Power Protocol: Reducing Power Dissipation on Off-Chip Data Buses*

K. Basu, A. Choudhary, J. Pisharath
ECE Department
Northwestern University
Evanston, IL 60208, USA
{kohinoor,choudhar,jay}@ece.nwu.edu

M. Kandemir
CSE Department
The Pennsylvania State University
University Park, PA 16802, USA
kandemir@cse.psu.edu

Abstract

Power consumption is becoming increasingly important for both embedded and high-performance systems. Off-chip data buses can be a major power consumer. In this paper, we present a strategy called "power protocol" that tries to reduce the dynamic power dissipation on off-chip data buses. To accomplish this, our strategy reduces the number of bus lines that need to be activated for data transfer by employing a small cache (called "value cache") at each side of the off-chip data bus. These value caches keep track of the data values that have recently been transmitted over the bus. The entries in these caches are constructed in such a way that the contents of both the value caches are the same all the time. When a data value needs to be transmitted over the bus, we first check whether it is in the value cache of the sender. If it is, we transmit only the index of the data (i.e., its value cache address) instead of the actual data value and, the other side (receiver) can determine the data value by using this index and its value cache. Our experimental results using a set of fifteen benchmark codes from embedded systems domain show that power protocol is very effective in practice, and reduces the bit switching activity on the data bus by as much as 70.7% (with a value cache of 128 entries). We also present results from an implementation that combines our strategy with 1-to-2 encoding, a popular bus encoding strategy for low power. Our results indicate that this combined optimization strategy reduces bit switching activity by 67.8% on the average (across all benchmarks). These reductions in bit switching activity lead to more than 7% reduction on overall system energy on the average for a value cache of 256 entries. We also study the sensitivity of our savings to the value cache capacity and data cache capacity.

*This work was supported in part by NSF Award #0093082 and by DARPA's Power-Aware Computing and Communications Program under contract No. FF33615-00-C-1631.

1. Introduction

Architectural-level power optimization can target different system components such as CPU, caches, main memory, and buses [5, 6, 22, 4]. Power spent in off-chip buses can be a significant portion of the overall power budget. As an example, the core power consumption of Intel Celeron at 266 MHz is 16W, while its off-chip bus (for a standard configuration) operating at 133 MHz consumes 3.3W [12, 13]. The contribution of off-chip bus power consumption to the overall power budget increases even more for embedded systems with low-power processor cores and memories, making off-chip buses a potential candidate for power optimization. Figure 1 shows the power consumption due to off-chip data bus for several embedded benchmark codes as a percentage of the overall power consumption (which includes processor datapath, caches, buses, TLB, register file, instruction issue logic, clock, and off-chip memory) for an embedded processor. From this figure, we see that the off-chip data bus consumes between 9.8% and 23.2% of the total power consumed by the system depending on the benchmark being run. So, reducing the power consumption of the off-chip data bus would reduce the overall power consumption of the system to a considerable extent.

Power dissipation on off-chip data buses can be reduced by at least two techniques: by reducing the number of bus lines activated during data transfer and by reducing the number of bit transitions on the active bus lines. Consequently, maximum benefits can be obtained by applying both the techniques. In this paper, we present a data bus transmit protocol, called *power protocol*, to reduce the power consumption on off-chip data buses. In this approach, we employ a small cache (called value cache, or VC for short) at each side of the off-chip data bus. These value caches keep track of the data values that have recently been transmitted over the bus. The entries in these caches are constructed in such a way that the contents of both the value caches are the same all the time. When a data value needs to be transmitted over the bus, we first check whether it is in the value cache of the sender (whether it is memory or cache). If it is, we transmit only the index of the data (i.e., its value cache address, or index) instead of the actual data value and, the other side (receiver) can determine the data value by using this index and its value cache. Since

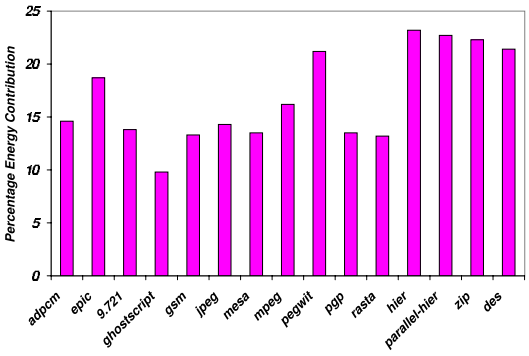

Figure 1. Contribution of the off-chip data bus power to the overall power consumption (0.18 micron technology; 8KB, direct-mapped data cache with 32 byte cache lines). The overall power consumption includes the power consumed in on-chip components (including data and instruction caches), off-chip buses as well as off-chip memory. We see that the off-chip data bus consumes between 9.8% and 23.2% of the total power consumed by the system depending on the benchmark being run.

the value caches employed by our power protocol are very small, the width of the index value is much smaller than the width of the actual data value. Consequently, fewer off-chip bus lines need to be activated for transmission. The details of our power protocol are explained in Section 3.

We applied this strategy to fifteen benchmark codes from the embedded systems area. The results given in Section 4 indicate that data bus power savings are around 54.8% on the average. Section 4 also evaluates the energy/performance overheads due to our optimization strategy. We compare our approach to two popular bus encoding schemes, namely, 1-to-2 encoding and Gray encoding, as well as a performance-oriented bus width enhancement scheme. Our strategy can also be integrated with these encoding mechanisms to further reduce the power consumption on the data bus. We report on results from an implementation that combines power protocol with 1-to-2 encoding. These results indicate that the combined optimization strategy can improve off-chip data bus power consumption by 67.8% on the average. More importantly, the combined strategy reduces the overall power dissipation for all value cache sizes used in our experiments. These improvements come from both reduced number of bus line activations and reduced bit switching activity on the bus. Based on the results of our experiments, we conclude that power protocol is an effective technique in reducing energy consumption. In Section 5, we summarize our major contributions.

2. Related Work

Most of the existing work in bus encoding aims towards improving the performance and effective bandwidth of the bus. For example, Citron and Rudolph have described a technique to encode data on the bus using a table-based approach [9]. In this method, they divide a given data item (to be transferred over the bus) into two parts. The lower part is sent without being encoded, while the upper part is encoded using a look-up table (LUT). Their method aims at reducing the size of the data being sent. The upper part of the data is broken into a key, which is used to search the data in the LUT, and a tag, which is used to match the data in the LUT to find if there is a hit. If there is a hit, the data sent over the bus is compacted to y [y=Comp(x)], where x is the original data and the lower d bits of y is the same as the lower d bits of x. On the receiving end, a component called bus expander expands y to original data x [x=Expn(y)]. It follows that x=Expn(Comp(x)). Citron and Rudolph have used a bus of smaller width than the normal word size of the architecture. Our work is different from theirs in at least two aspects. First, the two studies use different encoding strategies. Second, our work tries to reduce the power dissipation on data bus rather than increasing the effective capacity of the bus. Later in the paper, we compare our technique quantitatively to the strategy proposed by Citron and Rudolph.

