Int. Conf. On High Performance Computing, December 2003, Hyderabad, India.

FV-MSB: A Scheme for Reducing Transition Activity on Data Buses

Dinesh C Suresh[1], Jun Yang[1], Chuanjun Zhang[2], Banit Agrawal[1], Walid Najjar[1]

[1]Computer Science and Engineering Department
University of California, Riverside, CA 92521
[2]Electrical Engineering Department
University of California, Riverside, CA 92521
{dinesh,junyang,chzhang,bagrawal,najjar}@cs.ucr.edu

Abstract. Power consumption becomes an important issue for modern processors. The off-chip buses consume considerable amount of total power [9,7]. One effective way to reduce power is to reduce the overall bus switching activities since they are proportional to the power. Up till now, the most effective technique in reducing the switching activities on the data buses is Frequent Value Encoding (FVE) that exploits abundant frequent value locality on the off-chip data buses. In this paper, we propose a technique that exploits more value locality that was overlooked by the FVE. We found that a significant amount of non-frequent values, not captured by the FVE, share common high-ordered bits. Therefore, we propose to extend the current FVE scheme to take bit-wise frequent values into consideration. On average, our technique reduces 48% switching activity. The average energy saving we achieved is 44.8%, which is 8% better than the FVE

1. Introduction

Power dissipation for modern processors has become more and more important due to reliability concerns, packaging costs and mobility requirements. Among various components, the off-chip buses consume a significant amount of the total power. Stan et.al, have estimated that the power dissipated by the I/O pads of an IC ranges from 10% to 80%with a typical value of 50% for circuits optimized for low power [7]. This is due to high capacitance of the off-chip buses – up to three orders of magnitude higher than the average on-chip interconnect capacitance. It is known that power is roughly proportional to the product of capacitance and circuit switching activity. In other words, when the transition activities of the off-chip buses are increased, the power consumption is also increased proportionally. Hence, one efficient solution to reduce the power on the off-chip buses is to minimize the average transition activities through data encoding.

Both off-chip address buses and data buses are targets for encoding. However, the majority of the research in the past has focused on address buses. This is because the instruction addresses are mostly sequential, creating opportunities for simple and effective encoding. However, encoding for values transferred on data buses is not easy since data streams are less regular than address streams. Currently, encoding schemes that can be applied to data buses without prior knowledge of the applications include bus-invert encoding [7], working-zone encoding (WZE) [13], adaptive encoding [2] and frequent value encoding [8,1].

The bus-invert encoding transfers a data value either in its original form or its complement form depending on whose hamming distance with the previous bus transmission is smaller. Working-Zone-Encoding (WZE) caches references in each working zone. During subsequent references to the same working zone, the offset with respect to the previous reference is transmitted. Bus Expander [11] and Dynamic Base Register Caching (DBRC) [12] propose

compaction techniques to increase the effective bus width. DBRC uses dynamically allocated base registers to cache the high order bits of address values. Bus Expander encodes the high ordered several bits of a data value to effectively increase the bus width. The adaptive encoding scheme is capable of on-line adaptation to the value streams by learning statistics on the fly. As collecting the accurate statistics for the value streams can be very expensive, the proposed adaptive scheme operates bit-wise rather than word-wise. Thus, it looses the correlation among the bits of a single value. The frequent value-encoding (FVE) scheme is by far the most effective way of reducing the transition rate for data buses.

The FVE has exploited the high temporal locality in full-width data value streams successfully. However, we have found that the locality in the partial-width data values is also abundant. This is especially true for the high-ordered bits of the data values such as pointer values and small integer values. In this paper, we propose a new technique that exploits this locality on top of the original FVE. We use a separate small storage to capture the most frequent high-ordered bits of value streams and apply parallel FVE on both full-width and partial-width words. Our scheme achieves 48% reduction in switching activities over the unencoded data and provides an extra 8% energy saving over the original FVE.

In the next section, a brief overview of FVE mechanism is given. Then in Section 3, an in-depth observation of the values transmitted on the data bus is described. Section 4 describes our FV-MSB technique in detail. Evaluation results are illustrated in Section 5 and Section 6 respectively.

