Address Bus Encoding Techniques for System-Level Power Optimization

Luca Benini [$] Giovanni De Micheli [$] Enrico Macii [*] Donatella Sciuto [‡] Cristina Silvano [#]

[‡] Politecnico di Milano
Dip. di Elettronica e Informazione
Milano, ITALY 20133

[*] Politecnico di Torino
Dip. di Automatica e Informatica
Torino, ITALY 10129

[$] Stanford University
Computer Systems Laboratory
Stanford, CA 94305

[#] Università di Brescia
Dip. di Elettronica per l'Automazione
Brescia, ITALY 25123

Abstract

The power dissipated by system-level buses is the largest contribution to the global power of complex VLSI circuits. Therefore, the minimization of the switching activity at the I/O interfaces can provide significant savings on the overall power budget. This paper presents innovative encoding techniques suitable for minimizing the switching activity of system-level address buses. In particular, the schemes illustrated here target the reduction of the average number of bus line transitions per clock cycle. Experimental results, conducted on address streams generated by a real microprocessor, have demonstrated the effectiveness of the proposed methods.

1 Introduction

The increasing performance requirements and the existing gap between the speed of current microprocessors and the speed of the systems interfaces has pushed the designers to increase the bandwidth of the data transfers. For example, the PowerPC 620 processor can be configured with either a 64 or a 128-bit data bus to support the transfer of the 64-byte cache block within four clock cycles. Moreover, modern software applications span a very large address space: The latest versions of existing processors, such as the DEC Alpha AXP and the PowerPC 620, support a 64-bit address space. Hence, both data and address buses have become very wide for modern microprocessor-based systems.

Due to the intrinsic capacitances of the bus lines, a considerable amount of power is required at the I/O pins of a microprocessor when data have to be transmitted over the bus. More specifically, it has been estimated that the capacitance driven by the I/O nodes is usually much larger (up to three orders of magnitude [1]) than the one seen by the internal nodes of the microprocessor. As a consequence, dramatic optimizations of the average power consumption can be achieved by minimizing the number of transitions (i.e., the switching activity) on system-level buses.

Encoding paradigms for reducing the switching activity on the bus lines have been recently investigated. In [2], Stan and Burleson have proposed the use of a redundant encoding scheme, called the *bus-invert code*, to limit the average power. The bus-invert method performs well when patterns to be transmitted are randomly distributed in time and no information about pattern correlation is available. Therefore, the method seems to be appropriate for encoding the information traveling on data buses.

Concerning address buses, other techniques have been explored. They all rely on the well-known fact that the addresses generated by a microprocessor are often consecutive, due to the *spatial locality* principle [3], which states that the memory elements whose addresses are close to each other tend to be referenced during successive clock cycles.

To exploit this peculiar characteristics of the address buses, Su, Tsui, and Despain [4] have proposed to encode the patterns with the *Gray code*, since it guarantees a single bit transition when consecutive addresses are produced. The issue of modifying the Gray code so as to preserve the one-transition property for consecutive addresses of byte-addressable machines is discussed in [5].

The Gray code achieves the minimum switching activity among irredundant codes; however, better performance can be obtained by using redundant codes. In [6], we have proposed an encoding technique, called *T0 code*, which is based on the idea of avoiding the transfer of consecutive addresses on the bus by adding a redundant line, *INC*, to communicate to the receiving memory sub-system the information on the sequentiality of the addresses. The increments between consecutive patterns can be parametric, reflecting the addressability scheme adopted in the given architecture. The T0 code outperforms the Gray code in the case of streams of consecutive addresses of limited and unlimited lengths. Furthermore, savings are not offset by the power absorbed by the additional encoding/decoding circuitry.

The exploitation of the sequential behavior of addresses has also been approached, at a high level of abstraction, by Panda and Dutt [1]. That work introduces techniques for determining a mapping of the data to the physical memory which reduces the total switching activity on the address buses.

An additional contribution to the area of address bus encoding is provided in [7]. The target of the *Beach code* is that of reducing the bus activity in the cases where the percentage of in-sequence addresses is limited. In this situation, it may be possible to exploit other types of temporal correlations than arithmetic sequentiality that may exist between the patterns that are being transmitted over the address bus. More specifically, it has been noted that time-adjacent addresses usually show remarkably high block correlations. The encoding strategy is then determined depending on the particular stream being transmitted. Therefore, the Beach code best performs on special purpose systems, where a dedicated processor (e.g., core, DSP, micro-controller) repeatedly executes the same portion of embedded code.

