Power Optimization of System-Level Address Buses based on Software Profiling

W. Fornaciari ‡ § M. Polentarutti § D. Sciuto ‡ § C. Silvano ‡ §

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

§ CEFRIEL
Milano, ITALY 20133

ABSTRACT

The paper aims at defining a methodology for the optimization of the switching power related to the processor-to memory communication on system-level buses. First, a methodology to profile the switching activity related to system-level buses has been defined, based on the tracing of benchmark programs running on the Sun SPARC V8 architecture. The bus traces have been analyzed to identify temporal correlations between consecutive patterns. Second, a framework has been set up for the design of high-performance encoder/decoder architectures to reduce the transition activity of the system-level buses. Novel bus encoding schemes have been proposed, whose performance has been compared with the most widely adopted power-oriented encodings. The experimental results have shown that the proposed encoding techniques provide an average reduction in transition activity up to 74.11% over binary encoding for instruction address streams. The results indicate the suitability of the proposed techniques for high-capacitance wide buses, for which the power saving due to the transition activity reduction is not offset by the extra power dissipation introduced in the system by the encoding/decoding logic.

1. INTRODUCTION

In microprocessor-based systems, significant power savings can be achieved through the reduction of the transition activity of the system buses. The power consumption due to the transition activity of the I/O pads in a VLSI circuit ranges from 10% to 80% of the overall power with a typical value of 50% for circuits optimized for low-power [1]. Several encoding techniques have been recently proposed in literature to reduce the switching activity of high capacitance bus lines. In [2], the authors have proposed a redundant encoding scheme, the *Bus-Invert* code, which is suitable to transmit patterns randomly distributed in time, such as for data buses. The approach introduces delay overhead due to the majority voter included in the encoder. Concerning address buses, for which sequential addressing usually dominates, the *Gray* code has been proposed in [4] and [5]. In ([6], [7]), a redundant encoding scheme, the *T0* code, has been introduced, that avoids the transfer of consecutive addresses on the bus by using a redundant line, *INC*, to transfer to the receiving sub-system the information on the sequentiality of the addresses. For infinite streams of consecutive addresses, the *T0 code* enjoys the property of zero transitions occurring on the bus, while the *Gray* addressing requires one bit switching per each pair of consecutive patterns.

Other encoding techniques at the system-level have been reviewed in [3], while a general encoding/decoding framework aiming at reducing the transition activity has been recently proposed in [1]. Although most of the known low-power encoding techniques can be implemented by using this framework, the critical path to transmit the information on the bus can have a significant impact on the system-level performance. Other approaches consist of directly changing the way the information is stored in memory, so that the address streams have already low transition activity [8].

In this scenario, we propose a design framework to simulate the application of low-power techniques to system-level buses in microprocessor-based architectures. The most relevant features of the proposed design methodology are:

- The target system architecture is quite general and models the HW/SW communication on system-level buses in terms of the main parameters that affect the switching power of the system: power supply, frequency, transition activity and capacitive load;
- The proposed encoding/decoding architecture implements different classes of bus encoding techniques and represents a timing optimization of the architecure proposed in [1]: the critical path delay has been minimized to reduce the latency of the bus accesses.
- The methodology enables the profiling of the software execution in terms of transition activity on system-level buses: real bus tracing, derived from the execution of several application programs can be analyzed;
- Correlation metrics have been defined to characterize the streams transmitted on the buses during HW/SW communication with the purpose of identifying the encoding technique which best fits to each target application.
- To further improve the transition activity on system-level buses, novel encoding techniques have been defined by extending the capabilities of previous low-power codes or by combining the previous ones. The proposed encoding techniques are suitable for address buses, characterized by high locality of references.

CODES 2000 San Diego CAUSA

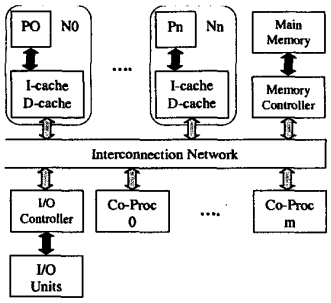

Figure 1: The target system architecture.