2. Previous Work

The frequent value-encoding (FVE) scheme is based on the observation that the data values transmitted on the data buses tend to reoccur frequently [9]. To take advantage of this feature, two identical codebooks are placed at two ends of buses to maintain those frequent values. The codebook is organized as a linear list where each value has a unique index. On each bus transaction, the codebook is looked up to see if the value to be transmitted is frequent or not. If it is frequent, a code is generated indicating the index in the codebook where the value is stored. Thus the receiving end can use this code to locate the frequent value in its own codebook. If a value is non-frequent, it is sent directly onto the bus without encoding. A control signal is needed to distinguish between a non-frequent value and a frequent value only when it can cause confusion at the receiving end.

A code for a frequent value has the flavor of “one-hot-code” meaning that there is only a single-bit “1” present in the code and all the rest bits are “0”s. The position of the bit “1” exactly determines the index of the frequent value in the codebook. For example, a code “1000” means the bus is transmitting the first frequent value in the four-entry codebooks on both ends of the bus. The advantage of using “one-hot-code” for frequent values is that it can guarantee low transition activity if the frequent values are abundant. The disadvantage is that the codebooks cannot contain more number of frequent values than the number of bus wires.

Changes are made to the frequent value set in the codebook by applying LRU replacement policy on every new value being looked up. Thus the codebooks essentially keep the most recently seen values over a window of ‘n’ values, where ‘n’ is the codebook capacity. The codebooks can be easily implemented in hardware using a Content Addressable Memory (CAM). For a k-bit wide data bus, we use a k-bit wide, k-entry CAM to hold the frequent values. For the rest of this paper, we will refer to the CAM that holds the frequent values as FV CAM.

As used in other techniques [2], a pair of correlator and decorrelator is added to the two ends of buses. They are inverse functions of each other and their purpose is to reduce the correlations between successive values. The schematic block diagram of the frequent value encoding is shown in Fig. 1.

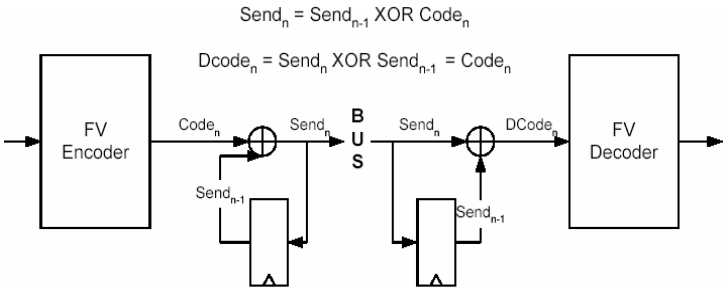

Fig. 1. Frequent Value Encoding Scheme

3. Motivation

Since the FVE scheme has a dynamically changing set of frequent values, it exploits the short-term temporal locality on the data bus effectively. Any improvement to the FVE scheme should focus on capturing more values in the FV CAM .We therefore design such a scheme. The values to be placed on the off-chip data bus depend on the nature of the data set, the program and the micro architectural features of the system. Even for a given application, different phases of the program might exhibit different access patterns. For example, the data access pattern for a normal program and for a linked-list traversal program might differ drastically. Table 1 shows an example of data trace segments for a linked-list traversal and a normal program.

A list might often contain a set of contiguously allocated pointer values and hence the values on the data bus might often differ by just a few bits. If we have multi-threaded execution pattern or if we have instruction values in between data streams, these consecutive values might be separated by other values and hence, passing consecutive pointer values might resulting a lot of transitions. Let us look at how such consecutive pointer values are handled by the FVE scheme. Each time a new pointer value is encountered, it is treated as a non-frequent value and hence, the data is sent unencoded. In the example shown in Table 1, the highlighted values in the linked list trace correspond to pointer values and are always sent unencoded in the FVE scheme. One might observe that, for most of the consecutive pointer values, most of the high ordered bits remain unchanged. Hence, if we can cache the high ordered bits in a separate CAM, many values that were earlier treated as non-frequent by the FVE scheme can be encoded as frequent values.