In this paper, we propose innovative encoding techniques targeting the reduction of the average number of transitions on the address buses of microprocessor-based systems. Here, the data and address buses are at the core of the interface between the processor and the memory and I/O sub-systems. In particular, they are used, on-processor, to access the first and the second-level caches, as well as, off-processor, to access the external lower level caches, the main memory (usually through a memory controller), and the I/O sub-systems (to support direct memory accesses from the I/O controllers). Our goal is to avoid any modification to the standard memory components, hence adding the encoding circuitry inside the processor, and the decoding logic inside the memory and the I/O controllers.
In the next section, we summarize the main characteristics of currently available codes, and we compare their performance through analytical and experimental analysis. In Section 3 we propose new encoding schemes that combine the properties of existing approaches. This with the ultimate objective of exploiting the advantages offered by each technique, and thus identifying the best method for the specific microprocessor, the MIPS RISC [8], we have used for the experiments. Power and timing efficient implementations of the encoding/decoding logic for such encoding technique are also presented.

2 Previous Work

2.1 Bus-Invert Encoding

In [2], Stan and Burleson have proposed a redundant code, called the ***bus-invert*** code, to decrease the average power. A redundant bus line, called *INV*, is needed to signal to the receiving end of the bus which polarity is used for the transmission of the incoming pattern. The encoding depends on the Hamming distance (i.e., the number of bit differences) between the value of the encoded bus lines at time $t-1$ (also counting the redundant line at time $t-1$) and the value of the address bus lines at time t. The Hamming distance is compared to $N/2$, where N is the bus width (assuming N even without loss of generality). If the Hamming distance between two successive patterns is larger than $N/2$, the current address is transmitted with inverted polarity and the redundant line is asserted; otherwise, the current address is transmitted *as is*, and the *INV* line is de-asserted. The bus-invert method can be expressed by the following equation:

$$(\mathrm{B}^{(t)}, \mathrm{INV}^{(t)}) = \begin{cases} (\mathrm{b}^{(t)}, 0) & \text{if } \mathbf{H}^{(t)} \leq N/2 \\ (\bar{\mathrm{b}}^{(t)}, 1) & \text{if } \mathbf{H}^{(t)} > N/2 \end{cases} \tag{1}$$

where $\mathbf{B}^{(t)}$ is the value of the encoded bus lines at time t, $INV^{(t)}$ is the additional bus line, $\mathbf{b}^{(t)}$ is the address value at time t, $\mathbf{H}^{(t)} = (\mathbf{B}^{(t-1)}|INV^{(t-1)}, \mathbf{b}^{(t)}|0)$ is the Hamming distance, and N is the bus width of $\mathbf{b}^{(t)}$.
The corresponding decoding scheme is simply defined as:

$$\mathrm{b}^{(t)} = \begin{cases} \mathrm{B}^{(t)} & \text{if INV} = 0 \\ \bar{\mathrm{B}}^{(t)} & \text{if INV} = 1 \end{cases} \tag{2}$$

2.2 Asymptotic Zero-Transition Encoding

As mentioned in Section 1, the Gray code achieves its asymptotic best performance of a single transition per emitted address when infinite streams of consecutive addresses are considered. However, the code is optimum only in the class of irredundant codes, that is, codes that employ exactly N-bit patterns to encode a maximum of 2^N data words. Adding redundancy to the code, better performance can be achieved by adopting the T0 code [6]. The T0 method requires a redundant line, *INC*, to signal with value one that a consecutive stream of addresses is output on the bus. If *INC* is high, all other lines on the bus are frozen, to avoid unnecessary switchings. The new address is computed directly by the receiver. On the other hand, when two addresses are not consecutive, the *INC* line is low and the remaining bus lines are used as standard binary codes for the new addresses.
Obviously, this redundant code outperforms the Gray code on unlimited streams of consecutive addresses. Since all addresses are consecutive, the *INC* line is always asserted, and the bus lines never switch. As a consequence, the asymptotic performance of the T0 code is zero transitions per emitted consecutive address. The increments between consecutive patterns can be parametric, reflecting the addressability scheme adopted in the given architecture. In this respect, our code has the same capabilities of the Gray schemes reported in [5].
The T0 encoding scheme can be formally specified as follows:

$$(\mathrm{B}^{(t)}, \mathrm{INC}^{(t)}) = \begin{cases} (\mathrm{B}^{(t-1)}, 1) & \text{if } \mathrm{b}^{(t)} = \mathrm{b}^{(t-1)} + \mathrm{S} \\ (\mathrm{b}^{(t)}, 0) & \text{otherwise} \end{cases} \tag{3}$$

where $\mathbf{B}^{(t)}$ is the value on the encoded bus lines at time t, $INC^{(t)}$ is the additional bus line, $\mathbf{b}^{(t)}$ is the address value at time t and $\mathbf{S}$ is a constant power of 2, called ***stride***.
The corresponding decoding scheme can be formally defined as follows:

$$\mathrm{b}^{(t)} = \begin{cases} (\mathrm{b}^{(t-1)} + \mathrm{S}) & \text{if INC} = 1 \\ \mathrm{B}^{(t)} & \text{if INC} = 0 \end{cases} \tag{4}$$

2.3 Analytical Performance Comparison

Let us compare the two bus encoding techniques discussed so far, considering the binary code as the reference for the comparison. For the analysis we consider a stream of unlimited length whose addresses have a random uniform distribution, and a stream of unlimited length whose addresses are consecutive. Table 1 reports the results of the comparison, where N is the address bus width. Notice that, in the case of the bus-invert code, the average number of transitions per clock cycle, T_N, is given by:

$$T_N = \frac{1}{2^N} \cdot \sum_{k=0}^{N/2} k C_{N+1}^{k} \tag{5}$$

where C_N^k is the binomial coefficient.

