictors are used, which use a 5-bit PC hash;

Fig.7 shows the relative performance by using different branch predictors while executing different benchmarks. All benchmarks are provided by the RISC-V test GitHub repository [19]. The relative performance is computed by dividing the total execution cycles of each branch predictor by the total execution cycles of no branch predictor. The left bar shows the baseline of this comparison. The right bar shows the 2-bit branch predictor built with HC-DRO has 4.89% to 28.87% speedup compared to not having any branch predictor.
6. Conclusion
Building a dense and reliable memory structure is a challenge in single flux quantum technology. In this context, a destructive readout cell with a storage capacity of two fluxons and three fluxons (HC-DRO) is built and the nominal parameter values are presented here. The advantages of having multiple fluxon storage capability are presented by the design of register files and the design branch predictor circuit in the context of a 5-stage pipeline microprocessor. For the register file, all the peripheral circuits required for using HC-DROs as basic memory cell are also designed. The basic read and write operations are explained with an implementation that is efficient in Josephson junction count. The JJ count is reduced by 50% with the use of HC-DROs for a 32×32-bit register file. A branch prediction circuit system is built using the HC-DROs which will be in the Instruction Fetch(IF) stage of a 5-stage pipeline. It has all the operations needed for a microprocessor in modern computer architecture. Verilog simulation result of the branch prediction system verifies our design and its functionality.

References
[1] Likharev K K and Semenov V K 1991 IEEE Trans. Appl. Supercon. 1 3–28
[2] Mukhanov O A 2011 IEEE Trans. Appl. Supercon. 21 760–769
[3] Gaj K, Friedman E G and Feldman M J 1997 J. VLSI Sig. Proc. Sys. for Sig., Image Vid. Tech. 16 247–276
[4] Smith J E 1981 A study of branch prediction strategies Proceedings of the 8th annual symposium on Computer Architecture (IEEE Computer Society Press) pp 135–148
[5] Katam N K and Pedram M 2018 IEEE Trans. Appl. Supercon. 28 1–9
[6] Onomi T, Kondo T and Nakajima K 2008 High-speed single flux-quantum up/down counter for neural computation using stochastic logic J.Phys: Conf. Ser. vol 97 p 012187
[7] Ando Y, Sato R, Tanaka M, Takagi K, Takagi N and Fujimaki A 2016 IEEE Trans. Appl. Supercon. 26 1–5
[8] Bunyk P, Leung M, Spargo J and Dorojevets M 2003 IEEE Trans. Appl. Supercon. 13 433–436
[9] Sato R, Hatanaka Y, Ando Y, Tanaka M, Fujimaki A, Takagi K and Takagi N 2016 IEEE Trans. Appl. Supercon. 27 1–5
[10] Katam N K, Kawa J and Pedram M 2019 Challenges and the status of superconducting single flux quantum technology 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE) (IEEE) pp 1781–1787
[11] Tanaka M, Sato R, Hatanaka Y and Fujimaki A 2016 IEEE Trans. Appl. Supercon. 26 1–5
[12] Kirichenko A F, Sahu A, Filippov T V, Mukhanov O A, Dotsenko A V, Dorojevets M and Kasperek A K 2013 Demonstration of an 8×8-bit rsfq multi-port register file 2013 IEEE 14th International Superconductive Electronics Conference (ISEC) (IEEE) pp 1–3
[13] Rylov S V 2019 IEEE Trans. Appl. Supercon. 29 1–5
[14] Fujiwara K, Yamashiro Y, Yoshihawa N, Fujimaki A, Terai H and Yorozu S 2003 Supercon. Sci. Tech. 16 1456
[15] Vernik I, Kirichenko A, Mukhanov O and Ohki T 2016 IEEE Trans. Appl. Supercon. 27 1–5
[16] Hennessy J L and Patterson D A 2011 Computer architecture: a quantitative approach (Elsevier)
[17] Powell M D, Agarwal A, Vijaykumar T, Falsafi B and Roy K 2001 Reducing set-associative cache energy via way-prediction and selective direct-mapping Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture (IEEE Computer Society) pp 54–65
[18] Spike, a RISC-V ISA Simulator https://github.com/riscv/riscv-isa-sim
[19] Repository hosts unit tests for RISC-V processors https://github.com/riscv/riscv-tests