Table 1. Data trace segments for linked list and a for loop

Linked List trace	Normal program
10005098	00000048
0000001a	00000006
00000012	00000042
100050d8	8048c0f6
00000000	00000047
00000036	00000002
100050f8	00000006
0000001a	00000046
00000012	00000006

Besides pointer values, many non-frequent values might differ from frequent values by small magnitudes. If such non-frequent values were separated from frequent values by other un-related values, we would have a lot of switching activity on the bus. The values 42 and 47 are separated by an unrelated value in the normal program trace shown in Figure2. We find that besides caching frequent values, caching a few upper order bits would reduce the transition activity in the bus significantly.

Based on the above observations, we want to devise a scheme in which both entire data value and its most significant bits (MSB) are kept in CAMs. The FV CAM serves its original purpose while the MSB CAM is to capture the locality in the high-ordered bits as described. We will study how many high-ordered bits are needed and how they interact with the FV CAM. Since we are using more hardware than before, our design goal is to further reduce the transition activity, and thus, the overall power consumption, without adding expensive hardware resources such as extra control signals. Moreover, we design the layout of our codec so that it is fast and consumes power economically. We provide power and delay analysis of the codec and consider the power overhead of the additional hardware when evaluating our technique.

4. FV-MSB Encoding

Fig. 2 gives a high-level picture of our FV-MSB design. The FV-CAM functions as the original FVE scheme and MSB CAM is to capture more localities of the high-ordered *m* bits for every value being transferred. If a value hits in both CAMs, we give higher priority to the FV CAM since it helps reduce more switching activity (middle AND gate). In other words, in the event of a FV CAM miss, the MSB CAM is useful (AND gate on top). If the value missed in both CAMs, the original binary form is sent over the bus as usual (the AND gate at the bottom). The gates in Fig. 2 represent the logic design of the FV-MSB scheme. In reality, each wire should be wired in a similar way. We choose to search both CAMs in parallel in order to minimize the delay on the critical path.

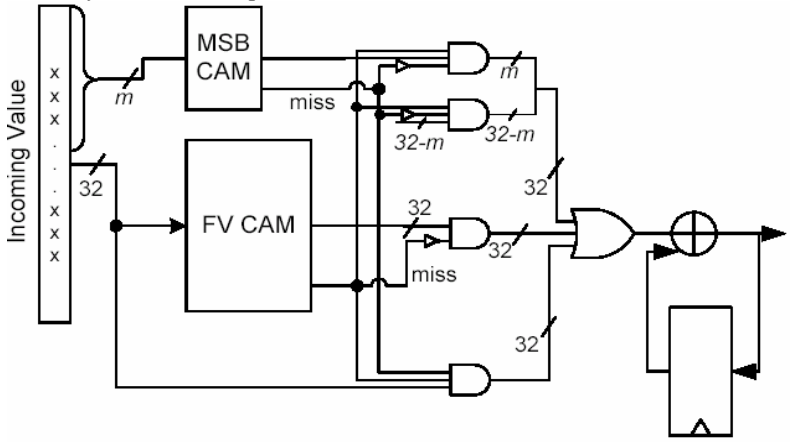

Fig. 2. Frequent Value-MSB encoding scheme

Two difficulties arose while we were designing the FV-MSB scheme. The first is that the MSB CAM might require another control signal just as the original FVE since a non-frequent value may look like a value whose high-ordered *m*-bits are encoded by the MSB CAM. The second issue is the value of *m*. If *m* is small, we would have a higher hit rate in MSB CAM,

but each hit does not bring too much reduction in switching. If *m* is large, the MSB CAM would have lower hit rate, but each hit results in better switching reduction. Next we illustrate these two issues separately.

4.1. Removing the Additional Control Signal

The original FVE requires a control signal to inform the decoder that the transmission on the data bus is a non-frequent value but having a "one-hot code" form. Since the MSB CAM is essentially a smaller scale FV-CAM, a similar control signal would be necessary to guarantee the correctness of the encoding process. This means that we would use two extra off-chip pins and wires to set up special encoding condition. Since off-chip pins and wires are expensive resources and also introduce toggles on the bus, we strive to design the FV-MSB scheme such that the extra control signal is not needed.