Stream Type	Code	Avg. Trans. per Clock	Avg. Trans. per Clock per Line	Avg. I/O Pow. Diss.
Out-of-Seq Addresses	Binary	$N/2$	0.5	1
	T0	$N/2$	0.5	1
	Bus-Inv	T_N	T_N/N	$\frac{T_N}{N/2}$
In-Seq Addresses	Binary	$N/2$	0.5	1
	T0	0	0	0
	Bus-Inv	T_N	T_N/N	$\frac{T_N}{N/2}$

Table 1: Analytical Performance Comparison.

2.4 Experimental Performance Comparison

In this section, we first analyze the behavior of the addresses generated by a RISC microprocessor, then we provide experimental results for comparing the above discussed encoding techniques. The number of transitions occurring on the address bus during the execution of benchmark programs, derived from different application domains, has been used as the metric to estimate the power consumption.

Benchmark	Stream Length	In-Seq Addr.	Binary Trans.	T0		Bus-Invert	
				Trans.	Savings	Trans.	Savings
gzip	119102	60.16%	232587	140845	39.44%	232586	0.00%
gunzip	63884	64.94%	118409	81433	31.23%	118094	0.27%
ghostview	404595	57.28%	678295	470949	30.57%	678236	0.01%
espresso	1751673	60.17%	3014854	2239400	25.72%	3014649	0.01%
nova	544994	52.99%	959912	836708	12.83%	959334	0.00%
jedi	14690249	53.96%	23145174	18521606	19.98%	23145174	0.00%
latex	700317	71.42%	1400540	200490	85.68%	1400540	0.00%
matlab	6400326	76.56%	12000527	5800465	51.66%	12000527	0.00%
oracle	500326	69.99%	880547	680485	22.72%	880547	0.00%
Average		63.04%			35.52%		0.03%

Table 2: Experimental Comparison of Existing Encoding Schemes for Instruction Address Streams.

Benchmark	Stream Length	In-Seq Addr.	Binary Trans.	T0		Bus-Invert	
				Trans.	Savings	Trans.	Savings
gzip	34393	52.47%	131651	97157	26.20%	118184	10.23%
gunzip	14602	10.59%	92349	91739	0.66%	81953	11.26%
ghostview	112689	10.74%	554499	542323	2.20%	507987	8.39%
espresso	570750	5.56%	3956543	3945485	0.28%	3426699	13.39%
nova	220050	10.56%	1381975	1368763	0.95%	1204694	12.82%
jedi	6145049	12.67%	47268786	47156168	0.24%	41114510	13.02%
latex	200109	0.01%	763	771	-1.05%	558	26.87%
matlab	1204110	0.00%	5400375	5400385	0.00%	5351262	0.91%
oracle	120111	0.01%	466606	466616	0.00%	466039	0.12%
Average		11.39%			3.37%		10.78%

Table 3: Experimental Comparison of Existing Encoding Schemes for Data Address Streams.

Benchmark	Stream Length	In-Seq Addr.	Binary Trans.	T0		Bus-Invert	
				Trans.	Savings	Trans.	Savings
gzip	153495	57.43%	833799	738693	11.41%	809194	2.95%
gunzip	78486	52.81%	482093	427145	11.40%	406744	15.63%
ghostview	517284	58.25%	3067635	2795525	8.87%	2911494	5.09%
espresso	2322423	54.39%	15147444	13958602	7.85%	13404439	11.51%
nova	765045	56.63%	5526484	5325856	3.63%	5412030	2.07%
jedi	20835298	57.88%	152427486	142947798	6.22%	137950083	9.50%
latex	900426	55.55%	7002828	6202750	11.43%	5202521	25.71%
matlab	7604436	65.37%	42405967	34005893	19.81%	39474425	6.91%
oracle	620437	59.13%	3939551	3479461	11.68%	3594330	8.76%
Average		57.62%			10.25%		9.79%

Table 4: Experimental Comparison of Existing Encoding Schemes for Multiplexed Address Streams.

The addresses generated by a microprocessor obey to the spatial locality principle, that is, they are often in-sequence and, in general, they present a very high correlation. However, the things are quite different if the addresses correspond to instructions or data. In fact, addresses corresponding to instructions usually show a more sequential behavior with respect to data addresses. Instructions are usually stored in adjacent locations of the memory space, while only structured data such as arrays are generally stored in consecutive memory locations to achieve better locality. Clearly, there are exceptions to this behavior: Control-flow instructions cause interruptions in the sequence of consecutive addresses on the instruction flow, and data not stored in arrays are often accessed without any regular pattern. Nevertheless, sequential addressing dominates on average.