In the recent past, the encoding paradigms for reducing the switching activity on the bus lines have been investigated (e.g., [24, 23, 18]). Most of the common encoding techniques rely on the well-known spatial locality principle. An analysis of several existing low-power bus encoding techniques such as T0, bus inversion, and mixed encoding has been performed by Fronaciari et al [11]. Among the different techniques that have been studied for reducing the power dissipation of the address bus, Gray code addressing appears to be a very efficient scheme [18]. Encoding techniques at the system-level have been reviewed by Stan and Burleson [24]. Shin, Chac, and Choi have explored a partial bus invert coding technique where they proposed a heuristic to select only portions of the bus to invert for further improving the effectiveness of the bus inversion coding [20]. They have also proposed a combination of bus inversion with transitional signaling for a typical signal processing application [21].

Bus inversion is known to perform very well in the arena of power reduction over the data bus [5]. In bus inversion, a data item is sent as it is, or in a bit-inverted form, depending on the state of the bus during the previous transaction. The main idea is to reduce the dynamic power dissipation by reducing the number bit swings (that is, to minimize the number of charging- discharging of the capacitive lines) of the bus. A similar technique is the 1-to-2 encoding, which uses the same principle as bus-inversion, but is static in nature [24]. If 'n' bits are required to represent all the data, 1-to-2 encoding uses 'n+1' bits to encode them. In this encoding scheme, each data 'x' is represented either as its binary 'X', or as the bit inverted form of the binary equivalent ' $\sim$X'. Therefore, there are two codes to choose from, namely, 'X' or '$\sim$X', following the principle of minimizing the number of bit switches in the consecutive bus transactions. Thus, the number of bit transitions on the bus is reduced, thereby reducing the power consumption. Our work is different from these studies in the sense that rather than trying to minimize

the bit switching activity, it is more oriented towards reducing the width of the data to be sent over the bus; this also reduces the bit switching activity. Also, as our experiments demonstrate, the effectiveness of our strategy can be further increased by combining it with 1-to-2 encoding strategy.

Techniques using look-up tables to compress data have been explored by Bishop and Bahuman who have proposed an adaptive bus coding to reduce bit transition activity by exploiting the repetition of the data [3]. Childers and Nakra have studied the effect of reordering the memory bus traffic to reduce power dissipation [7]. Yang et al [27] have proposed the use of a compression cache to compress the data stored in the cache line to at least half their length so that a single cache line can effectively store two cache-lines of data. However, their main objective is to compress the data in the on-chip cache and increase the cache hit rate. Since their approach sends the data across the bus in a compressed form, it also results in a reduction in bus activity. Also, the compression strategies used in our work and in [27] are entirely different. Farrens and Park [10] have presented a strategy that caches the higher order portions of address references in a set of dynamically allocated base registers. Our bus energy optimization strategy is also similar to the strategy proposed by Yang and Gupta [26]. However, our value cache update strategy is different from theirs and we evaluate the energy behavior system wide.

3. Power Protocol

In this section, we present the details of our power protocol that tries to reduce the power dissipation on off-chip data buses.

3.1. Objective

Our objective is to reduce the power dissipation on off-chip data bus. As mentioned earlier, this can be achieved in at least two ways: (i) reducing the width of the data transferred over the bus, and (ii) reducing the switching activity on the bus. Our approach tries to achieve the first option by exploiting the locality of the data values communicated over the off-chip data bus. However, once the width of the data (that needs to be transmitted) has been reduced, we can also expect a reduction (in general) on the average bit switching activity per transfer. In addition, this switching activity can be further reduced by using well-known bus encoding schemes in conjunction with our strategy (e.g., 1-to-2 encoding). In the following, we first discuss our encoding strategy, and then give an illustrative example.

3.2. Protocol Details

In the proposed protocol, we focus on the data values transferred over the off-chip data bus, and use an adaptive, application- independent technique to reduce the power dissipation of the bus. This approach uses two small look-up tables called *Value Caches* (VCs): one at the memory side of the data bus (off-chip) and one at the cache side (on-chip). These VCs are used to store the value of the data that is being sent over the bus (see Figure 2).

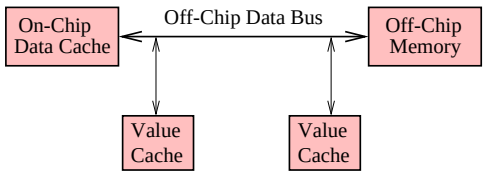

Figure 2. An abstract view of the power protocol architecture. We employ a value cache at each side of the off-chip data bus. The value caches keep track of the data values that have recently been transmitted over the bus. When a data value needs to be transmitted over the bus, we first check whether it is in the value cache of the sender. If it is, we transmit only the index of the data (i.e., its value cache address) instead of the actual data value and, the other side (receiver) can determine the data value by using this index and its value cache. The objective here is to reduce the number of active bus lines needed for data transfer.

We use a value-based caching strategy in our protocol instead of the traditional address-based caching. More specifically, when the processor requests a data item and it is not found in the conventional on-chip cache, the address is sent to the off-chip memory. There, the value of the data is found from the memory using the address received from the processor. This value 'v' of the data 'x' is used to search the VC. If the value of the data to be sent is not found in the VC (a VC miss), the verbatim data, which is 'W' bits wide, is sent over the bus to the receiver side (on-chip cache). At the same time, the value of the data is cached in the sender's VC at some index location 'i'. The index 'i' depends on the placement and replacement policy implemented for the VC on the sender side.

The receiver side runs the same placement and replacement policy for the VC as the sender. Thus, the value of the data sent over the bus is copied in the receiver's VC at the *same* index location, as in the sender's VC. So, by construction, it is guaranteed that the VCs in both the sides have the same values of data in the same positions at any point in time. In other words, the same data item can be addressed by both sender and receiver using the same index (to VCs).

On the other hand, if the value 'v' of the data 'x' to be transferred is found at some location indexed 'j' in the VC of the sender (a VC hit), there is no need to transfer the entire W-bit wide data. Instead, we only transfer the value of index 'j' over the bus, which is only 'w'-bits wide ($w<W$), where w depends on the size of the VC used. At the receiver side, we use this index to retrieve the actual value of data from the receiver's VC. The protocol ensures that the same value of data is present at the same index location of the receiver's VC. We use one extra control bit to indicate whether the data being sent over the bus is the verbatim data

or an index to the VC. A memory-write activity is handled in a similar fashion.