Let us first look at those cases that require two control signals. At the receiving side, the decoder should be able to tell the following situations:

1. Is the incoming value a frequent or a non-frequent value? A frequent value may not appear as a pure "one-hot code" since high-ordered *m* bits might be encoded while the remaining 32-*m* bits contain "1"s
2. If the incoming value is a frequent value, was it encoded using the MSB or the FV CAM? An incoming value with the "hot" bit being among the high-ordered *m* bits confuses the decoder in selecting the right CAM to decode.

Let's assume the four combinations of the two control signals represent different encoding meanings as the following. 1) "00" means the value missed both CAMS and the bus transmission is an unencoded value; 2) "01" means the value hit in the MSB CAM; 3) "10" means the value hit in the FV CAM; 4) "11" is left unused.

To remove one signal, we need to combine two cases in the above categories so that we are left with only two cases for which a single signal would suffice. Combining "00" and "01" is impossible due to the reason listed in 1 above. Combining "00" and "10" is also impossible since a non-frequent value with "one-hot code" form cannot be distinguished. Therefore, we can only combine "01" and "10" but at the cost of encoding less number of frequent values that are being hit in the MSB CAM. Here, we choose to give up encoding those values that hit in the MSB but miss in the FV-CAM and their lower 32-*m* bits are "0"s. Those values will be transmitted as unencoded non-frequent values. After this modification, the receiving side will see only the following forms of bus values:

a) A value of pure one-hot code form.
b) A value of "one-hot code" in the high-ordered *m* bits and non-zero in the low-ordered 32-*m* bits.
c) A value of other forms including non-frequent values having "one-hot code" form and frequent values whose "hot" bit is in the high-ordered *m* bits.

Cases a) and b) are encoded frequent values and can be indicated by a signal setting at "1", and case c) is for non-frequent values and can be indicated by the signal setting at "0". Thus, we successfully removed the additional control signal.

4.2. Selecting the Size of *m*

The number of bits *(m)* per value we choose to store in the MSB CAM is also the number of entries of the CAM. Intuitively, small values of *m* bring in large number of hits in the MSB CAM. However, large number of hits need not necessarily imply a large reduction in switching activities. For example, if *m=3*, then there are about 3/8 ($m/2^m$) of the total values hit in the MSB CAM. However, the average number of reduction in switching activities is only

0.5 (3/2 –1). If *m* is larger, the hit rate will decrease, but the effect of the hit on switching reduction will increase. To find the best value of *m*, we conducted experiments where we varied *m* from *8* to *30* with step value of *2* and measured 1) hit ratios in MSB CAM, and 2) overall switching reductions of the FV-MSB. The experiments were carried out with configurations specified in Section 5

Fig. 3 plots the averaged ratio of the MSB CAM hits normalized to the FV CAM hits over the 7 benchmarks we ran. The curve is bell shaped which shows an interesting feature. Intuitively, the hit ratio should be higher when *m* is smaller. Meanwhile the number of entries in the MSB CAM is fewer which means that the MSB CAM keeps fewer number of high-ordered *m* bits and hence, captures less locality. These two contradictory conditions cause the curve to rise from 8-entry to 16-entry MSB CAMs. The curve reaches the peak at 16-entry CAM and falls from then on.

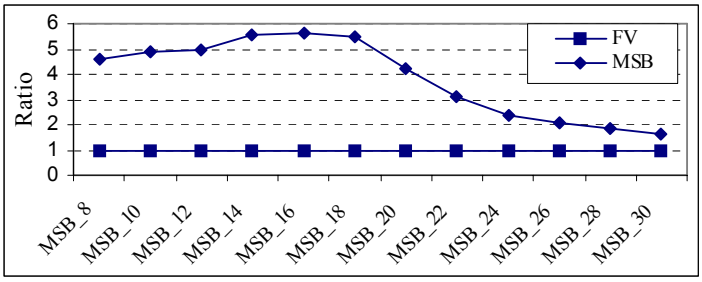

Fig. 3. Number of hits in the MSB table vs. number of hits in the FV table

Fig. 4 shows the percentage of switching reduction in the same experiment. As we expected, having higher hit rate in the MSB CAM does not necessarily result in higher switching reductions. However, a low hit rate will definitely hurt the reductions in the switching activities. Hence, an optimal point that gives the peak reduction is desired. From the graph, we can see that the MSB CAM with 20 entries outperforms all other MSB CAMs. Therefore, we use a 20-entry MSB CAM to hold the high 20-bits of data values in our FV-MSB scheme.