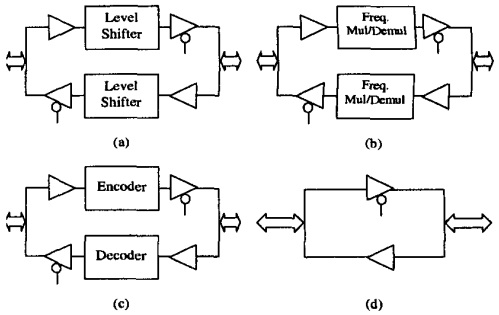

Figure 2: The power-oriented bus interfaces.

- Experiments have been carried out on real bus streams generated by tracing benchmark programs on the Sun SPARC V8 architecture. The results have shown an average reduction in transition activity up to 74.11% over binary encoding for instruction address streams.
- The implementation of the encoding/decoding architecture demonstrated how, for high-capacitance buses, the power saving due to the transition activity reduction is not offset by the power overhead introduced by the encoding/decoding logic.

The paper is organized as follows. In Section 2, we describe the target system architecture, the proposed high-performance encoding/decoding architecture, and the novel power-oriented bus encoding techniques. The methodology used to profile the software execution is described in Section 3 along with the correlation metrics. The experimental results in terms of both transition activity and power savings have been presented in Section 4. Finally, some concluding remarks and future development have been reported in Section 5.

2. TARGET SYSTEM ARCHITECTURE

A power-oriented methodology operating at the system-level is mandatory for the HW/SW architectural exploration during the first phases of the design flow. The methodology should be tightly related to the characteristics of the system architecture, mainly in terms of the target processors, the memory sub-system, the system-level buses and the co-processors.

Figure 1 shows the block diagram of our target system architecture, which is a shared memory multi-processor system that can be implemented by using either the System-On-a-Chip approach or the multi-chip approach. The system includes one or more processors, the instruction caches (I-caches), the data caches (D-caches), the memory controller, the main memory, the I/O controllers, the peripheral units, and the co-processors to support specific applications (such as MPEG encoding). All these basic blocks are connected through address, data and control buses implemented by using different topologies. Given the target architecture, our main focus is to investigate the HW/SW communication either on the sub-system-level buses, such as the processor-to-cache buses (which have been coloured in dark grey in Figure 1) or on the system-level buses. At these interfaces, we introduce a *Bus Interface* (BI) to model and optimize the four parameters which impact the switching power of the system: power supply, frequency, switching activity and capacitive load. In Figure 2, we propose four architectures to implement the BI module. These architectures model respectively voltage scaling *(a)*, frequency multiplier/demultiplier *(b)*, bus encoding/decoding to modify the transition activity of buses *(c)*, and bus buffering to decouple capacitive loads *(d)*.

2.1 Encoder/Decoder Architecture

A general framework for low-power bus encoding schemes has been recently proposed in [1]. The generic architecture can be specialized by using different alternatives for the internal decorrelating functions to derive most of the known low-power encoding techniques. However, the critical path to transmit the information on the bus can have a significant impact on the system-level performance. In fact, the critical path delay of the encoder is through the functions $\mathbf{f_1}$, $\mathbf{f_2}$, and *xor*, where $\mathbf{f_1}$ can implement either a *xor* or a *dbm* logic block, while $\mathbf{f_2}$ can implement the identity, *inv*, *vbm*, or *pbm* functions.

