

Low-Leakage Asymmetrical SRAM Cell: Register File and Branch Prediction Table Characterization

Navid Azizi

*Electrical and Computer Engineering
University of Toronto*

Abstract

We propose a novel approach that leverages circuit- and architecture-level techniques to drastically reduce leakage power dissipation in register files and branch prediction structures. We observe that the resident bit values of ordinary programs exhibit a strong bias towards zero or one at the bit level in these structures. We introduce a family of high-speed dual-V_t SRAM cell designs that exploit this bit-level bias to reduce leakage power while maintaining low access latency. We propose two SRAM based designs. The first is statically biased towards the zero bit value. The other uses run-time selective inversion to increase the number of zero-holding bits. We evaluate our designs using the SPEC95 benchmarks and for a commercial 0.13 μ , 1.2V CMOS technology.

1 Introduction

As a result of technology trends, leakage (static) power dissipation has emerged as a first-class design consideration in high-performance processor design. Historically, architectural innovations for improving performance relied on exploiting ever larger numbers of transistors operating at higher frequencies. To keep the resulting switching power dissipation at bay, successive technology generations have relied on reducing the supply voltage. In order to maintain performance, however, this has required a corresponding reduction in the transistor threshold voltage. Since the MOSFET sub-threshold leakage current increases exponentially with a reduced threshold voltage, leakage power dissipation has grown to be a significant fraction of overall chip power dissipation in modern, deep-submicron ($< 0.18\mu$) processes. Moreover, it is expected to grow by a factor of five every chip generation [1]. For processors it is estimated that in 0.10 μ technology, leakage power will account for about 50% of the total chip power [2].

Unlike switching power, which is primarily a function of transistor activity, leakage power is proportional to the total number of transistors on chip. As such, a number of recent proposals for reducing leakage power have focused on-chip SRAM-based memory structures — such as multiple levels of instruction and data caches and TLBs, register file, etc. — accounting for a dominant fraction of high-end chip's transistors. Unfortunately, many of these techniques are circuit- and technology-centric and fundamentally trade off performance for lower leakage in less performance critical circuits by using higher-threshold voltage transistors.

The distribution of bit values (zeros or ones) in SRAM cells has been shown to be highly skewed towards a higher number of zeros both in the data and the instruction stream [3]. A family of novel SRAM cell designs that take advantage of the bit distribution and offer drastically reduced leakage power compared to conventional SRAM cells has been designed. Traditional SRAM cells are symmetrically composed of transistors with identical leakage and threshold characteristics. The new *asymmetric* SRAM cell designs offer low leakage with little or no

impact on latency. In the asymmetric SRAM cells, selected sets of transistors are "weakened" to reduce leakage when the cell is storing a zero (the common case).

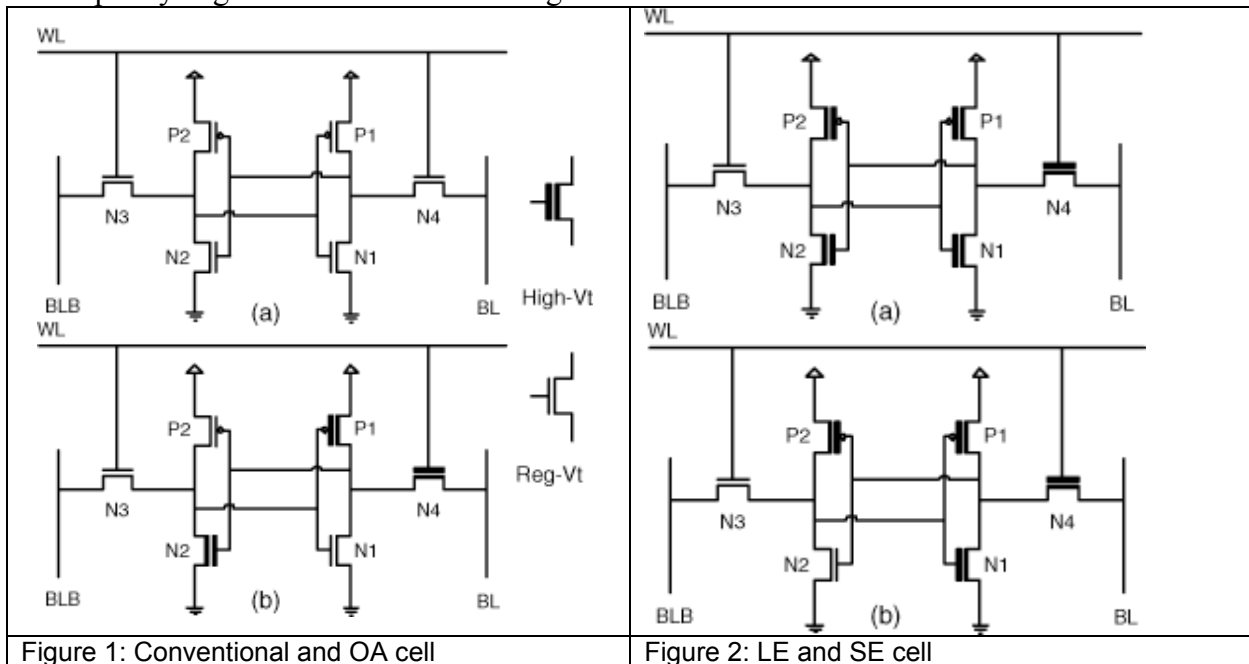
In this paper the bit distribution of register files and branch predication tables is examined to ascertain the leakage reduction possibilities in these SRAM based structures. Two SRAM structures are examined for leakage reduction possibilities: The first is statically biased towards the zero bit value. The second uses dynamic selective inversion to increase the fraction of the stored bit that hold a zero.