These considerations are supported by the data contained in Tables 2 to 4, where we report the percentage of in-sequence addresses measured on real address streams occurring on the 32-bit address bus of the MIPS microprocessor, when different benchmark programs are executed. Three distinct cases have been considered:

- The instruction address bus;
- The data address bus;
- The instruction/data multiplexed address bus.

The tables report also the number of transitions on the address bus, for the above mentioned classes of codes, and the percentage of transitions saved with respect to binary encoding.

The average percentage of sequential addresses in the benchmark streams is higher for instructions addresses (63.04%) than for data address streams (11.39%). This behavior can be due to the fact that references to automatic variables such as loop counters destroy the sequentiality of the address streams even if array data structures are accessed sequentially. As expected, when the probability of consecutive addresses is high, as for instruction addresses, a scheme such as T0 can be effective to reduce the transitions count, offering an average savings of 35.52% (see Table 2), while the bus-invert approach does not introduce any savings.

On the other hand, when the probability of in-sequence addresses is very low, as in the case of the data addresses reported in Table 3, the T0 code can provide only marginal advantages with respect to binary encoding (3.37%). The binary encoding can thus represent a good alternative for data addresses, since it does not require any redundancy and therefore no encoding and decoding circuitry. Among redundant codes, another alternative for data address buses can be represented by the bus-invert code, that is advantageous for the minimization of the average number of transitions (10.78% savings on average).

When the address bus is multiplexed (i.e., mixed instructions and data addresses), as in the MIPS architecture, the sequential behavior is often interrupted, when the selection signal switches from instruction to data and vice versa. Hence, the multiplexed address bus shows an intermediate behavior (48.64%) for the data in Table 4. As a matter of fact, the sequentiality of the addresses on the bus is somewhat reduced by the time multiplexing and by the inherent randomness of the data addresses.

3 Mixed Bus Encoding Techniques

The different effects of the encoding schemes discussed in Section 2 on the number of bus transitions have suggested us the use of combined techniques which may exploit the best properties of each method to further improve the overall system-level power budget. In this section, we introduce some novel encoding options, and we present transition count results collected during the execution of the usual benchmark programs.

3.1 T0_BI Encoding

For architectures based on a single address bus used to transmit both instruction and data addresses, as in the case of external second-level unified data and instruction caches, a new encoding can be applied to exploit the advantages provided by the T0 and bus-invert approaches. The combined code, called *T0_BI code*, reduces the average power and requires two redundant lines, *INC* and *INV*. The formal definition of the code is the following:

$$(\mathrm{B}^{(t)}, \mathrm{INC}^{(t)}, \mathrm{INV}^{(t)}) = \begin{cases} (\mathrm{B}^{(t-1)}, 1, 0) & \text{if } \mathrm{b}^{(t)} = \mathrm{b}^{(t-1)} + \mathrm{S} \\ (\mathrm{b}^{(t)}, 0, 0) & \text{if } \mathrm{b}^{(t)} \neq \mathrm{b}^{(t-1)} + \mathrm{S} \wedge \\ & \quad \mathrm{H}^{(t)} \leq (N+2)/2 \\ (\bar{\mathrm{b}}^{(t)}, 0, 1) & \text{if } \mathrm{b}^{(t)} \neq \mathrm{b}^{(t-1)} + \mathrm{S} \wedge \\ & \quad \mathrm{H}^{(t)} > (N+2)/2 \end{cases} \tag{6}$$

where $\mathbf{B}^{(t)}$ is the value of the encoded bus lines at time t, $INV^{(t)}$ and $INC^{(t)}$ are the additional bus lines, $\mathbf{b}^{(t)}$ is the address value at time t, $\mathbf{H}^{(t)} = (\mathbf{B}^{(t-1)}|INC^{(t-1)}|INV^{(t-1)}, \mathbf{b}^{(t)}|0|0)$ is the Hamming distance, and N is the bus width of $\mathbf{b}^{(t)}$.
The corresponding decoding scheme is given by:

$$\mathrm{b}^{(t)} = \begin{cases} \mathrm{B}^{(t)} & \text{if INC} = 0 \wedge \text{INV} = 0 \\ \bar{\mathrm{B}}^{(t)} & \text{if INC} = 0 \wedge \text{INV} = 1 \\ \mathrm{b}^{(t-1)} + \mathrm{S} & \text{if INC} = 1 \end{cases} \tag{7}$$