3.3. Example

As an illustrative example, we consider the sequence of bus transactions given in Figure 3. We assume that initially the values 100 and 200 are not present in the VC. During Transaction #1, A is sent from the memory to the cache. The requested data item, stored at some address (say address X of the memory) has a value 100. The memory controller searches the VC for the value 100 and detects a miss. Therefore, the value 100 is sent over the off-chip data bus. Also, following our power protocol, the value 100 is stored at the same location (say 5) of the VCs of both the source and the destination. For Transaction #2, the memory controller searches for the value 200, cannot find the value in the VC, and repeats the steps as described above. At this point the value caches at the both ends contain data values 100 and 200. In Transaction #3, the memory controller finds that a value 100 has to be sent to serve a read request (of the same memory location as before, or of a different memory location that has the same value). But, note that the value 100 is already present in the VC of the sender in location 5 as a result of Transaction #1. Therefore, instead of sending the value 100, the memory controller just sends the index value 5. The receiver, on the other hand, fetches the value of the actual data (100 in this case) from location 5 of its VC. Finally, in Transaction #4, we want to send the data item D having a value 200 to the memory (i.e., a write request). But, the value 200 is already cached in both the VCs as a result of Transaction #2 from memory to cache. Consequently, the index to the cached copy (present in the VC) of value 200 is used to complete Transaction #4 but in the reverse direction. This last transaction shows that, in power protocol, the data placed in the VCs during a transaction in one direction can be re-used (from the VC) during a transaction in the reverse direction.

Transaction Id	Transaction Direction	Data	Value	Status	Transmitted Entity
#1	Memory to Cache	A	100	Miss in VC	Full Data
#2	Memory to Cache	B	200	Miss in VC	Full Data
#3	Memory to Cache	C	100	Hit in VC	VC Index
#4	Cache to Memory	D	200	Hit in VC	VC Index

Figure 3. Example bus transactions using power protocol. Note that in the last two transactions, VC access generates hits. We also note that since VC operates with values and is placed at both ends of the bus, we achieve a hit as long as the same value needs to be transferred on the data bus. It does not matter whether the value belongs to the same variable or not; or it is transmitted from memory to cache or vice versa.

3.4. Discussion

Note that we add an additional layer of indirection between the two ends of the off-chip data bus. This might degrade performance by increasing the effective memory access time. However, as compared to the memory access latency (approaching several 100s of cycles), the latency introduced by the power protocol should be comparatively low. Moreover, when memory is accessed, the entire block of data is moved across the data bus. In such a scenario, the additional delay due to the VC protocol would affect only the first transaction, and all the subsequent accesses to the block would be found in the on-chip data cache (as long as the block is in the data cache). So, we can expect that power protocol would reduce the power consumption of the off-chip data bus without significantly affecting the performance of the system.

It should be stressed that while so far we have discussed our power protocol assuming that the one end of the bus is on-chip cache and the other end is off-chip main memory, it is possible to adapt our strategy to work with an off-chip L2 cache as well. In addition, power protocol can also be used to reduce the switching activity between on-chip L1 cache and on-chip L2 cache (although the results would not be as good as those with the off-chip bus). In fact, our strategy can be used between any two communicating devices in the system (with the VC support). Further, we are not restricted by point-to-point configurations. That is, our approach can be made to work in an environment where multiple devices are communicating over a shared (power-hungry) data bus. Obviously, in this case, among other things, we would need a coherence mechanism (the discussion of which is beyond the scope of this paper). A drawback of our strategy is the extra space needed by two value caches (one on-chip and the other off-chip). In this paper, we do not present a detailed study of the circuit space implications of our approach. As will be presented in the experimental results section, even a small VC (128 entries) generates reasonably good energy behavior; so, we can expect that the space overhead due to our optimization will not be excessive.

4. Experiments

4.1. Simulation Environment and Benchmarks

We enhanced the SimpleScalar/Arm architecture simulator [2] for our experiments. The SimpleScalar simulator, originally developed at the University of Wisconsin, models a modern microprocessor with a five-stage pipeline: fetch, decode, issue, write-back, and commit. We implemented our power protocol technique within the framework of the sim-outorder tool from the SimpleScalar suite, extended with the ARM-ISA support at the University of Michigan [1]. Specifically, we modeled a processor architecture sim-

ilar to that of Intel StrongARM SA-1100. StrongARM SA-1100 is the second member of the StrongARM family. It is a highly-integrated communications microcontroller that provides power efficiency and low cost. The modeled architecture has a 16KB direct-mapped instruction cache and a 8KB direct-mapped data cache. Both the cache sizes are of 32 byte-length. Both instruction cache and data cache have 1 cycle latencies. Off-chip memory latency is assumed to be 100 cycles. We extended the simulator such that the stores are made using a four-line write buffer (as in SA-1100). We also model a 256-entry full associative TLB with a 30-cycle miss latency. The off-chip bus is 32 bit-wide. To obtain the power numbers, we used the Wattch simulator from Princeton University (with parameters for 0.18 micron technology) [5].

To test the effectiveness of our power protocol, we used a benchmark suite that consists of 15 embedded applications. These binaries are built on a NetWinder Developer ARM workstation from Rebel Systems (www.rebel.com) using GNU GCC version 2.95.1 with optimization enabled (-O), and compiled to use the Linux/ARM system calls. Eleven of these applications are from the MediaBench suite [15]. Two of the remaining applications (`hier` and `parallel-hier`) are motion estimation codes [28] frequently used in embedded video applications. `zip` is a data compression code based on the Lempel-Ziv algorithm; it uses entropy encoding. Finally, `des` is a cryptography code. It is a block cipher that transforms 64-bit data blocks under a 56-bit secret key, by means of permutation and substitution. It is frequently employed in smart-card applications. We believe that this benchmark set is representative for embedded applications that can be found on the market. The important characteristics of these applications are listed in Figure 4. The third, fourth, and fifth columns in this table give, respectively, the number of execution cycles, the number of off-chip data references, and the base energy consumption. The base energy consumption includes the energies spent in on-chip components, off-chip buses and off-chip memory, and obtained assuming a 1.5V/300 MHz embedded (SA-1100 like) processor (0.18 micron technology) with the configuration parameters mentioned in the previous paragraph. Later in the paper, we modify some of these parameters to measure the robustness of our strategy.