Compared to a conventional register file, a statically biased register file dissipates only 16% leakage power. Branch predication structures can dissipate at most 22% of conventional structures.

The rest of this paper is organized as follows: In section 2, the asymmetric cell family and sense amplifier is presented. In section 3 selective inversion is discussed. Section 4 and 5 present the simulation results for the characterization for the register files and branch predication tables respectively. Finally, we conclude the paper in section 6.

1 Asymmetric SRAM Cells

Fig. 1(a) shows a conventional SRAM cell. In the inactive state, when the cell is not being written to or read from, most of the leakage power is dissipated by the transistors that are *i*) off and that *ii*) have a voltage differential across their drain and source. In Fig. 1(a), if the cell were storing a '0', transistors P1, N2 and N4 would dissipate leakage power. If the cell was storing a '1' then transistors P2, N1 and N3 would dissipate leakage power. A simple technique for reducing leakage power would be to replace all transistors with high-Vt ones, but this unacceptably degrades the bitlines discharge times.



Since ordinary programs exhibit a strong bias in cache-resident bit values [TECH], another possibility to reduce leakage power, but at the same time keep read access times short, is to

choose a preferred stored value and to only replace those transistors that contribute to the leakage power in this state with high-Vt transistors, as seen in Fig. 1(b).

This original asymmetric cell (OA) cell was simulated at 110°C using SPICE models of a commercial 0.13 μ m, 1.2V CMOS technology and it exhibited the same leakage as the all regular-Vt (RV) cell when holding a logical '1', but it decreased leakage by 40X when holding a logical '0'. The read access time, however, of this OA cell is degraded. Due to N2's and N4's higher threshold voltage, they increase the bitline discharge time. The discharge time for the BLB and BL are 12.2% and 46.4% longer than the discharge time for the all regular-Vt case respectively. Read times can be made to match the faster read time by using a set of dummy bitlines and a novel sense amp, as is discussed below.

Starting with the asymmetric cell of Fig. 1(b), a total of 9 meaningful variations that offer different leakage and performance characteristics have been investigated. In the interest of space, the two best designs, which are shown in Fig. 2 are discussed. The leakage enhanced (LE) cell in Fig. 2(a) offers better leakage behavior than that of Fig. 1(b) because it also dissipates reduced power when holding a logical '1' since N1 and P2 have been made high Vt. Compared to the all regular-Vt cell it decreases leakage by 40X and 7X when holding a logical '0' and '1' respectively. The discharge times for this cell are 12.2% and 61.2% longer on BLB and BL respectively compared to the regular-Vt cell, but again dummy bitlines and a new sense amplifier allow the read times to match the fast side of the cell regardless of the data being stored (as will be shown below).

The speed enhanced (SE) cell in Fig. 2(b) dissipates higher leakage compared to the cell in Fig. 2(a) but it allows for read times that are virtually identical to that of the regular-Vt cell. Compared to the all regular-Vt cell the SE cell decreases leakage by more than 2X and 7X when holding a logical '0' and '1' respectively. The discharge time along BL is 61.2% longer compared to the regular-Vt cell, but dummy bitlines allow for quick sensing.

1.1 Sense Amplifier

A conventional sense amplifier is not suitable in the design due to the slow access time if the cell is storing a '0.' To obtain fast read times regardless of the data, a new sense amplifier was designed. The sense amplifier uses a set of dummy bitlines, which are always fast (as fast as the fast side of the asymmetric cell), to trigger the reading of a logical '0' thus achieving fast access times when the slow bitline is discharging. The dummy bitlines are tied to one column of dummy cells which all store a '1'. During every read operation one of the dummy cells will have its wordline asserted thus discharging DB. D and DB are connected to all sense amplifiers as shown in Fig. 3 thus timing the read for the complete byte.

2 Selective Inversion

Selective inversion is the process where stored data is inverted to increase the number of zeros. By having more zeros, the SRAM cells are in their lower leakage state and therefore leakage is reduced even further.

Selective inversion introduces additional inversion flags which impose overheads on power, area and potentially latency. Five different types of selective inversion have been investigated for time and power:

1. Majority Count per byte (MajByte)
2. Invert bytes with one or zero '1's (1-0Byte)
3. Invert 0 bytes (Inv0Byte)
4. Majority Count per word (MajWord)
5. Invert 0 words (Inv0Word)