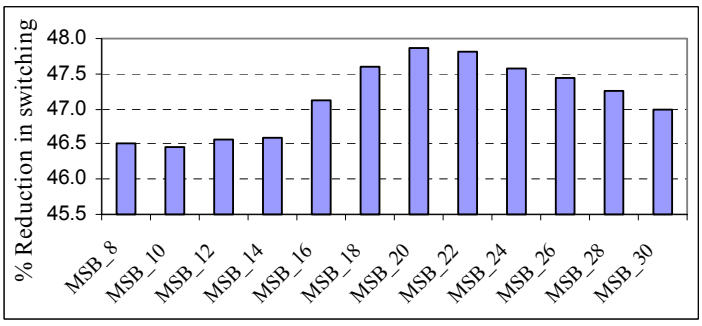

Fig. 4. Percentage of reductions in switching activity for SPEC2000INT

5. Evaluation

In this section, we present the experimental evaluations of FV-MSB and compare the data bus switching reductions with four other schemes. We used execution-driven Simplescalar-3.0 tool set [4] with a configuration that conforms to modern high performance processors. We modified the source code of *sim-outorder.c*, an out-of-order execution simulator to monitor the activities on the off-chip data bus which is between two level caches and the memory. The L1 cache size is 64KB for both data and instructions; the L2 cache is 512KB. We modeled the FV-MSB scheme together with four other analogous schemes for comparison. The descriptions of two out of four schemes are as follows (the first one is bus-inversion and the fourth one is FVE).

FVE-64 (a 64-entry FV CAM encoding): The FV-MSB scheme uses an MSB CAM in addition to the FV CAM. The total number of bits stored in the CAMs altogether is higher than the normal FVE. One way of comparing the two schemes is to increase the capacity of the CAM in FVE. However, the authentic FVE does not allow the number of entries more than the number of bus wires (in this paper, we assumed 32). A straightforward way of realizing it is to use a CAM whose number of entry is a multiple of 32. Meanwhile, we use the same number of control signals to select among different 32 entries. For example, if we use a 64-entry FV CAM in FVE, we need to use 1 signal to choose between the upper 32 entries and the lower 32 entries of the CAM. In addition, we need a second signal to indicate a non-frequent value having "one-hot code" form. Notice that this method is neither scalable in terms of CAM size nor realistic in terms of the number of control signals it requires. For the purpose of comparison, we pessimistically pick a 64-entry CAM to compare with our FV-MSB (32-bit, 32-entry FV CAM and a 20-bit 20 entry MSB CAM).

FV-Inversion: This scheme combines the traditional bus-invert and FVE together: whenever a value miss occurs in the FV CAM, the bus invert algorithm is applied. In reality, the two encodings can be implemented in parallel to reduce delays just as in our FV-MSB scheme. However, unlike our FV-MSB scheme, this combination also requires two additional control signals, one for bus-invert and the other for FVE.

Fig. 5 shows the percentage reduction in switching activities for SPECINT applications. FVE-64 and FV-Inversion have an additional control signal overhead compared to the FVE-32 and the FV-MSB scheme. Besides using more hardware than the FVE-32 scheme, they can only provide modest reduction in switching activities. On average, the FV-MSB scheme provides 10% more switching reductions compared with the FVE-32 scheme and provides 48% switching reductions compared with unencoded data. The increase in switching reductions is chiefly due to the fact that FV-MSB scheme can track pointer values effectively. Fig. 7 in Section 6 gives the percentage energy reduction for different schemes while running the SPECINT benchmarks.