Benchmark Code	Brief Description	Number of Cycles	Number of Off-Chip Refs	Energy Consumption
`adpcm`	Adaptive audio coding	14.72	1.18	31.65
`epic`	Experimental image compression	73.91	8.82	45.18
`g.721`	CCITT voice compression	377.14	57.02	55.27
`ghostscript`	Postscript interpreter	57.46	21.71	29.79
`gsm`	European standard for speech coding	298.04	19.88	42.83
`jpeg`	Lossy compression for still images	31.61	2.80	31.81
`mesa`	Public OpenGL clone	51.80	3.34	41.80
`mpeg`	Lossy compression for video	1,321.27	145.36	82.11
`pegwit`	Elliptical public-key encryption	162.92	20.18	87.54
`pgp`	IDEA/RSA public-key encryption	318.73	21.24	60.05
`rasta`	Speech recognition	177.33	13.61	90.28
`hier`	Motion estimation algorithm	1,418.70	220.04	115.11
`parallel-hier`	Motion estimation algorithm	1,387.68	150.63	108.97
`zip`	Lempel-Ziv data compression	1,698.54	252.73	126.58
`des`	DES crypto algorithm	1,587.82	236.42	119.07

Figure 4. Benchmarks used in our experiments. The third, fourth, and fifth columns in this table give, respectively, the number of execution cycles (in millions), the number of off-chip data references (in millions), and the base energy consumption (in millijoules). The base energy consumption includes the energies spent in on-chip components, off-chip buses and off-chip memory. The first eleven applications are from MediaBench; the rest are frequently-used embedded applications.

4.2. Measures of Interest

There are several performance measures of interest in evaluating the proposed strategy. These include the hit rate of the VCs, the volume of data transferred over the bus as compared to the volume of data transferred during normal bus transaction (i.e., without power protocol), the bus switching activity, and the reductions obtained in power dissipation of the off-chip data bus. It should be noted that while increasing the size of VC is expected to increase its hit rate, this may not always be the best choice as a larger VC will result in wider indices to be sent over the data bus, leading to more bus line activations and switching activity. Also, since our VCs themselves consume some power and their use might increase overall execution time, it is also important to evaluate the overall (system-wide) energy consumption (rather than just the off-chip data bus power dissipation). In the following subsections, we evaluate these measures in detail. In all the results presented in this section, the extra (control) bit needed by power protocol has been accounted for.

4.3. VC Hit Rates

The hit rate of the VC is given by the fraction of times the value of the data to be sent across the off-chip bus is found to be present in the VC over the total number of off-chip data bus transactions. The hit rate clearly depends on the size of VC (denoted using $|\mathrm{VC}|$). We use different sizes of VCs (from 8 words to 2K words) to measure the impact of VC size on the effectiveness of our power protocol. Since both power dissipation and performance are the parameters important for us, we cannot use a very large VC. In fact, a very large VC would most probably lead to excessive power dissipation, thereby offsetting the potential improvements achieved due to the protocol. The hit rate also depends on the replacement policy used. In general, an implementation can have two alternatives for that. Either we need to run a deterministic replacement policy on both the sender and the receiver sides, or send the index along with the actual verbatim data whenever there is a miss. This is necessary because, as mentioned earlier, at any point in time, we want to have the same data value in the same location of the VCs of both the sender and the receiver. However, sending more data is not desirable since we want to reduce the volume of data transfer (and save power). Therefore, in this work, we chose to experiment with deterministic policies only. The replacement policies used in this paper are the Least Recently Used (LRU) and the Least Frequently Used (LFU).

Figure 5 gives the hit rates for VCs of different capacities

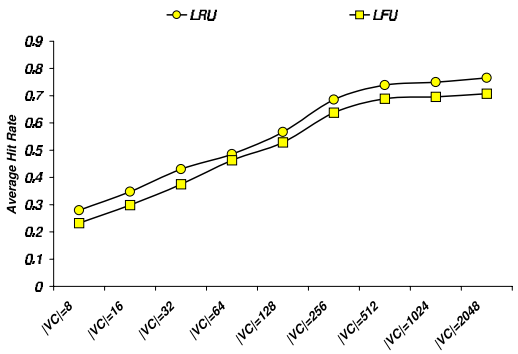

Figure 5. VC hit rates. These results indicate that as the VC capacity is increased the hit rate also increases. However, beyond a certain point, further increase in the VC capacity does not pay off much. We also see that LRU management generates better results than LFU management. These hit rates also indicate that there exists some degree of locality among the data values transmitted over the bus.

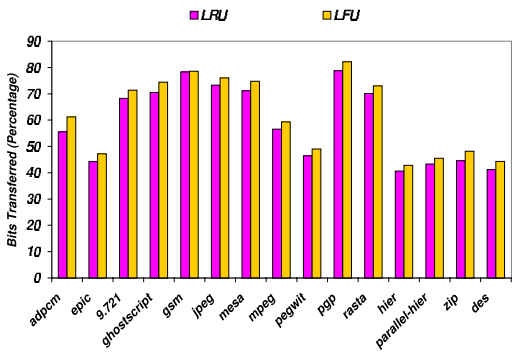

Figure 6. Traffic over the data bus due to power protocol (|VC|=128). These results show that there is approximately a 41.1% (resp. 38.2%) reduction in the volume of traffic through the data bus when LRU (resp. LFU) is used.

averaged over all fifteen benchmarks in our experimental suite. These results indicate that as we increase the size of the VC, the hit rate also increases. Note that the hit rate of the VC characterizes the fraction of times a particular data value is found in the VC during the off-chip data transfer. Also, VC is only searched for those data items that are not found in the on-chip data cache. Further, VC tries to capture data values and not addresses. Given these, we believe that the VC hit rates observed in Figure 5 are reasonably good. In fact, these hit rates indicate that there exists some degree of locality among the data values transferred over the off-chip bus. On an average, the hit rate increases as the size of the VCs increases, reaches a peak, and then saturates. This leads to a trade-off between the size of the VCs and the gain that we can have due to reduction in the volume of data transferred over the bus as a result of power protocol. Clearly, larger VCs themselves will consume more power and will be slower. Therefore, additional diminishing gains must be considered against the overhead of larger VCs. In fact, none of our benchmarks showed any significant improvement in hit rate when we increase the VC size beyond 512 words (the detailed results are not presented here). To conclude, it is encouraging to observe that even a VC less than 512 words is able to capture the value locality in off-chip data transfers to a large extent.

4.4. Traffic through the Data Bus

Traffic through the data bus is measured by the number of bits that are actually sent over the bus as a result of the off-chip bus transactions. This traffic is largely dependent on the hit rate of the VC. When there is a hit, we send the index (the position of the data value in the VC) instead of the actual data. Clearly, higher the VC hit rate, smaller is the number of times the W-bit (full-width) data needs to be sent over the bus. Therefore, on the average, we send fewer number of bits and this can significantly save power by allowing us to switch off the unused portions of the bus. Most commercially available buses have the capability of powering off individual bytes [25]. In other words, a data bus that is 32 bits wide, can actually transfer 8, 16, 24, or 32 bits of data. Thus, the available technology could enable us to implement a 256-entry VC (indexed by 8 bits) without any significant additional hardware for bus-control.