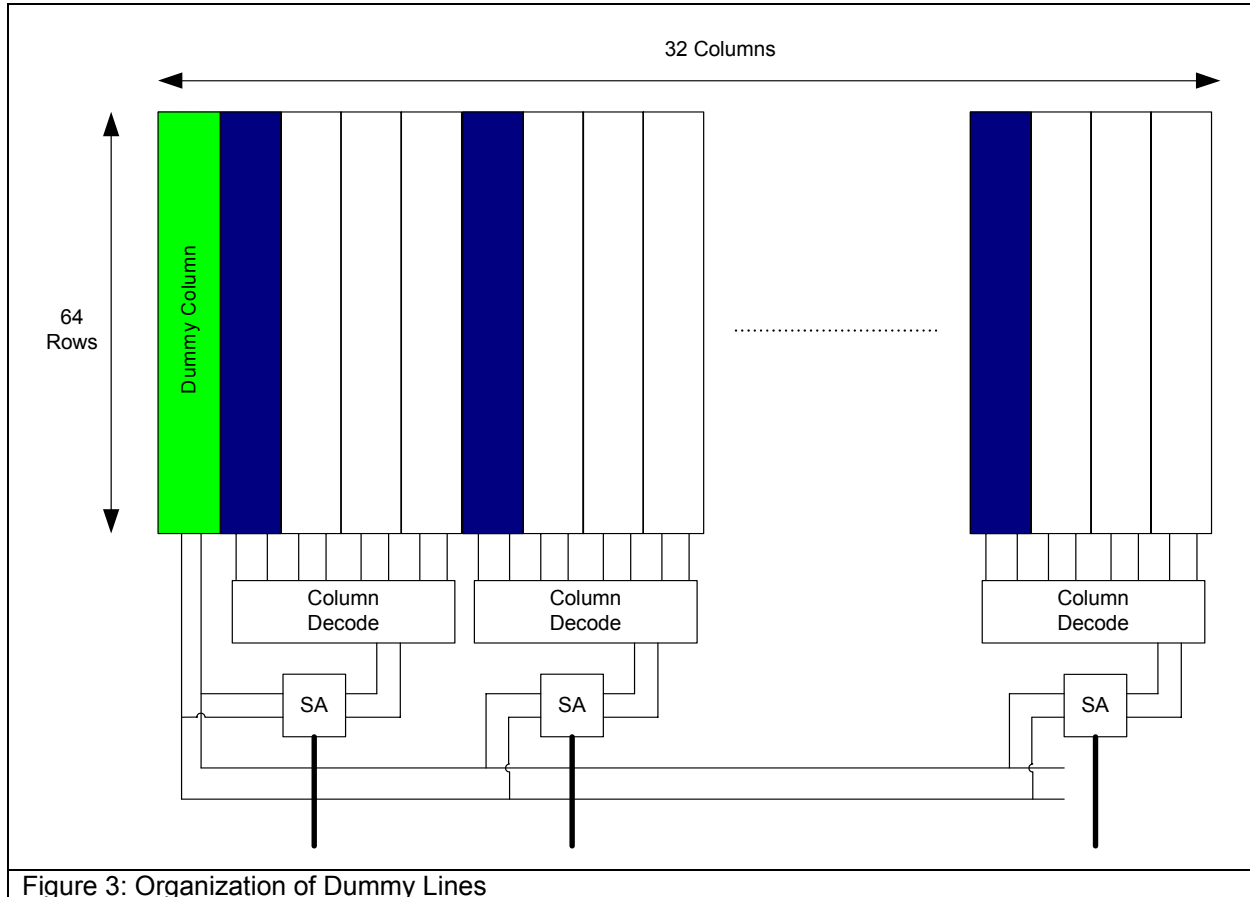


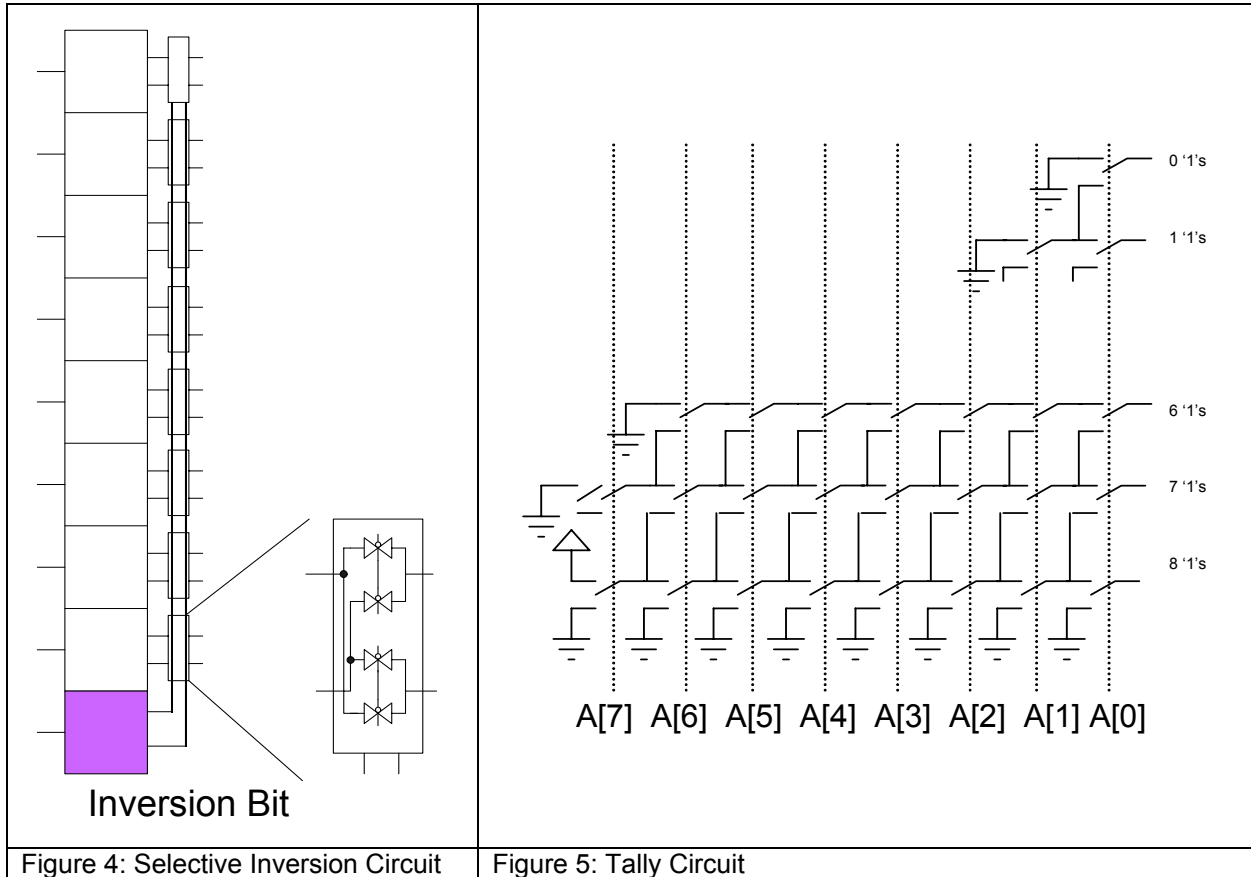
Figure 3: Organization of Dummy Lines

With the addition of inversion flags an additional latency is added to both the read access and write access times. During a read, the inversion flag has to be used to invert the data read from the array. The circuit shown in Fig. 4 performs this process: the inversion flag is used to invert the Q and Qbar outputs of the registered data which is read from the SRAM. This circuit adds only 38ps to the cycle time, or 3.8% for a 1GHz clock for 8-bit inversion (cases 1,2,3) and 180ps for a 32-bit inversion (cases 4,5).

During a write access, a calculation has to be performed to determine if the data must be inverted before it is stored. The tally circuit shown in Figure 5 can perform this function. For case 1 the complete circuit is used and an 4-input OR gate has to be used to determine if inversion must occur. For case 2 only the bottom two rows of the circuit are needed, while case involves only the final row. For case 3 and 4 a larger but similar circuit can be envisioned. The latency of this circuit for cases 1, 2 and 3 are respectively 910ps, 450ps and 200ps. Because of the large penalty involved for this 8-bit computation, the 32-bit computation was not simulated.

The extra computations before a write, and during a read also adds to the power consumption of the access. These numbers, by themselves, do not show much since their effect depends heavily

on the size of the SRAM being accessed. For larger SRAMs the cost involved in the computation may be amortized by the large leakage savings.