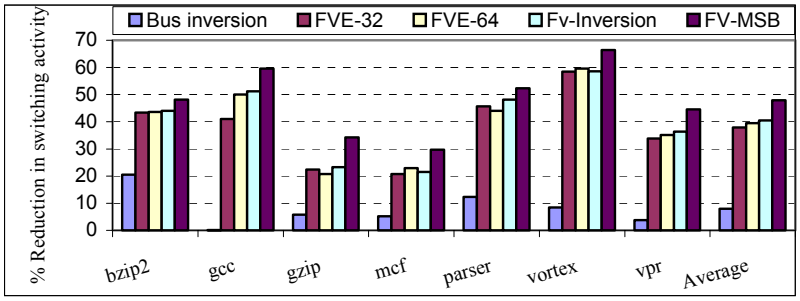

Fig. 5. Percentage reductions in switching activity for SPEC2000INT

6. Energy, Delay and Area Measurement

To determine the power consumption and delay of the coder itself, we created an actual layout of the CAM and other components of the FV-MSB scheme. Fig. 6 (left) is the CAM cell circuit, which is composed of a conventional six-transistor SRAM cell and dynamic XOR comparators. Since CAM search time is critical in our FV-MSB design, we used two separate bit lines: Cbit and Bit, to decrease the capacitance on the Cbit search line. Fig. 6 (right) shows the layout of the CAM cell. We used Cadence [3] layout tools and extracted the circuit from the layout.

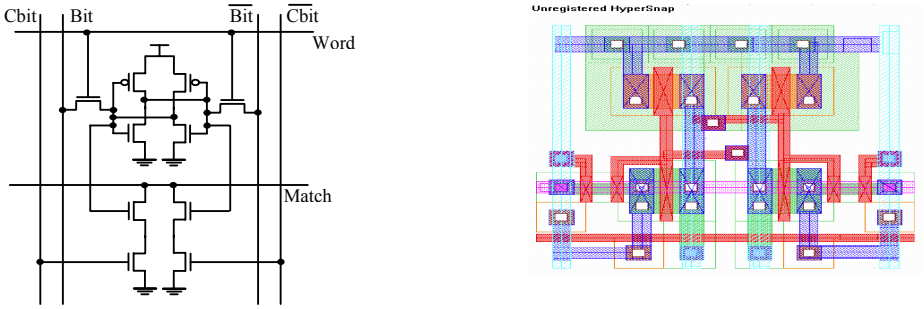

Fig. 6. CAM cell circuit (left) and layout (right)

The technology we used was TSMC 0.18µ, the most advanced modern CMOS technology available to universities through the MOSIS program [6]. The dimensions of our CAM cell are 5.3µm x 5.6 µm (29.7(µm) ^{2} area). We used Cadence's Spectra to simulate the net list of the extracted circuits. We measured the access time and energy consumption of the encoder from the outputs of the simulation. Table 2 lists the results for original FVE coder in which the comparator and timestamps are necessary constructs. We derived our energy calculations based on these results.

$$E_{Total} = E_{CAMs} + E_{timestamp} + E_{comparator} + E_{Xor} + E_{32\ and,or}$$

Table 2. Energy measurement for different FVE components

Component	Energy consumption	time
Comparator	1.27 pJ/access	0.2ns
XOR gate	0.095 pJ/Transition pair	0.1ns
64 entry CAM	28 pJ	0.2ns
timestamps	0.07pJ	0.5ns

Using the above equation, we calculated the value of E_{Total} for FVE-32, FVE-64 and FV-MSB to be 17.38pJ, 34.65pJ, and 36.65pJ respectively. The off-chip bus energy per bus wire is given by the formula:

$$E \propto C \times V^2 \times A,$$

where *C* is the off-chip bus capacitance, *V* is the supply voltage and *A* is the number of bus transitions. Assuming bus capacitance and bus voltage values of 60pF [5] and 3.3 volts respectively, the energy per bus transition is given by:

$$E_{bus\text{-}transition} = 60 \times 10^{-12} \times 3.32 = 600pJ$$

If we have an average of 10 transitions on the 32 bus wires during each cycle, one might notice that the per-cycle-energy dissipation in the FV-MSB codec is less than the energy spent in a single off-chip bus cycle by a factor of 200. Even with a supply voltage of 1.8V, the energy consumed by the FV-MSB codec is nearly 60 times lesser than the per-cycle off-chip bus energy. Hence, we can conclude that the energy consumed by the circuit components of our encoding scheme is quite insignificant when compared with the energy saved through the reductions in switching activities. The total energy dissipated in the bus is the sum of the total energy consumed during the bus transition and the energy consumed by the encoder and decoder (two times the encoder energy). We plotted the results in Fig. 7. Here, we compare three techniques: FV-32, FV-64 and FV-MSB. We can clearly see that FV-MSB saves more energy than the other two by a factor of 8%.