3.2 Dual_T0 Encoding

For architectures based on multiplexed address buses, such as the one of the MIPS microprocessor, two address streams α and β, with quite different behavior, are time-multiplexed on the same address bus. Stream α, corresponding to instruction addresses, has high probability of having two consecutive addresses on the bus in two successive clock cycles; on the contrary, stream β, corresponding to data addresses, has almost no in-sequence patterns. The control signal, *SEL*, available in the standard bus interface to de-multiplex the bus at the receiver side, is asserted when stream α is transmitted; otherwise, *SEL* is de-asserted. An extension of the T0 code, called *dual_T0 code*, can be effective in these cases, providing the application of the T0 code and the updating of the encoding/decoding registers whenever *SEL* is asserted; otherwise, the binary code is applied and the encoding/decoding registers hold.
Analytically, the dual_T0 encoding can be expressed as:

$$(\mathrm{B}^{(t)}, \mathrm{INC}^{(t)}) = \begin{cases} (\mathrm{B}^{(t-1)}, 1) & \text{if SEL} = 1 \wedge \mathrm{b}^{(t)} = \tilde{\mathrm{b}}^{(t)} + \mathrm{S} \\ (\mathrm{b}^{(t)}, 0) & \text{otherwise} \end{cases} \tag{8}$$

where $\mathbf{B}^{(t)}$ is the value on the encoded bus lines at time t, $INC^{(t)}$ is the additional bus line, $\mathbf{b}^{(t)}$ is the address value at time t, $\mathbf{S}$ is the stride, *SEL* is the selection signal, and $\tilde{\mathbf{b}}^{(t)}$ is given by:

$$\tilde{\mathrm{b}}^{(t)} = \begin{cases} \tilde{\mathrm{b}}^{(t-1)} & \text{if SEL} = 0 \\ \mathrm{b}^{(t-1)} & \text{if SEL} = 1 \end{cases} \tag{9}$$

The corresponding decoding scheme follows:

$$\mathrm{b}^{(t)} = \begin{cases} (\tilde{\mathrm{b}}^{(t)} + \mathrm{S}) & \text{if INC} = 1 \\ \mathrm{B}^{(t)} & \text{if INC} = 0 \end{cases} \tag{10}$$

3.3 Dual_T0_BI Encoding

Similarly to what we have done for the T0_BI code, we can define the the *dual_T0_BI code*. With this scheme, which is expected to work well on multiplexed buses, the overall switching activity can be reduced by resorting the T0 code anytime stream α is transmitted on the bus, while the bus-invert is used for stream β. As for the T0_BI code, the dual_T0_BI can offer significant savings for the average power. The dual_T0_BI encoding is defined as:

$$(\mathrm{B}^{(t)}, \mathrm{INCV}^{(t)}) = \begin{cases} (\mathrm{B}^{(t-1)}, 1) & \text{if SEL} = 1 \wedge \mathrm{b}^{(t)} = \tilde{\mathrm{b}}^{(t)} + \mathrm{S} \\ (\bar{\mathrm{b}}^{(t)}, 1) & \text{if SEL} = 0 \wedge \mathrm{H}^{(t)} > \mathrm{N}/2 \\ (\mathrm{b}^{(t)}, 0) & \text{otherwise} \end{cases} \tag{11}$$

where $\mathbf{B}^{(t)}$ is the value on the encoded bus lines at time t, $INCV^{(t)}$ is the additional bus line, $\mathbf{b}^{(t)}$ is the address value at time t, $\mathbf{S}$ is the stride, *SEL* is the selection signal, $\mathbf{H}^{(t)} = (\mathbf{B}^{(t-1)}|INCV^{(t-1)}, \mathbf{b}^{(t)}|0)$ is the Hamming distance, N is the bus width of $\mathbf{b}^{(t)}$, and $\tilde{\mathbf{b}}^{(t)}$ is defined as before.
The decoding scheme can be summarized with the following equation:

$$\mathrm{b}^{(t)} = \begin{cases} (\tilde{\mathrm{b}}^{(t)} + \mathrm{S}) & \text{if INCV} = 1 \wedge \text{SEL} = 1 \\ \bar{\mathrm{B}}^{(t)} & \text{if INCV} = 1 \wedge \text{SEL} = 1 \\ \mathrm{B}^{(t)} & \text{if INCV} = 0 \end{cases} \tag{12}$$

3.4 Experimental Performance Comparison

In order to test out the performance of the newly proposed encoding techniques, we have used again the MIPS microprocessor as reference architecture on which measuring the number of address bus transitions required by the execution of the usual set of benchmark programs.
Tables 5 to 7 summarize the results, in terms of number of transitions and transitions savings with respect to the pure binary encoding.
When the percentage of in-sequence addresses is high, as in the case of instruction address streams (see Table 5), the savings provided by the T0_BI, the dual_T0 and the dual_T0_BI codes are very remarkable (35.52%, on average, with respect to pure binary). However, the same savings have been obtained by using the simple T0 code. Thus, the latter seems to be the most preferable solution in this case, due to the relatively low cost of the T0 encoding/decoding circuitry.
Regarding the data address streams, shown in Table 6, no savings are provided by the dual_T0 code, while the T0_BI and the dual_T0_BI offer some advantages with respect to binary encoding (12.82% and 10.66%, respectively). By comparing these results to the ones provided by the bus-invert method, we can claim that the T0_BI represents the most effective solution for average power minimization.
If we consider multiplexed address buses (see Table 7), the dual_T0_BI code shows the best savings (22.25% on average) if compared to the T0_BI and dual_T0 codes (19.56% and 12.15% over pure binary, respectively). Furthermore, the dual_T0_BI provides the ***absolute*** best savings, thus being the most effective code, concerning the average power, for the address bus of the MIPS microprocessor.