While it seems that the performance, power, and area penalties incurred with selective inversion are too high for the process to be valuable, especially for the high performance register files and branch prediction tables, selective inversion will be analyzed since it bounds worst case leakage.

3 Register File Characterization

Using SimpleScalar and 12 of the SPEC95 benchmarks for the ALPHA architecture, both the integer and floating-point register files were characterized for their bit distribution. Since the register file can be thought as a small cache for the data cache it would be reasonable that the bit-level bias towards 0 shown in the data cache would also appear in the register files.

3.1 Integer Register File Characterization

Fig. 6 illustrates the bit value distribution of the integer register file (GPR). The data indicates that there is much opportunity to exploit the distribution of bits in the GPR. On average, 78.3% of bit values are zero in the applications, with a maximum of 82% and minimum of 70% of bit values being zero.

The static biasing towards zero within the GPR in programs allows for dramatic leakage savings when the LE and SE cells are used. On average the LE cell reduces leakage by 32x and the SE reduces leakage by 5.8x.

When selective inversion is implemented the percentage of zeros becomes even greater. Fig. 7 shows the bit-level distribution under the different inversion policies. When data is inverted with the Maj1Byte policy, the percentage of '0's within the register file becomes 85.7%; leading to a corresponding leakage reduction of 34.4x and 6.2x with the LE and SE cell respectively. The selective inversion policy reduces leakage by an additional 7.3%.