Figure 6 shows, for LRU and LFU VC replacement policies, the number of bits actually transferred over the data bus (with power protocol) as a fraction of the bits transferred without the use of VC (i.e., without power protocol). The VC capacity used is 128 entries (i.e., $|VC|=128$). We observe from these results that, on an average, there is approximately a 41.1% (resp. 38.2%) reduction in the volume of traffic through the data bus when LRU (resp. LFU) is used. To see how this data bus traffic gets affected when the VC capacity is changed, we performed another set of experiments. The results given in Figure 7 are the average values (i.e., the number of bits normalized with respect to the case without any optimization) across all fifteen benchmark codes in our suite. We observe that the bus traffic decreases with an increase in the VC size up to a certain point, beyond which it starts increasing. Thus, there is another trade-off at this level. As we increase the size of the VC, the number of bits required to address the VC also increases. Even though the number of VC hits increases, a larger size of the VC results in transferring a larger number of index bits. This in many cases offsets the advantage of the higher VC hit rate. For example, we see from Figure 7 that the total bus traffic with $|VC|=2048$ is higher than that with $|VC|=256$.

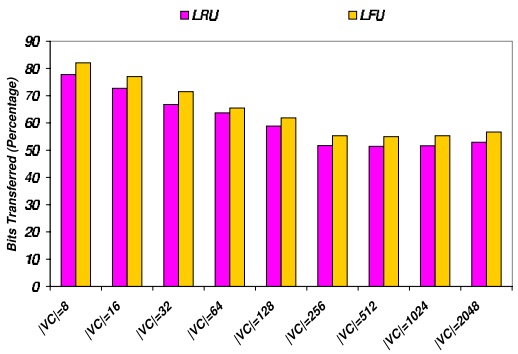

Figure 7. Sensitivity of the bus traffic to the VC size (averaged over all benchmarks). It can be observed that increasing the VC capacity arbitrarily may not be a good idea. A larger size of the VC yields better hit rate but it also results in transferring a larger number of index bits over the bus.

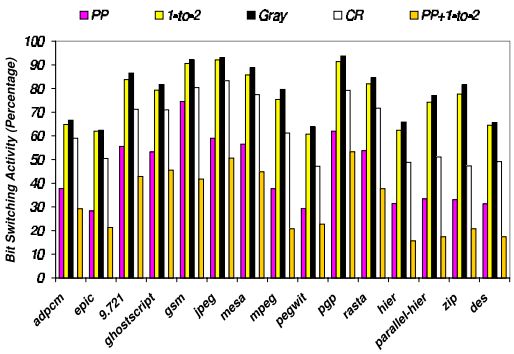

Figure 8. Bit switching activity (SA) on the data bus (|VC|=128). Average saving on the bit switching activity for power protocol is around 55%. We see that power protocol outperforms three previously-proposed bus encoding strategies. These results clearly show that power protocol outperforms three previously-proposed encoding strategies. Combining 1-to-2 encoding with our strategy brings an additional 12.9% saving on the average. Note that these savings in bit SA directly translates to power savings on the off-chip data bus.

4.5. Switching Activity on the Data Bus

Switching activity (SA) on the off-chip data bus gives the number of times the bus lines are discharged and recharged between ”0” and ”1”, and is directly responsible for the dynamic power dissipation on the bus. It is well-known that a smaller value of SA implies reduction in dynamic power consumption. The first bar (denoted PP) for each benchmark in Figure 8 gives the amount of bit switching activity when using power protocol as a fraction of the switching activity for the original case (i.e., without power protocol). We consider only the LRU policy as it always outperforms the LFU policy. We observe that, on the average, the bit switching activity is reduced by around 55%.

To see how previously-proposed bus encoding schemes compare to ours, we give in the second and the third bars for each application in Figure 8 the bit switching activities due to 1-to-2 encoding and Gray encoding, respectively. As before, these are the values normalized with respect to the switching activity of the original scheme (without any encoding). We see that the normalized bit switchings for 1-to-2 encoding and Gray encoding are 76.4 and 78.8, respectively. These values are much higher than the corresponding (normalized) value with our power protocol (45.1), indicating that our strategy outperforms these low-power encoding schemes. The fourth bar in Figure 8 shows (for each benchmark) the bit switching activity when the encoding strategy proposed by Citron and Rudolph (denoted CR in the figure) [9] is employed. For a fair comparison, both power protocol and CR use the same-sized cache structures (VCs and LUTs). In selecting the best cache configuration for the CR strategy, we experimented with different associativities and found that a four-way set-associative cache (LUT) generates the best results. The results in Figure 8 indicate that our power protocol strategy outperforms the remaining three strategies. The reason that it performs better than 1-to-2 encoding and Gray encoding is that these two strategies try to reduce only the switching activity and do not attempt to reduce the number of active bus lines. Our experience shows that reducing the number of active bus lines is, in general, more effective than reducing the bit activity. Our approach also results in a better power consumption behavior than CR. One reason for this is the poor hit rate of the cache (look-up table, LUT) used in CR. The mechanism used in [9] indexes the cache (LUT) using some parts of the high-order bits of the data value. We found out that such an addressing/caching strategy does not result in very good locality as far as data values are concerned. In particular, with the small cache size, the hit rate for the look-up table was around 28-30% on the average (across all benchmarks). Also, the portion of the data that is sent over the bus un-encoded in the CR method effectively increases the amount of bit switching activity on the data bus. In contrast, in power protocol, we store the entire data value in the (value) cache and encode all of it.

There is a potential to reduce the switching activity on the bus further if we can encode the data and the indexes in such a way that the number of bits changing between two consecutive bus transactions is minimized. Consequently, we also combined 1-to-2 encoding strategy with our power protocol. As mentioned earlier, 1-to-2 encoding uses the same principle as bus inversion [24]. For each transaction, this new combined strategy uses an additional bit to represent the encoded data, as compared to the normal power protocol described earlier. Therefore, one more bit is transferred for each bus transaction as compared to our base power protocol technique. But, the advantage of this combined technique is that it can reduce the overall bit switching

activity. The last bar for each application in Figure 8 (denoted PP+1-to-2) shows the bit activity of the VC scheme (under LRU) which incorporates 1-to-2 encoding. We observe from this figure that, using 1-to-2 encoding, the off-chip data bus bit activity is reduced to 32.2% (of the original bit activity without the power protocol) on an average, which is an additional 12.9% improvement over power protocol. When we combined power protocol with Gray encoding, we obtained very similar results; so, they are not presented here.