Benchmark	Stream Length	In-Seq Addr.	Binary Trans.	T0_BI		Dual_T0		Dual_T0_BI	
				Trans.	Savings	Trans.	Savings	Trans.	Savings
gzip	119102	60.16%	232587	140845	39.44%	140845	39.44%	140845	39.44%
gunzip	63884	64.94%	118409	84003	29.06%	81433	31.23%	81433	31.23%
ghostview	404595	57.27%	678295	481845	28.96%	470949	30.57%	470949	30.57%
espresso	1751673	60.17%	3014854	2286402	24.16%	2239400	25.72%	2239400	25.72%
nova	544994	52.99%	959912	838481	12.65%	836708	12.83%	836708	12.83%
jedi	14690249	53.96%	23145174	18521606	19.98%	18521606	19.98%	18521606	19.98%
latex	700317	71.42%	1400540	200490	85.68%	200490	85.68%	200490	85.68%
matlab	6400326	76.56%	12000527	5800465	51.66%	5800465	51.66%	5800465	51.66%
oracle	500326	69.98%	880547	680485	22.72%	680485	22.72%	680485	22.72%
Average		63.05%			34.92%		35.52%		35.52%

Table 5: Experimental Comparison of Mixed Encoding Schemes for Instruction Address Streams.

Benchmark	Stream Length	In-Seq Addr.	Binary Trans.	T0_BI		Dual_T0		Dual_T0_BI	
				Trans.	Savings	Trans.	Savings	Trans.	Savings
gzip	34393	52.47%	131651	83540	36.54%	131651	0.00%	118184	10.23%
gunzip	14602	10.59%	92349	82928	10.20%	92349	0.00%	81953	11.26%
ghostview	112689	10.74%	554499	500750	9.69%	554499	0.00%	507987	8.39%
espresso	570750	5.56%	3952944	3471146	12.27%	3956543	0.00%	3426699	13.39%
nova	220050	10.56%	1381975	1233051	10.77%	1381975	0.00%	1218763	11.81%
jedi	6145049	12.67%	47268786	42116937	10.90%	47268786	0.00%	41114510	13.02%
latex	200109	0.01%	763	577	24.38%	763	0.00%	558	26.87%
matlab	1204110	0.00%	5400375	5369926	0.56%	5400375	0.00%	5351262	0.91%
oracle	120111	0.01%	466606	466293	0.07%	466606	0.00%	466039	0.12%
Average		11.40%			12.82%		0.00%		10.66%

Table 6: Experimental Comparison of Mixed Encoding Schemes for Data Address Streams.

Benchmark	Stream Length	In-Seq Addr.	Binary Trans.	T0_BI		Dual_T0		Dual_T0_BI	
				Trans.	Savings	Trans.	Savings	Trans.	Savings
gzip	153495	57.44%	833799	715717	14.16%	736373	11.68%	709769	14.88%
gunzip	78486	52.81%	482093	356057	26.14%	426705	11.49%	343335	28.78%
ghostview	517284	58.25%	3067635	2661381	13.24%	2785341	9.20%	2785341	14.29%
espresso	2322423	54.39%	15147444	12349832	18.47%	13586402	10.31%	11883056	21.55%
nova	765045	56.63%	5526484	5218195	5.57%	5254072	4.92%	5054617	8.53%
jedi	20835298	57.88%	152427486	131617956	13.65%	141103112	7.43%	126132486	17.25%
latex	900426	55.55%	7002828	4202462	39.99%	6202718	11.43%	4002340	42.85%
matlab	7604436	65.37%	42405967	32257681	23.93%	34005845	19.81%	30775139	27.43%
oracle	620437	59.13%	3939551	3116685	20.89%	3479391	11.68%	2964637	24.75%
Average		57.62%			19.56%		12.15%		22.25%

Table 7: Experimental Comparison of Mixed Encoding Schemes for Multiplexed Address Streams.

4 Encoding and Decoding Logic

The results of Section 3.4 show that, for a muxed bus, the dual_T0_BI code is the most effective in terms of transition count. It is now necessary to evaluate if the power savings achievable through activity reduction is not offset by the circuitry required to implement the code on a system bus. After describing the the basic encoder/decoder architecture, we analyze their power consumption when the encoding scheme is used to minimize the power of on-chip and off-chip buses.