Starting from the architecure in [1], we propose an encoder/decoder (*Encdec*) which maintains wide generality while minimizing the critical path delay to reduce bus latency. The general encoder section of the Encdec is shown in Figure 3. The encoder receives as input $\mathbf{b}^{(t)}$, the information value at time t, and it generates $\mathbf{B}^{(t)}$, the value on the encoded bus lines at time t. It consists of registers for $\mathbf{b}^{(t-1)}$ and $\mathbf{B}^{(t-1)}$, and three combinational logic blocks:

- a *predictor* block P, that generates a prediction $\widehat{\mathbf{b}}^{(t)}$ of the current value of $\mathbf{b}^{(t)}$ based on the past value $\mathbf{b}^{(t-1)}$:

$$\widehat{\mathbf{b}}^{(t)} = P\,(b^{(t-1)}) \tag{1}$$

- a *decorrelator* block D, that decorrelates the input $\mathbf{b}^{(t)}$:

$$\mathbf{e}^{(t)} = D\,(b^{(t)}, \widehat{b}^{(t)}) \tag{2}$$

- a *selector* block S, that select among its inputs $\mathbf{b}^{(t)}$, $\mathbf{B}^{(t-1)}$, and $\mathbf{e}^{(t)}$.

The amount of hardware in the encoding functions has been kept as small as possible, and the critical path delay (through the D and S blocks) has been minimized to reduce the latency of bus accesses. A pass-gate dedicated implementation has been devised for both logic blocks on the critical path. As an example, for the S block, the *mux* function has been implemented by two pass-gates and one inverter, while the

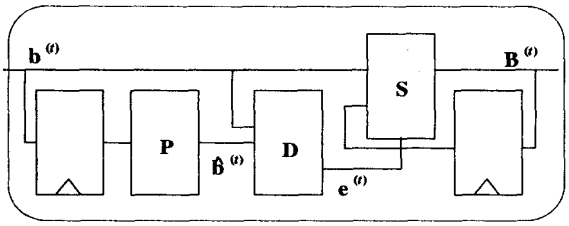

Figure 3: The encoder section of the proposed Encdec architecture.

Encoding	*P*	*D*	*S*	*Redundancy*
T0	Inc	Xor	Mux	Y
Bus-Invert	Id.	Xor	Inv.	Y
T0-Bus-Invert	Inc./Id.	Xor	Mux/Inv.	Y
T0-Xor	Inc.	Xor	Xor	N
Offset	Id.	Diff.	-	Y
Offset-Xor	Id.	Diff.	Xor	Y
T0-Offset	Inc./Id.	Xor/Diff.	Mux	Y
T0-Xor-Offset	Inc./Id.	Xor/Diff.	Xor	Y

Table 1: Mapping of different encoding schemes to the Encdec architecture. Note the registers on $B(t)$ are missing for Offset code.

xor function requires two pass-gates and two inverters. In both cases, the critical path through the S block is given by the propagation delay through one inverter and one pass-gate.

Table 1 summarizes the different implementations of the encoding functions P, D, and S corresponding to different code classes. The first three rows of Table 1 show codes already present in literature, while the following five rows represent newly derived codes.

2.2 Low-Power Encoding Techniques

In this section, we introduce novel encoding techniques to reduce the switching activity on high-performance wide buses, that can be implemented by using the proposed Encdec architecture. The codes have been derived by extending the capabilities of previous codes or by simply combining previous codes.

In the *T0-Xor code*, we extend the capabilities of the *T0 code* proposed in ([6], [7]) by combining it with a *xor* function:

$$\mathbf{B}^{(t)} = \begin{cases} [\mathbf{b}^{(t)} \oplus (\mathbf{b}^{(t-1)} + \mathbf{S})] \oplus \mathbf{B}^{(t-1)} & t > 0 \\ \mathbf{b}^{(t)} & t = 0 \end{cases} \quad (3)$$

where $\mathbf{B}^{(t)}$ is the value on the encoded bus lines at time t, $\mathbf{b}^{(t)}$ is the bus value at time t and S is the *Stride*. The presence of the decorrelating *xor* function enables us to avoid the introduction of the redundant line INC to the bus, necessary for the *T0 code*. Notice that the *T0-Xor code* can also be derived from the architecture proposed in [1], where it has been indicated as *Inc-Xor* code.