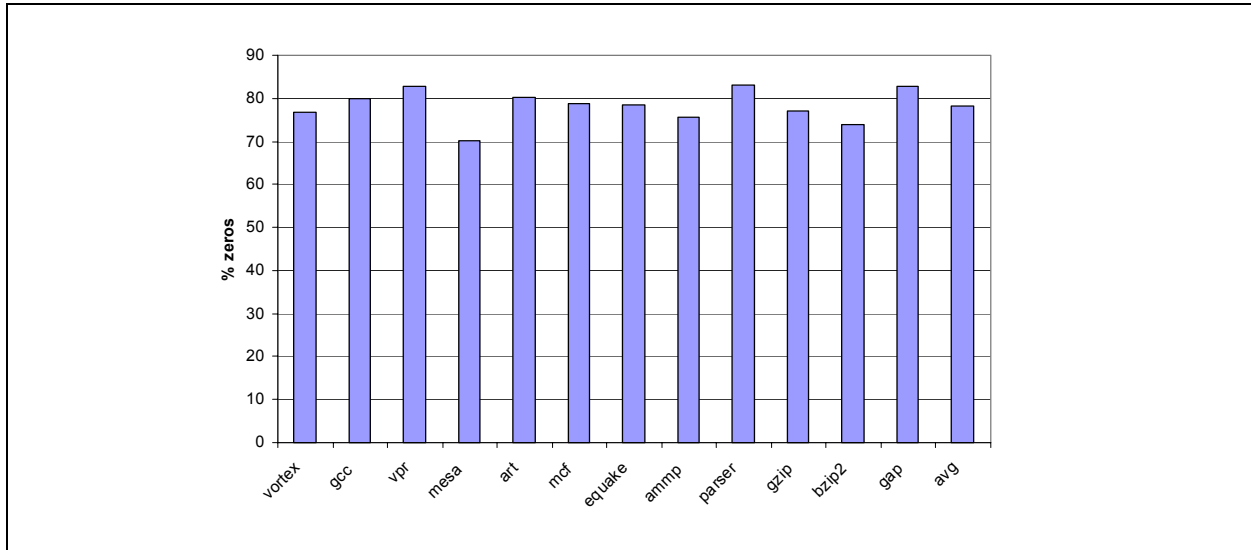


Figure 6: Integer Register File Bit-Level Distribution

Interestingly, however, when the most simple policy, the Inv0Byte, is implemented, half of the decrease obtained from the MajByte policy is achieved. The percentage of '0's within the register file is 81.4% with a corresponding leakage reduction of 33x and 6x for the LE and SE cell respectively.

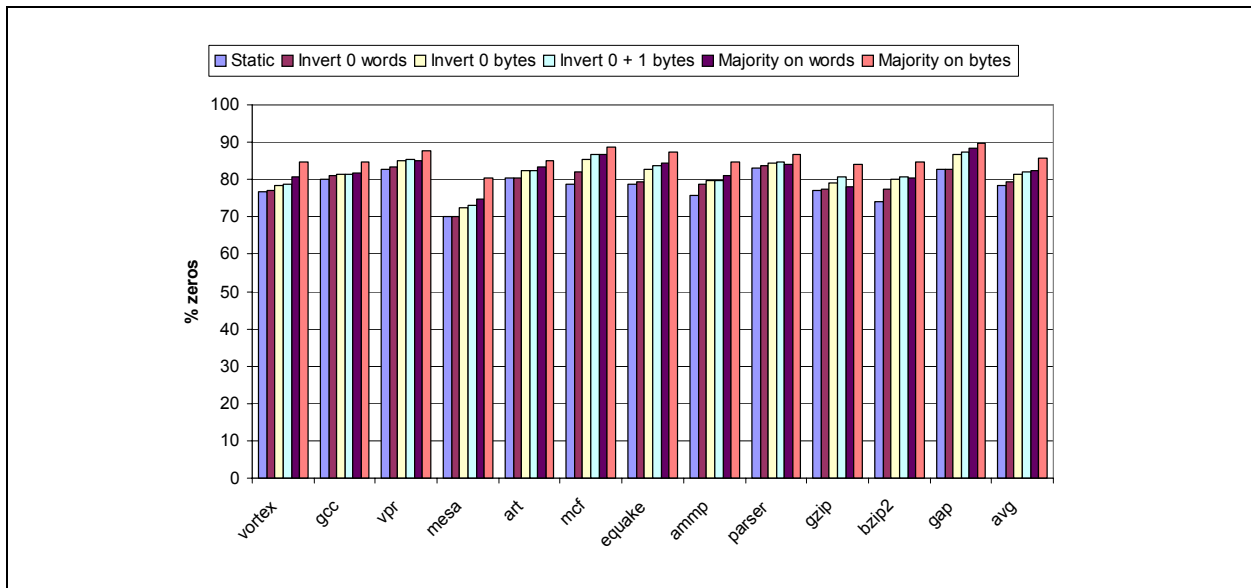


Figure 7: Integer Register File Bit-Level Distribution with Different Selective Inversion Policies

3.2 Floating Point Register File Characterization

The floating point register file (FPR) shows a different behaviour than that of the integer register file. Since not all programs use floating point values, there is the possibility that the FPR will not be accessed and contain a majority of '0's.

Fig. 8 shows the bit-level distribution among the benchmarks. Many programs show a zero-bit bias above 90%, with the minimum percentage of zeros occurring in bzip2 with 74%. On average the FPR has 93% of its contents filled with zeros. This spectacular bias allows for a leakage reduction of 34.4x and 6.2x for the LE and SE cells.

With the Maj0Byte and Inv0Byte selective inversion policies the percentage of zeros becomes 94.2% and 95.4%. Selective inversion does not change the percentage of zeros much compared to the static case since the FPR bias towards zero is so large; and correspondingly the extra leakage decreases due to selective inversion are minimal. For bzip2, however, which has the lowest percentage of zeros, selective inversion can decrease leakage by an additional 11%. Therefore if there is a large portion of floating point programs being used it may be worthwhile to implement selective inversion for the FPR. Fig. 9 shows the bit-level distribution among all benchmarks under different selective inversion policies.



Figure 8: Floating-Point Register File Bit-Level Distribution

4 Branch Prediction Table Characterization

Branch prediction tables consist of lookup tables that when given a branch address (and possibly other information), predict if the branch is taken or not. Branch prediction may be performed in many ways including simple counters, pattern recognition and more complex functions. After branch prediction has occurred other hardware may exist that speculates on the address that the branch may be targeted to.

Using SimpleScalar the contents of two different branch prediction tables were analyzed: bimodal and 2-level predictors. Each table was analyzed at two sizes 1024 and 8192 entries. Furthermore, the contents of the branch target buffer (BTB) were analyzed during the analysis of the branch predictors.