4.1 Architectures

The encoder architecture can be derived directly from Equation 11. It consists of a section for the T0 encoding which generates the INC signal, a section for the bus-invert logic providing the INV signal, and the output multiplexor, which is controlled by the input signal SEL and by signal $INCV = INC + INV$. While the architecture of the T0 section was fully described in [6], the bus-invert section has been realized by a Hamming distance evaluator of the encoded bus lines at time $t - 1$ concatenated with the $INCV$ signal and the address value at the present time t, followed by a majority voter to decide if the computed Hamming distance is greater than half of the bus width. Besides the encoded bus lines, the circuit outputs the $INCV$ signal (the SEL signal, needed by the decoder, is already present on the bus). The circuit has been synthesized using Synopsys Design Compiler, and implemented onto a $0.35\mu m$, 3.3 *Volt* library from SGS-Thomson. Its critical path is 5.36 *nsec*, and it is through the bus-invert section and the output mux.

Concerning the decoder, its architecture can be obtained easily from Equation 12, and it is reminiscent of the T0 decoder detailed in [6] (SEL and $INCV$ are the control signals of the multiplexer).

4.2 Power Analysis: On-Chip Busses

The analysis is carried out for three codes: Binary, T0, and dual_T0_BI. The binary encoder and decoder consist only of internal buffers, the T0 circuitry is the one presented in [6], and the dual_T0_BI encoding/decoding logic has the structure described in Section 4.1.

The same reference input switching activities (derived from the benchmark address streams) are applied to the inputs of the three encoders, and the power consumption is estimated using the probabilistic mode of Synopsys Design Power at a frequency rate of 100 MHz.

Concerning the decoders, the reference switching activities derived from the benchmark streams cannot be used to estimate the power of the T0 and the dual_T0_BI decoders; this is because the circuits receive, as input, the encoded address streams whose switching activities are reduced. Therefore, the switching activities used for the estimation are derived from such streams. Table 8 reports the total power consumption results. The power value of the dual_T0_BI encoder is approximately one order of magnitude worse than the one of the T0 encoder for on-chip loads up to 0.4 pF, while for higher values the difference is reduced. On the other hand, the power values of the decoders for the T0 and dual_T0_BI codes are comparable, due to the similarity in their architectures.

Load (pF)	Binary Enc/Dec (mW)	T0		Dual_T0_BI	
		Encoder (mW)	Decoder (mW)	Encoder (mW)	Decoder (mW)
0.01	0.1319	6.2891	3.1422	60.3139	3.6518
0.02	0.1660	6.3786	3.2043	60.4658	3.7238
0.04	0.2342	6.5579	3.3284	60.6372	3.8687
0.06	0.3023	6.7393	3.4525	60.8119	4.0176
0.08	0.3704	6.9219	3.5766	60.9895	4.1665
0.10	0.4381	7.1048	3.7088	61.1708	4.3163
0.16	0.6398	7.6538	4.1084	61.7393	4.7664
0.20	0.7742	8.0198	4.3772	62.1226	5.0682
0.30	1.1104	8.9376	5.0523	63.1054	5.8232
0.40	1.4465	9.8562	5.7274	64.1040	6.5785
0.50	1.7827	10.7749	6.4025	65.1125	7.3337
0.60	2.1188	11.6935	7.0776	66.1232	8.0889
0.70	2.4549	12.6122	7.7527	67.1388	8.8441
0.80	2.7911	13.5309	8.4278	68.1622	9.5993
0.90	3.1272	14.4496	9.1029	69.2003	10.3545
1.00	3.4633	15.3682	9.7780	70.2499	11.1098
2.00	6.8246	24.5550	16.5289	80.8256	18.6619
3.00	10.1859	33.7418	23.2798	91.5777	26.2141
4.00	13.5472	42.9286	30.0308	102.4174	33.7663
5.00	16.9085	52.1154	36.7817	113.2670	41.3184
10.00	33.7150	98.0493	70.5364	167.7545	79.0793

Table 8: Enc/Dec Power Consumption for On-Chip Loads.

4.3 Power Analysis: Off-Chip Busses

When a off-chip system bus is considered, it is necessary to introduce input and output pads at the chip interface. Pads usually represent the most power consuming part of the entire chip. The binary encoder is thus constituted only by the output pads driving typical output loads. This same structure must be added at the outputs of the T0 and dual_T0_BI encoders. Concerning the decoding circuitry, only the power dissipated by the logic is considered in the analysis, since the power consumption due to the input pads has shown to be negligible with respect to the one consumed by the output pads seen by the encoders.
The reference input switching activities are applied to the three encoders. In the binary encoder, the output pads commute at the reference switching activities. Conversely, the T0 and the dual_T0_BI encoders receive, as input, the reference switching activities, which are reduced at their outputs in the same manner as for the on-chip case. The encoders outputs drive the typical input capacitances of the output pads (0.01 *pF* for a 8 *mA* output pad). The reduced encoders output switching activities constitutes the switching activities applied to the output pads, which drive huge external loads. Therefore, the major power gains of the proposed codes derive from such reduction in the switching activities applied to the output pads.
The T0 and the dual_T0_BI decoders receive, as input, the reduced switching activities resulting at the corresponding output pads, and drive typical on-chip capacitances.
Power results are reported in Table 9. By looking at the data, we can see that the use of the T0 code is convenient for loads between 20 and 100 *pF*, while for larger values the use of the dual_T0_BI code is recommended.