Table 2 also shows the latency for different components of the FV-MSB scheme. The critical path of the FV-MSB scheme is composed of CAM access, updating timestamps, selection AND/OR gates and the XOR gates. Adding delays for each one of them gives a total of 1.1ns delay (0.2 + 0.5 + 0.1 + 0.3). Notice that the comparator in the original FVE operates in parallel with the CAM access, therefore not taking the extra critical path delay.

Finally, the FV CAM and the MSB CAM are the major contributors to the area overhead. Since a CAM cell is of area 29.7 $(\mu m)^2$, multiplying it by the total number of bits in the FV-MSB yields 42 x 10^{-3} mm^2 (29.7 × (32×32 + 20×20)). Thus, our conclusion is that the FV-MSB scheme is an economic design in terms of energy, delay and area overhead

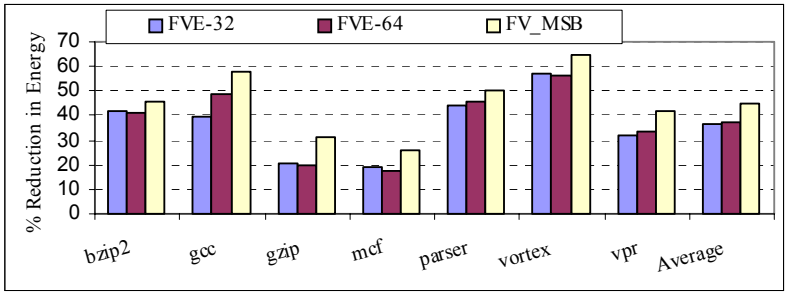

Fig. 7. Percentage of energy reductions for SPEC2000INT

7. Acknowledgement

This work was supported in part by NSF Award ITR 0083080.

8. Reference:

[1] K. Basu, Q. Choudhary, J. Pisharath and M. Kandemir, "Power Protocol: Reducing Power dissipation on off-chip Data Buses", The 35th IEEE/ACM International symposium on Microarchitecture, pages 345-355, 2002.

[2] L. Benini, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Synthesis of Low-Overhead Interfaces for Power-Efficient Communication Over Wide Buses", ACM/IEEE Design Automation Conference, pages 128-133, 1999.

[3] Cadence Corporation: http://www.cadence.com

[4] D. Burger, and T. Austin, "The SimpleScalar Tool Set, Version 2.0", Technical Report 1342, Universoty of Wisconsin-Madison, Computer science Department, 1997

[5] H. Messmer, "The Indispensable PC Hardware Book, Fourth edition, Addison-Wesley, 2002

[6] The Mosis Service: http://www.mosis.com/.

[7] M.R. Stan and W.P. Burleson, "Bus-invert coding for low-power I/O", IEEE Transactions on very Large Scale Integration (VLSI) systems, pages 49-58, Vol.3, 1995

[8] J. Yang and R. Gupta, "FV Encoding for Low-Power Data I/O", ACM/IEEE International Symposium on Low Power Electronic Design, pages 84-87, 2001

[9] Y. Zhang, J. Yang and R. Gupta, "Frequent Value Locality and Value-centric Data Cache Design", The Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX), pages 150-159, 2000.

[10] A. Efthymiou and J.D. Garside, "An Adaptive Serial-Parallel CAM Architecture for Low-Power Cache Blocks,", Proceedings of the International Symposium On Low Power Electronics and Design, 2002.

[11] D. Citron and L. Rudolph, "Creating a wider bus using caching techniques ", Proceedings of the first International symposium on High Performance Computer Architecture, pp 90-99, Jan 1995.

[12] Matthew Farrens and Arvin Park. "Dynamic base register caching: A technique for reducing address bus width" In Proceedings of 18th Annual International Symposium on Computer Architecture, pages 128--137, May 1991. Toronto, Canada.

[13] E. Musoll, T. Lang, and J. Cortadella. "Working-zone encoding for reducing the energy in microprocessor address buses". IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 6, 1998