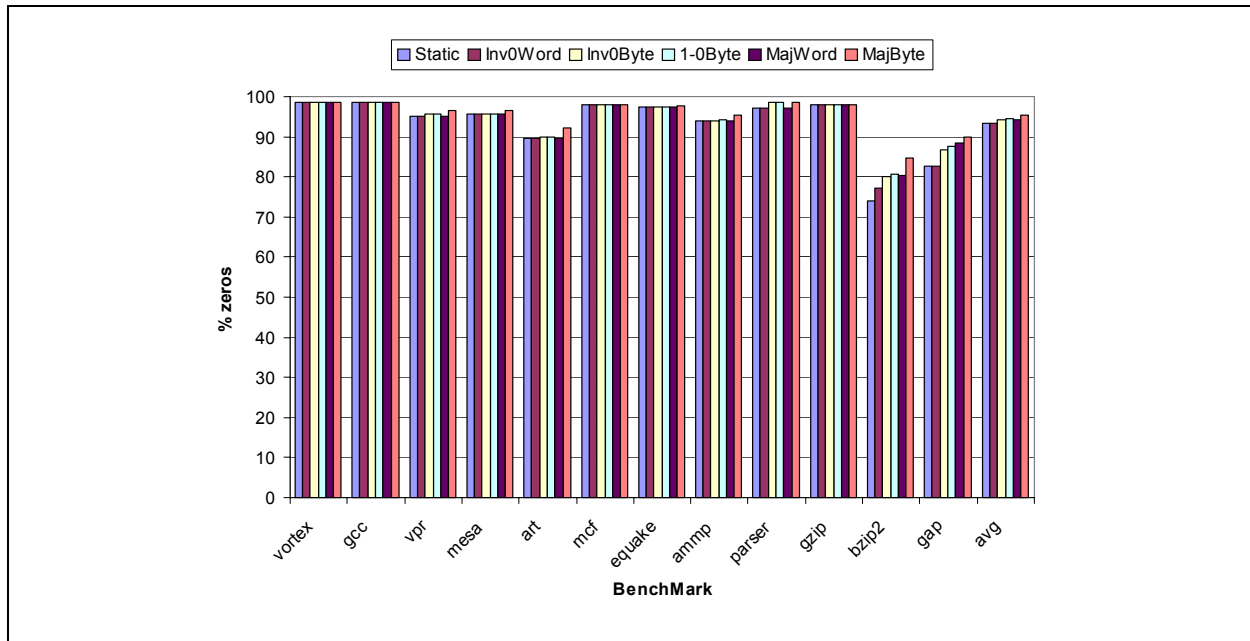


Figure 9: Floating-Point Register File Bit-Level Distribution with Different Selective Inversion Policies

4.1 Branch Predictor Table Characterization

The bimodal predictors consisted of 2 bits indicating if a branch would be taken or not. Fig. 10 shows the bit-level distribution for both configurations of the bimodal predictor. There does not seem to be a consistent bias to '0' or '1,' with averages of 52% and 50.3% zeros for table sizes of 1024 and 8192 respectively. Interestingly with the smaller size table, certain programs show a large bias (vortex at 58.8% zeros and parser at 56.3% zeros), but with the larger table size the biases all but disappear. A lack of a bias, however, does not mean that the LE and SE cells will not decrease the leakage of the tables. Since the LE and SE cells dissipate lower leakage in all states, the LE cell will reduce leakage by approximately 23x under both table sizes, and the SE cell will reduce leakage by 4.5x.

The 2 level predictors exhibited much the same behaviour as the bimodal predictors (for the second level tables). On average there was no clear bias for any bit level at both table sizes. There was, however, a larger variation per program from 46.3% to 61.9% zeros at a table size of 1024. With the larger table size the variations become more moderate. Due to the lack of a bias for any bit value, leakage reduction for the LE and SE stays near 23x and 4.5x respectively. Fig. 11 shows the bit-level distribution at both table sizes. The first level table for the 2 level predictor (which is quite small) showed a large bias towards '0's with 85% of the bits being zero.

4.2 Branch Target Address Table Characterization

The BTB target tables exhibited much the same behaviour under all scenarios and thus only the results which were obtained with the bimodal predictor with a 1024 size table will be shown. Fig. 12 shows the percentage of zeros for all policies. A large proportion of programs show a very large bias towards '0' (6 programs above 98% zeros). Other programs also showed a large bias of over 80% towards '0.'; on average 94% of bit values were zero. Selective inversion does not change the proportion of zeros largely due to the existing skew in bit-values. The leakage reduction with the use of the LE and SE cells becomes 37.3x and 6.6x respectively.