We also studied the sensitivity of these savings in the bit switching activity to the VC capacity. The first bar (denoted PP) in Figure 9 shows the bit switching activity due to power protocol normalized to that of the original case (i.e., without power protocol). On an average, power protocol leads to significant reductions in the bit switching activity across all VC sizes. We see that the bit switching activity is initially reduced with an increase in the VC size. But, there is an optimal VC size beyond which the SA starts increasing. The point where the SA starts increasing is the same as the point where the increase in the size of the VC does not improve the VC hit rate too much. But, at the same time, there is an increase in the number of index bits (which is dependent on the size of the VC) that have to be sent over the bus for each hit, and as a result, there is an increase in the overall bit switching activity. As we can see from Figure 9, in general, a VC size of 256 words yields a very good bit switching activity. On an average (across all VC capacities), the power protocol reduces the SA on the off-chip data bus by 69.8% as compared to the normal data transfer over the bus. Therefore, since the bit switching has a direct impact on power consumption of data buses, we can conclude that our power protocol reduces the off-chip bus power consumption significantly. The second bar in Figure 9 (denoted CR) for each VC size gives the average bit switching activity (across all benchmarks) when the strategy in [9] is employed. We see that our strategy outperforms the one in [9] for all VC sizes due to the reasons explained above. Finally, the last bar (for each VC size) in Figure 9 shows the average switching activity when our strategy is combined with 1-to-2 encoding. It can be observed that, with large VC sizes, the original switching activity is reduced by nearly 80% when the combined strategy is employed.

4.6. Overall Energy Savings

In the previous subsection, we found that our power protocol reduces the bit switching activity of the off-chip data bus to a great extent. In fact, with large VC sizes, the power protocol augmented by 1-to-2 encoding reduced the switching activity of the data bus to 20% of the original bus activity. The power consumption of the bus varies linearly with the amount of switching activity on the bus. Consequently, we can conclude that the dynamic power consumption on the off-chip data bus is reduced by up to 80% when power protocol is used with large VCs. However, as noted earlier, the VCs themselves add to the power consumption of the system. Therefore, we need to evaluate the effect of our VC-based strategy on the overall energy consumption of the system. Also, as mentioned earlier, the proposed protocol has an impact on execution time (due to VC look-ups); this may lead to extra power consumption in other system components. Consequently, in this subsection, we consider the energy consumption of the entire system. First, we give power models used for off-chip data bus and VCs in detail. After that, we present energy numbers.

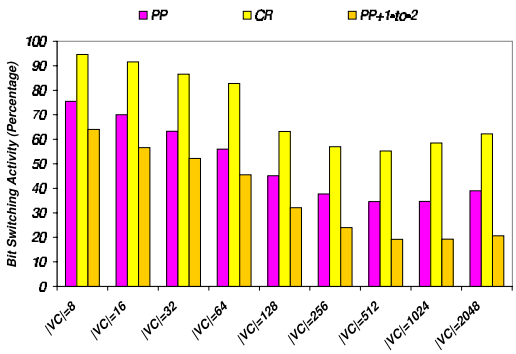

Figure 9. Sensitivity of the bit switching activity (SA) to the VC size (averaged over all benchmarks). As before, there is a tradeoff between addition savings due to a large VC and energy loss due to the larger number of bits to be transferred.

4.6.1 Bus Power Model

We use a model similar to the one discussed by Catthoor et al [6]. In general, estimating energy consumed by off-chip interconnects is difficult. We can approximate the capacitance for the bus (C_{bus}) using:

$$C_{bus} = C_{metal} * \text{Number of Bus Lines}$$

In this expression, C_{metal} is the capacitance of the metal interconnect for each bus line. Using the numbers in [6], it is estimated to be 20pF. 6pF of this is due to memory I/O pin, and 5pF is due to bus driver. The remaining components are off-chip interconnect, I/O pin, on-chip interconnect, and have capacitive loads of 3pF, 5pF, and 1 pF (for 0.18 micron technology), respectively. C_{bus} gives the effective capacitive load to be driven during a bus transaction, and we use the equation above to compute the energy consumption of the bus while running the benchmark codes in our experimental suite.

4.6.2 VC Power Model

Our VC is basically a small content addressable memory (CAM), and we use the Wattch power model (built on top of the model proposed by Palacharla et al [19]) of a CAM to evaluate the power consumed by the VC. Wattch models the taglines and matchlines, and computes the capacitive load for a CAM like structure as:

$C_{CAMtagline}$ = C_{gate}(CompareEnable)*NumberTags

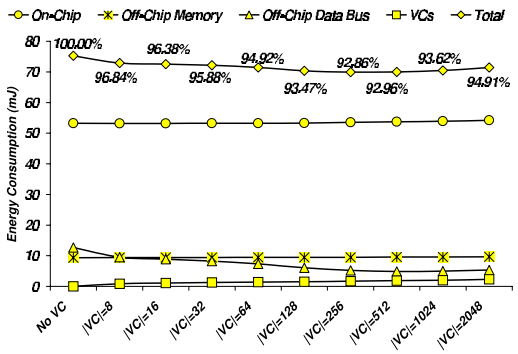

Figure 10. Energy consumption for different system components and overall energy savings. It can be seen that increasing the VC capacity reduces the energy consumption on the off-chip data bus; but, it also increases the energy spent in VCs. Therefore, arbitrarily increasing the VC capacity does not perform very well.

$$C_{CAMmatchline} = \begin{array}{l} +C_{diff}(\text{CompareDriver})+C_{metal}*\text{TLLength} \\ 2*C_{diff}(\text{CompareEnable})*\text{TagSize} \\ +C_{diff}(\text{MatchPreCharge})+C_{diff}(\text{MatchOR}) \\ +C_{metal}*\text{MLLength} \end{array}$$

Modeling the matchlines and taglines take into account the diffusion capacitance, the capacitance due to the driver, and the metal wire capacitance. For the matchline, we have the additional capacitance arising due to the huge OR structure that is present to detect the match. This model takes into account the number of tags (columns), the number of bits per tag-match, and the number of ports on the CAM (one in our case). The number of entries in the CAM is equal to the number of VC words. The CAM is associatively searched with the value of the data item. If there is a match and the valid bit is set, a read enable word line corresponding to the CAM entry is activated. An encoder (ROM) is used to encode the read enable word lines. The C_{gate}, C_{metal}, and C_{diff} parameters in the equations above are set for 0.18 micron technology. To reduce the per access energy consumption in VCs, we can also adopt the optimization proposed by Juan et al [14]. This strategy modifies the CAM cell by adding another transistor in the discharge path. As discussed in [14], this is motivated by the fact that, with the original CAM cell, to precharge the matchline, it is necessary to place 0 to all bit lines. Instead, in the modified cell, the control line is used to precharge the matchline.