In the *Offset code*, we transmit on the bus the difference between the current value of $\mathbf{b}^{(t)}$ and the previous value $\mathbf{b}^{(t-1)}$. When the bus values transmitted on the buses are highly correlated, the value on the encoded bus lines $\mathbf{B}^{(t)}$ is reduced, and it is kept constant for in-sequence data.

In the *Offset-Xor code*, we first compute the difference ($\mathbf{b}^{(t)}$ - $\mathbf{b}^{(t-1)}$), then we execute a *xor* function to decorrelate the encoded bus lines:

$$\mathbf{B}^{(t)} = \begin{cases} (\mathbf{b}^{(t)} - \mathbf{b}^{(t-1)}) \oplus \mathbf{B}^{(t-1)} & t > 0 \\ \mathbf{b}^{(t)} & t = 0 \end{cases} \quad (4)$$

In the *T0-Offset code*, we extend the capabilities of the *T0 code* by adopting the *T0* scheme for in-sequence bus values, while for the out-of-sequence bus values we use the *Offset code*, since the encoding of the difference ($\mathbf{b}^{(t)}$ - $\mathbf{b}^{(t-1)}$) could imply less transitions on the bus lines with respect to the binary encoding. The *T0-Xor-Offset code* can be derived by combining the *T0-Xor* scheme for in-sequence bus values, while for out-of-sequence bus values we adopt the *Offset* code. In the *T0* code with variable stride, namely *T0-Var code*, the *Stride* between consecutive patterns can be parametric. To represent the most frequent distances occurring between consecutive addresses, we use n values of the *Stride* $\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_n$, so we need $log_2(\mathbf{S}_n)$ redundant lines. In the reduced Bus Invert code, namely *Red-BI code*, we exploit the fact that the most significant bits of the system bus present a less significant transition activity with respect to the least significant bits. Thus we reduce the threshold, after which we invert the bus value, to a number less than $N/2$.

3. SOFTWARE EXECUTION PROFILING

A software tool, $\mathcal{MAYA}$, has been developed to trace the transition activity of system-level buses during the execution of benchmark programs, to analyze the bus traces in terms of correlation metrics, and to implement bus encoding techniques. The tracing and analysis capabilities of the tool have been maintained independent to each other, to allow the acquisition of bus sequences obtained by different tracing programs. In the current version of $\mathcal{MAYA}$, we integrate the Shade-Spix tool [9], which combines an efficient instruction-set simulator with a flexible tracer of the execution of application programs. The current version of Shade runs on Sun SPARC systems and it simulates the SPARC (Version 8 and 9) and MIPS I instructions sets. The results reported in this paper have been carried out on the Sun SPARC V8 architecture.

The structure of the $\mathcal{MAYA}$ tool is mainly composed of the internal tracer, the external tracer and the analyzer. The internal and external tracers enable two different data acquisition modes. The internal tracer is based on the Shade tool to profile data streams derived from the execution of real application programs on the target architecture, while the external tracer receives as input a user-defined external sequence with given characteristics in terms of data correlation. Due to the length of the information streams to be processed (in the order of millions of patterns), the internal tracer provides a single pattern at a time to the analyzer. The analyzer evaluates *on-the-fly* the correlation metrics, implements the encoding techniques, and calculates the bus transition activity.

3.1 Correlation Metrics

The correlation metrics we introduce analyze the effects of the principle of locality on the bus streams derived by the software profiling. Being $\mathbf{B}^{(t)}$ the value on the bus lines at time t, and S the *Stride*, the consecutive addresses satisfy the condition:

$$\mathbf{B}^{(t)} = \mathbf{B}^{(t-1)} + \mathbf{S} \quad (5)$$

Given a stream composed of $N + 1$ values of bus lines: $\{\mathbf{B}^{(0)}, \mathbf{B}^{(1)}, \cdots, \mathbf{B}^{(N)}\}$, we define the metric P_{seq} as the