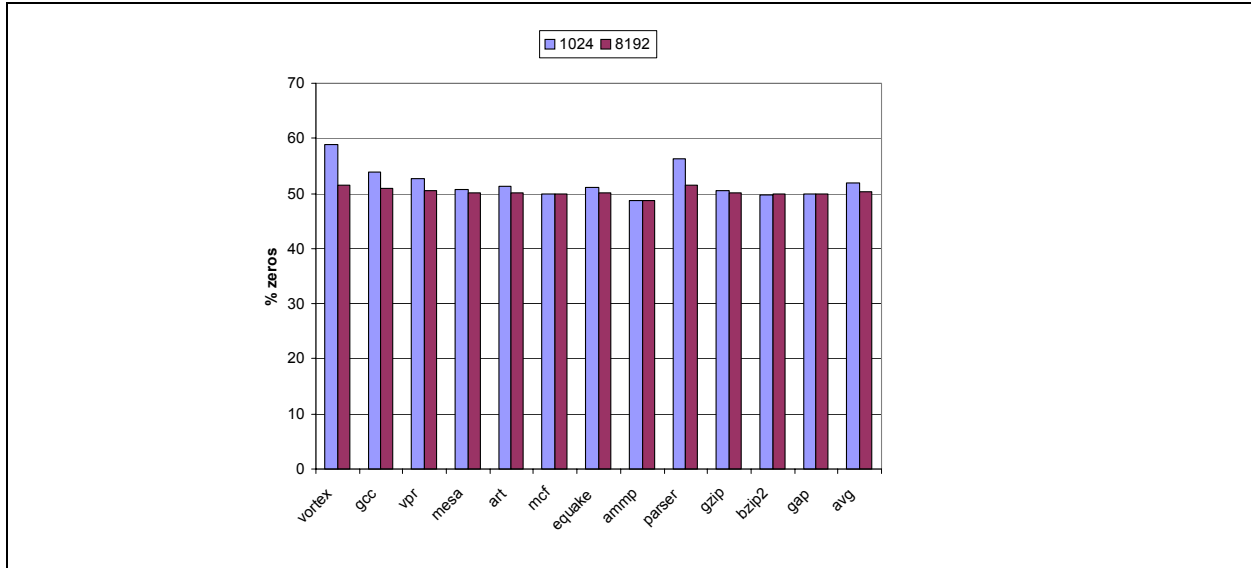


Figure 10: Bimodal Predictor Characterization

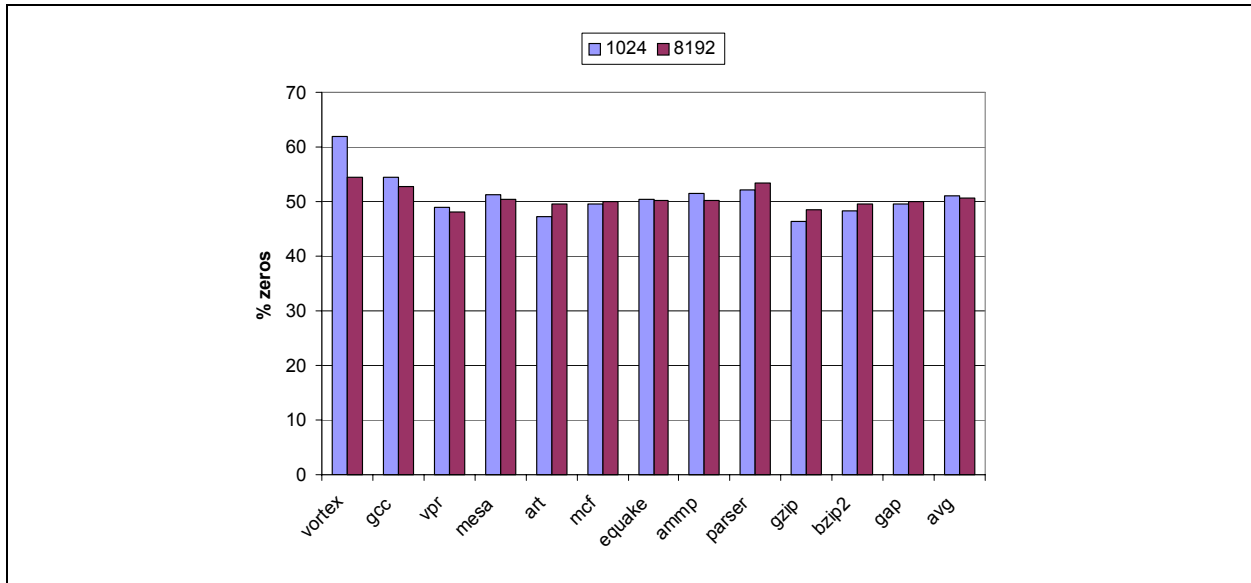


Figure 11: Level 2 of a Two-Level Predictor Characterization

The BTB does not only contain the target of the branch, but must also include the original branch address in a lookup table. The lookup table can be implemented with SRAM or with content-addressable-memory (CAM). The storage state of a CAM cell in many instances is the same as that of a SRAM cell, and thus leakage savings can be obtained if the asymmetrical cells are used. Thus the tag portions of the BTB were also characterized. Fig. 13 shows the results, which are similar to the BTB target tables. On average 93% of the bit values are zero thus allowing for a leakage reduction of 36.9x and 6.6x for the LE and SE cells respectively.

5 Conclusion

The program behaviour of programs show that the contents of the GPR and FPR are largely skewed towards zero bits. The asymmetrical cells presented allow for a circuit-level technique to level the program behaviour to reduce leakage with little or no performance penalty. Due to

the high performance requirements for these structure selective inversion latency penalties are too large and thus do not allow extra leakage reduction to be obtained. With the use of the asymmetrical cells a leakage reduction of up to 34x and 6x may be obtained.