4.6.3 Results

We compute the distribution of the energy consumption of the various components of the system with different sized VCs. VCs are modeled as the CAMs, and, as mentioned earlier, they themselves consume some amount of energy. The size of the VC directly affects the reduction in the volume of data being transferred and also reduction in the bit switching activity. Figure 10 gives the energy consumptions of different system components when VCs of different sizes are employed assuming 1 cycle extra latency in off-chip memory data accesses due to VC. The energy consumptions given here are the values averaged over all fifteen benchmarks in our experimental suite. The 'On-Chip' part includes the energy consumption in the processor datapath, instruction issue logic, caches, TLB, register file, and clock. 'Total' represents the sum of the energies consumed in 'On-Chip', 'Off-Chip Memory', 'Off-Chip Data Bus', and 'VCs'. Next to each point in 'Total', we give the normalized (percentage) energy value (with respect to the case with no VC). We see from these results that increasing the VC capacity reduces the energy consumption on the off-chip data bus; but, it also increases the energy spent in VCs. We observe from 'Total' that the best energy value is obtained when VC has 256 entries (an average energy saving of 7.14%). Using a larger VC leads to a higher energy consumption due to VC overhead. Since our strategy incurs 1 cycle extra latency during off-chip data accesses, we also observe slight energy variations (increases) in the 'On-Chip' and 'Off-Chip Memory' parts. Overall, we can conclude that employing power protocol brings energy benefits even if one includes its energy impact on other system components.

To see how these results are affected when the data cache size is modified, we performed another set of experiments. Figure 11 gives the normalized energy consumptions for different data cache sizes (from 2KB to 128KB on the x-axis) and VC capacities, averaged over all our benchmarks. Recall that our default data cache size was 8KB. All cache configurations experimented here have a line size of 32 bytes and are direct-mapped. We see that data cache capacity affects the effectiveness of our strategy to some extent. In particular, when the data cache size is very large, we do not have to much traffic on the off-chip data bus. Consequently, the impact of our power protocol on the overall system energy is reduced. We also note that a data cache size of 4KB generates the best savings from power protocol. In fact, with this data cache capacity, when $|\text{VC}|$ is 256, we achieve the best result, and save 11.9% overall system energy.

To study how much additional overall energy benefits will come when power protocol is combined with 1-to-2 encoding, we also measured the energy savings when the two techniques are applied together. The results are given in Figure 12. As in Figure 10, the values attached to 'Total' indicate the normalized (percentage) energy values (with respect to the case with no VC). We see from these results that the maximum energy saving occurs when VC size is 512, and in this case, we reduce the overall energy consumption by 10.07%. This means an additional 2.93% energy savings over the pure power protocol.

4.7. Impact on Performance

As mentioned earlier, our power protocol can also cause an increase in the number of execution cycles. This is because in off-chip accesses we may spend extra execution cycles during VC access. Figure 13 presents execution cy-

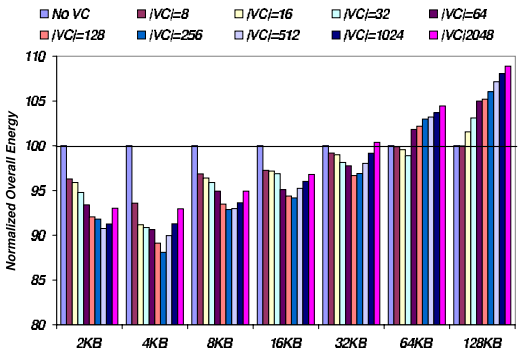

Figure 11. Normalized overall energy consumptions for different data cache sizes (x-axis) and VC capacities (averaged over all benchmarks). The bars in each data cache size are normalized with respect to the first bar (i.e., 'No VC'). Beyond a certain cache size it does not make sense to use power protocol as the inherent data bus activity becomes negligible.

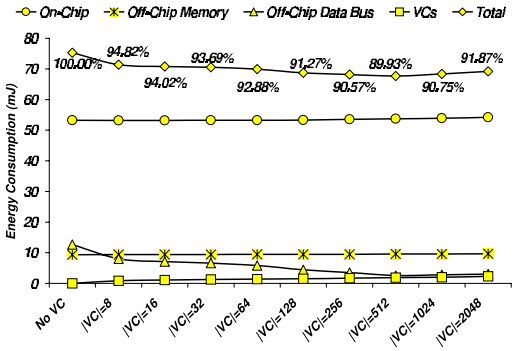

Figure 12. Energy consumption for different system components and overall energy savings when power protocol is combined with 1-to-2 encoding. The maximum energy saving occurs with a VC capacity of 512, and in this case, we reduce the overall energy by 10.07%.

cles normalized to the original execution cycles assuming an off-chip memory latency of 100 cycles (our default off-chip latency value). For each VC capacity, we present three different results corresponding to the cases where an off-chip memory access through VC costs extra 1 cycle, 5 cycles, and 10 cycles. Figure 14 gives similar results when the off-chip memory latency is 50 cycles. We see that when we assume 1 cycle overhead, the impact on performance was less than 1% (resp. 2%) with a 100 cycle (resp. 50 cycle) off-chip access latency. Therefore, we can conclude that power protocol improves energy consumption without much impact on performance.

While the values (extra cycle overheads) such as 5 cycles or 10 cycles may not seem to be very realistic, there is an interesting optimization that one might want to consider. There are some new TLB architectures (e.g., [8]) where the main TLB is partitioned into two or more sub-TLBs. The idea behind this is to reduce the dynamic energy consumption per TLB access. However, such strategies can lead to performance loss when these sub-TLBs are checked sequentially. Therefore, there exists a tradeoff between the energy gain due to access to a smaller TLB and the potential performance loss. Such partitioning techniques can be applied to our VC as well. If such a partitioned VC is utilized in conjunction with our power protocol, we can expect that the memory accesses through VC can take more than 1 cycle. The results in Figure 13 indicate that even when we incur 5 cycle average extra penalty (which is too high even for a partitioned VC), the performance overhead is less than 3% for all VC sizes (assuming a default off-chip latency of 100 cycles). Obviously, when off-chip memory latency is smaller (as in some embedded systems), we can expect a larger negative impact on performance. However, the advantage of employing such a partitioned VC is that the average per access energy consumption is reduced. A study of this tradeoff is in our future agenda.

5. Conclusions

In this paper, we started out with a goal to reduce the volume of data transfer and bit switching activity on the off-chip data bus in order to reduce dynamic power consumption of the buses, which are big consumers of power in modern embedded systems due to their high capacitive loads. We proposed a technique, called power protocol, which caches the data transferred on the off-chip bus on both sides of the bus in such a way that both caches (VCs) have identical entries. Thus, whenever a data value is found on the sender side of the VC, instead of sending the data, the corresponding index in its VC (which is much smaller than the actual data bit-width) is sent.

Simulation results using fifteen embedded benchmark programs showed that on an average the base VC scheme resulted in a 41.1% reduction in the volume of data transfer over the bus (when VC uses LRU). We also showed that base VC reduced the number of bit-transitions on the data bus by approximately 55%. Further enhancements to VC using 1-to-2 encoding showed that an additional 12.9% reduction in bit switching activity is possible. Since VC itself is a cache and therefore would consume power, it is important to keep its size and complexity low. Also, its energy and performance impact on the entire system should be quantified. Our experimental results revealed that using a VC of 256 bytes leads to a 7.14% reduction in overall energy consumption. Combining power protocol with 1-to-2 encoding took the overall energy saving up to 10%. The magnitude of savings depend on both the VC size and the data cache size. We also found that the impact of power protocol in execution time is small.