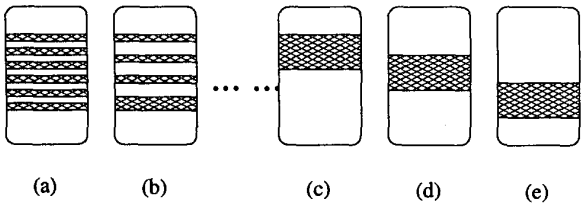

Figure 4: Given P^*_{seq}, the consecutive addresses in the stream can be distributed in many ways.

number of in-sequence addresses over N:

$$P_{seq} = \frac{\sum_{t=1}^{N} \delta_{\mathbf{B}^{(t)},\mathbf{B}^{(t-1)}+\mathbf{S}}}{N} \quad (6)$$

where δ is the function delta of Kroeneker. Note that the value of P_{seq} (where $0 \leq P_{seq} \leq 1$) multiplied by 100 represents the percentage of in-sequence addresses. For a given value P^*_{seq}, the consecutive addresses in the stream can be distributed in many ways, as shown in Figure 4. The extreme cases are: *(i)* the in-sequence addresses are grouped in P^*_{seq} sets of two consecutive values; *(ii)* the in-sequence addresses are grouped in a single set of (N P^*_{seq}) consecutive addresses. We define the ζ metric to take into consideration the distribution of consecutive address in the stream:

$$\zeta = \frac{\sum_{t=2}^{N} \delta_{\mathbf{B}^{(t-1)},\mathbf{B}^{(t-2)}+\mathbf{S}}(1 - \delta_{\mathbf{B}^{(t)},\mathbf{B}^{(t-1)}+\mathbf{S}})}{N} \quad (7)$$

In practice, the value of the ζ metric is incremented by 1 each time we exit from a sequence of consecutive addresses, thus when the condition C is satisfied:

$$\mathbf{C} = \begin{cases} \mathbf{B}^{(t)} \neq (\mathbf{B}^{(t-1)} + \mathbf{S}) \\ \mathbf{B}^{(t-1)} = (\mathbf{B}^{(t-2)} + \mathbf{S}) \end{cases} \quad (8)$$

The values of ζ can vary from $\frac{1}{N}$ (case *i*) to P^*_{seq} (case *ii*). Given P^*_{seq}, the cases (a), (b) and (c) of Figure 4 provide different values for ζ, while the cases (c), (d), and (e) provides the same values for ζ. The L metric (where $1 \leq L \leq (NP^*_{seq})$) represents the average length of the in-sequence sets:

$$L = \frac{P_{seq}}{\zeta} \quad (9)$$

4. EXPERIMENTAL RESULTS

Aim of this section is to evaluate and to compare the performance of the proposed codes in terms of bus switching activity and power dissipation. The set of benchmark programs used to compare the bus encoding techniques has been summarized in Table 2, as well as the main characteristics of the selected streams. For each benchmark program, we report the stream length, the percentage of in-sequence addresses, ζ, and L for both data and instruction address streams.

4.1 Transition Activity Results

To evaluate the effectiveness of the proposed encodings, a comparison between their transition activity and the transition activity of other encoding schemes in literature, with respect to the transition activity of the binary encoding has been performed. The results have been derived for both the instruction address streams and the data address streams.

Globally, none of the analyzed codes is suitable for both instruction and data address streams due to the peculiar characteristics of the streams. The comparison results show that, the *T0-Xor* and *Red-BI* codes are the most effective in terms of transitions count for instruction and data address streams, respectively.

Table 3 reports the percentage of transition savings with respect to binary encoding for instruction address streams (in particular, the last row shows the percentage saving of each code averaged over the whole set of benchmarks). Being high the percentage of in-sequence addresses, the savings provided by the *T0*-based codes as well as the *Offset*-based codes are remarkable with respect to binary. On the contrary, the *BI*-based codes do not provide any advantage over binary. Better results are given by the irredundant *T0-Xor* code (74.11%), which outperforms both the other irredundant schemes (*Offset* and *Offset-Xor*) and the redundant schemes (*T0*, *T0-BI*, *T0-Offset*, and *T0-Var*). The irredundant *Offset* and *Offset-Xor* codes present significant advantages (56.39% and 41.35%, respectively). The redundant *T0*-based codes offer advantages in the order of 60% versus the binary code. Among them, the results are in favour of the *T0-Var* code (64.18% saving), however the simple *T0* code (providing 61.64% saving) represents the most suitable solution, due to the reduced cost in terms of redundancy and encoder/decoder logic.