5 Conclusions and Future Work

A comparative analysis of existing low-power bus encoding techniques, such as the T0 and the bus-invert codes, has allowed us to come up with mixed encoding schemes which exploit the best characteristics of the codes above. More specifically, we have proposed the T0_BI, the dual_T0, and the dual_T0_BI codes, and we have discussed their performance concerning the reduction in switching activity obtained on the multiplexed address bus (i.e., instruction and data addresses travel on the same bus) of the MIPS microprocessor when real programs are executed.

Load (pF)	Binary Global (mW)	T0		Dual_T0_BI	
		Pads (mW)	Global (mW)	Pads (mW)	Global (mW)
5	43.4593	38.3835	47.8148	31.9732	95.9389
10	60.1896	53.1597	62.5910	44.2817	108.2474
15	76.9198	67.9360	77.3673	56.5902	120.5559
20	93.6501	82.7123	92.1436	68.8988	132.8645
25	110.3804	97.4885	106.9198	81.2073	145.1730
30	127.1107	112.2648	121.6961	93.5158	157.4815
40	160.5713	141.8174	151.2487	118.1329	182.0986
50	194.0318	171.3699	180.8012	142.7500	206.7157
60	227.4924	200.9224	210.3537	167.3670	231.3327
70	260.9530	230.4750	239.9063	191.9841	255.9498
80	294.4135	260.0275	269.4588	216.6012	280.5669
90	327.8741	289.5801	299.0144	241.2182	305.1839
100	361.3347	319.1326	328.5639	265.8353	329.8010
110	394.9327	348.8066	358.2379	290.5535	354.5192
120	428.5308	378.4806	387.9119	315.2718	379.2375
130	462.1288	408.1546	417.5859	339.9900	403.9557
140	495.7270	437.8285	447.2598	364.7082	428.6739
150	529.3250	467.5026	476.9339	389.4265	453.3922

Table 9: Enc/Dec Power Consumption for Off-Chip Loads.

The dual_T0_BI has shown to be the most effective scheme, since it has produced a 22.25% savings over the pure binary encoding, while the T0 code — the best approach known so far — has only given a 10.25% switching activity reduction.
We have implemented the dual_T0_BI encoding/decoding logic, and we have compared its power performance to the ones of the binary and T0 encoders and decoders when on-chip and off-chip buses with different capacitive loads have to be driven.
Concerning future work, we are now looking into the problem of identifying the most appropriate encoding schemes for different types of memory hierarchies (e.g., main memory, L1 and L2 caches), address bus connections and I/O subsystems. As a first step in this direction, we are thus working on the characterization of existing microprocessors (e.g., MIPS, SPARC, PowerPC, DEC-Alpha, PA-RISC, Intel) with respect to these architectural options.

References

[1] P. R. Panda, N. D. Dutt, "Reducing Address Bus Transitions for Low Power Memory Mapping," *EDTC-96: IEEE European Design and test Conference*, pp. 63-67, Paris, France, March 1996.

[2] M. R. Stan, W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 3, No. 1, pp. 49-58, March 1995.

[3] J. L. Hennessy, D. A. Patterson, *Computer Architecture - A Quantitative Approach*, II Edition, Morgan Kaufmann Publishers, 1996

[4] C. L. Su, C. Y. Tsui, A. M. Despain, "Saving Power in the Control Path of Embedded Processors," *IEEE Design and Test of Computers*, Vol. 11, No. 4, pp. 24-30, Winter 1994.

[5] H. Mehta, R. M. Owens, M. J. Irwin, "Some Issues in Gray Code Addressing," *GLS-VLSI-96: IEEE 6th Great Lakes Symposium on VLSI*, pp. 178-180, Ames, IA, March 1996.

[6] L. Benini, G. De Micheli, E. Macii, D. Sciuto, C. Silvano, "Asymptotic Zero-Transition Activity Encoding for Address Busses in Low-Power Microprocessor-Based Systems," *GLS-VLSI-97: IEEE 7th Great Lakes Symposium on VLSI*, pp. 77-82, Urbana-Champaign, IL, March 1997.

[7] L. Benini, G. De Micheli, E. Macii, M. Poncino, S. Quer, "System-Level Power Optimization of Special Purpose Applications: The Beach Solution", *ISLPED-97: IEEE/ACM International Symposium on Low Power Electronics and Design*, pp. 24-29, Monterey, CA, August 1997.

[8] G. Kane and J. Heinrich, *MIPS RISC Architecture*, Prentice Hall, 1994.