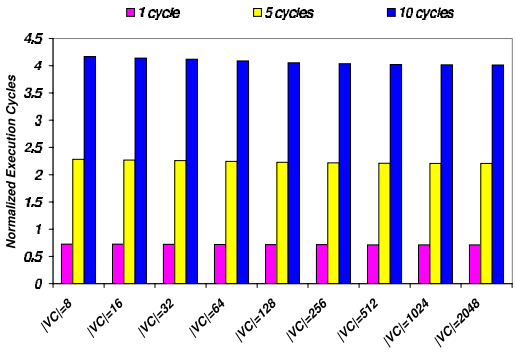

Figure 13. Percentage increase in execution cycles assuming an off-chip latency of 100 cycles. For each VC capacity, we present three different results corresponding to the cases where an off-chip memory access through VC costs extra 1 cycle, 5 cycles, and 10 cycles.

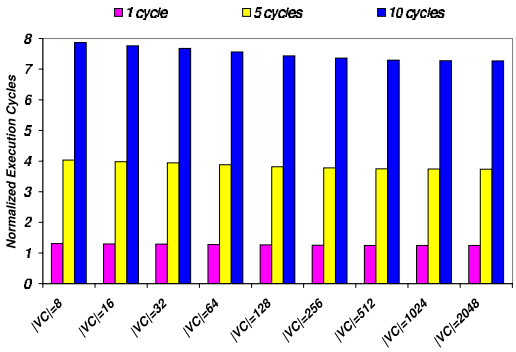

Figure 14. Percentage increase in execution cycles assuming an off-chip latency of 50 cycles. For each VC capacity, we present three different results corresponding to the cases where an off-chip memory access through VC costs extra 1 cycle, 5 cycles, and 10 cycles.

References

[1] T. M. Austin. The SimpleScalar/ARM Toolset. http://www.eecs.umich.edu/∼taustin/simplescalar

[2] T. M. Austin and D. Burger. The SimpleScalar Architectural Research Tool Set. http://www.cs.wisc.edu/∼mscalar/simplescalar.html

[3] B. Bishop and A. Bahuman. A low-energy adaptive bus coding scheme. In *Proc. IEEE Workshop on VLSI*, April 2001, Orlando, FL.

[4] D. Brooks, M. Martonosi, J-D. Wellman, and P. Bose. Power-performance modeling and tradeoff analysis for a high-end microprocessor. In *Proc. PACS 2000*, pp. 126–136.

[5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proc. International Symposium on Computer Architecture*, June 2000.

[6] F. Catthoor, S. Wuytack, E. De Gref, F. Balasa, L. Nachtergaele, and A. Vandecappelle. *Exploration of Memory Organisation for Embedded Multimedia System Design*, Kluwer Academic Publishers, 1998.

[7] B. R. Childers and T. Nakra. Re-ordering memory bus transactions for reduced power consumption. In *Proc. ACM SIGPLAN 2000 Workshop on Languages, Compilers and Tools for Embedded Systems*, June 2000.

[8] J-H. Choi, J-H. Lee , S-W. Jeong , S-D. Kim , and C. Weems. A low-power TLB structure for embedded systems. *IEEE Computer Architecture Letters*, Volume 1, January 2002.

[9] D. Citron and L. Rudolph. Creating a wider bus using caching techniques. In *Proc. IEEE Symposium on High Performance Computer Architecture*, January 1995, pp. 90–99.

[10] M. Farrens and A. Park. Dynamic base register caching: a technique for reducing address bus width. In *Proc. 18th Annual International Symposium on Computer Architecture*, Toronto, Canada, May 1991.

[11] W. Fornaciari, D. Sciuto, and C. Silvano. Power estimation for architectural exploration of HW/SW communication on system-level buses. In *Proc. International Workshop on Hardware/Software Codesign*, 1999.

[12] http://www.plastech.ru/html/techsupport/faqs/cel thermal.pdf

[13] http://www.cyberresearch.com/pciso64.html

[14] T. Juan, T. Lang, and J. J. Navarro. Reducing TLB power requirements. In Proc. *International Symposium on Low Power Electronics and Design*, 1997.

[15] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. MediaBench: a tool for evaluating and synthesizing multimedia and communication systems. In *Proc. the 30th International Symposium on Microarchitecture*, pp. 330–35, December 1997.

[16] K.-J. Lin and C.-W. Wu. A low-power CAM design for LZ data compression". *IEEE Transactions on Computers*, Vol. 49, No. 10, October 2000.

[17] D. Liu and C. Svensson. Power consumption estimation in CMOS VLSI chips. *IEEE Journal of Solid State Circuits*, vol. 29, no. 6, June 1994.

[18] H. Mehta, R. M. Owens, and M. J. Irwin. Some issues in Gray code addressing. In *Proc. 6th IEEE Great Lakes Symposium on VLSI*, 1996.

[19] S. Palacharla, N. P. Jouppi, and J. E. Smith. Complexity-effective superscalar processors. In *Proc. 24th International Symposium on Computer Architecture*, pp. 206–218, Denver, CO, June 1997.

[20] Y. Shin, S. Chae, and K. Choi. Partial bus-invert coding for power optimization of system level bus. In Proc. *International Symposium on Low Power Electronics and Design*, August 1998, pp.127–129.

[21] Y. Shin and K. Choi. Narrow bus encoding for low power systems. In Proc. *ASP-DAC2000*, January 2000, Yokohama, Japan.

[22] P. Sotiriadis and A. Chandrakasan. Low power bus coding techniques considering inter-wire capacitances. *IEEE Custom Integrated Circuits Conference 2000*, pp. 507–510.

[23] M. R. Stan and W. P. Burleson. Bus invert coding for low-power IO. *IEEE Transactions on VLSI Systems*, March 1995.

[24] M. R. Stan and W. P. Burleson. Low power encodings for global communication in CMOS VLSI. *IEEE Transactions on VLSI Systems*, December 1997.

[25] Techfest. PCI Local Bus Technical Summary. http://www.techfest.com/hardware/bus/pci.htm.

[26] J. Yang and R. Gupta. FV encoding for low-power data I/O. In *Proc. ACM/IEEE International Symposium on Low Power Electronics and Design*, Huntington Beach, CA, August 2001.

[27] J. Yang, Y. Zhang, and R. Gupta. Frequent value compression in data caches. In *Proc. 33rd IEEE/ACM International Symposium on Microarchitecture*, Monterey, CA, December 2000.

[28] N. D. Zervas, K. Masselos, and C. E. Goutis. Code transformations for embedded multimedia applications: impact on power and performance. In *Proc. ISCA Power-Driven Microarchitecture Workshop*, 1998.