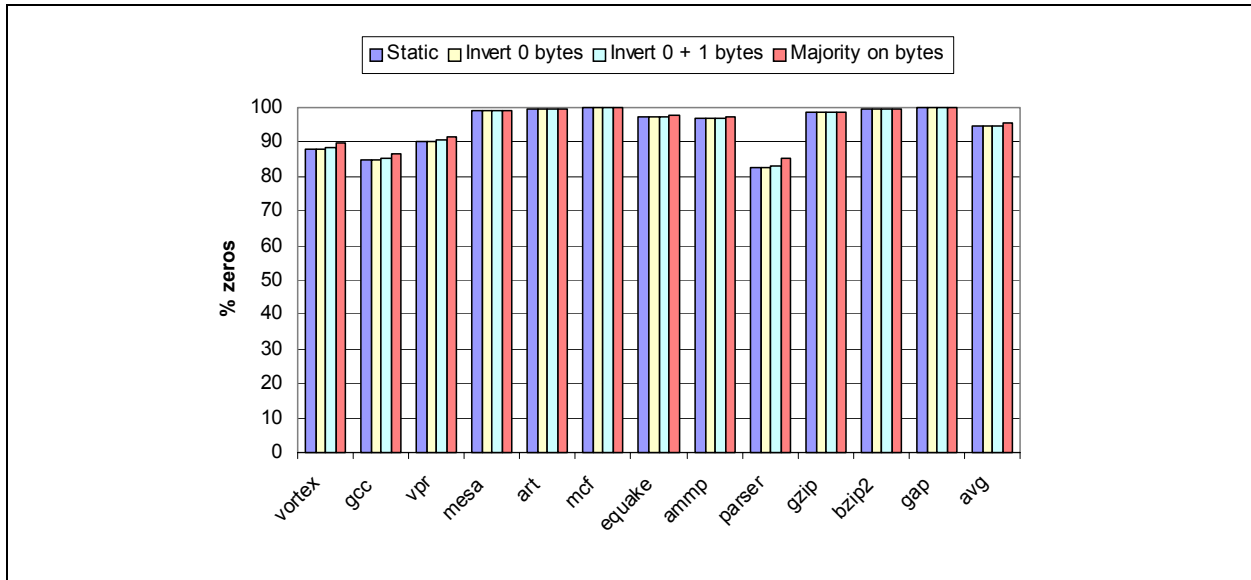


Figure 12: BTB Target Table Characterization

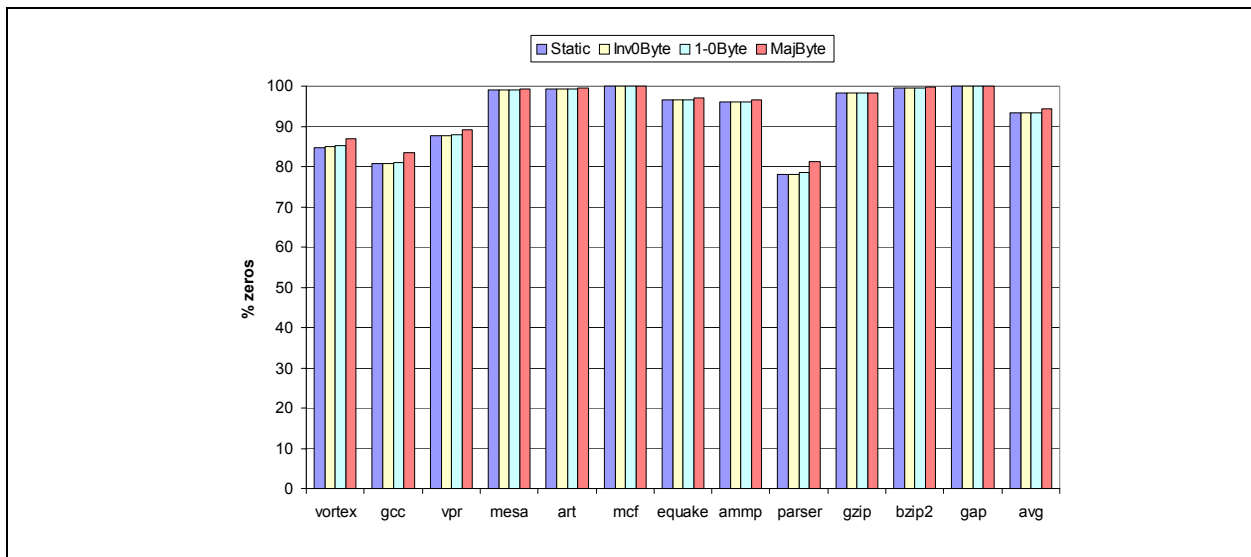


Figure 13: BTB Tag Table Characterization

Branch prediction tables show very little bias towards a specific bit value, but due to the leakage reduction in both storage states within the asymmetrical cells, a leakage reduction of up to 23x or 4.5x may be obtained.

Branch target prediction structures show a large bias towards '0', and thus allow for a leakage reduction of 37x or 6.6x with the use of asymmetrical cells.

Bibliography

- [1] S. Borkar, "Design challenges of technology scaling," *IEEE MICRO*, vol. 19, no. 4, pp. 23-29, July-August 1999.
- [2] T. Kam et. al., "EDA challenges facing future microprocessor design," in *IEEE Transactions on Computer-Aided Design*, vol. 19, no. 12, Dec. 2000.
- [3] N. Azizi, A. Moshovos, F. N. Najm, B. Falsafi, "Asymmetric-cell caches: exploiting bit value biases to reduce leakage power in deep-submicron, high-performance caches," *ECE Computer Group Technical Report TR-01-01-02*, Univ. of Toronto.