Among the set of benchmark programs, the instruction address streams corresponding to `gzip` and `quicksort` present the highest values of the P_{seq} and L correlation metrics (see Table 2). If we consider each column of Table 3 corresponding to *T0*-based and *Offset*-based codes (which better exploit address locality), we can observe how the percentage transition savings corresponding to `gzip` and `quicksort` are higher than the average value over the benchmark set, thus proving the effectiveness of the proposed metrics.

As expected, the *T0*-based codes, as well as the *Offset*-based codes, do not show significant transition savings (at most 3.6%) for data address streams (the complete results are not reported here for space reason). The redundant *BI*-based codes are the most suitable for data address buses. Among them, the *Red-BI* and *T0-BI* encodings (which respectively provide 15.55% and 10.05% of saving) outperform the simple *BI* encoding (9.6%).

4.2 Power Dissipation Results

Aim of this paragraph is to evaluate whether the power savings achieved through switching activity reduction is offset by the circuitry required to implement the encoding/decoding functions. We analyzed the power consumption results when the encoding scheme has been used for both *on*-chip and *off*-chip buses. The proposed high-performance Encdec architecture has been used to implement the different codes by using the 0.35 μ m and 3.3 V library supplied by ST Microelectronics. Power estimates have been obtained by using Synopsys Design Power at the maximum clock frequency achieved for each code implementation.

In this paper, we report the power consumption results for different values of *off*-chip capacitances for the following codes: *Binary*, *T0*, *Offset*, and *T0-Xor*. Two different implementations of the binary encoder have been considered: the first one includes registers and pads at the interface, while the second one includes pads only. The power consumption results have been reported in Figure 5 for different values of the bus capacitances (a single set of data has been plotted

Benchmark	*Comment*	*Data Addresses*				*Instruction Addresses*			
		Length	P_{seq}	ζ	L	*Length*	P_{seq}	ζ	L
`quicksort`	Sorting of 4096 integers in the range 0-255	1547779	10.87%	0.09	1.21	4138810	90.26%	0.10	9.03
`date`	OS call providing date and time	189702	3.37%	0.03	1.12	820102	84.28%	0.16	5.27
`gcc`	gcc quicksort.c -o quicksort	236544	3.50%	0.02	1.75	847654	83.48%	0.17	4.91
`gzip`	gzip maya.c ($MAYA$ code)	3417970	3.72%	0.03	1.24	10235521	91.63%	0.08	11.45
`go`	go 19 5 (19×19 chess-board with 5 difficulty level)	219026182	1.03%	0.01	1.03	957881788	89.46%	0.11	8.13
`hello`	Print 'hello world'	144898	2.72%	0.02	1.36	556686	84.82%	0.15	5.65
`latex`	latex comments.tex ($MAYA$ code documentation)	27762105	4.05%	0.04	1.01	108026949	87.05%	0.13	6.70
`vi`	Editing of maya.c	967410	1.42%	0.01	1.42	4844023	82.13%	0.18	4.56
`ghostview`	ghostview comments.ps	2857389	4.93%	0.03	1.64	12437226	83.95%	0.16	5.25
`xfig`	Graphical editor	5256745	4.13%	0.03	1.38	22028996	83.68%	0.16	5.23
`uncompress`	uncompress maya.c.Z	750585	7.74%	0.08	0.97	2704119	87.36%	0.13	6.72

Table 2: Description of the benchmark set

Benchmark	T0	Bus-Invert	T0-BI	T0-Xor	Offset	Offset-Xor	T0-Offset	T0-Var	Red-BI
`quicksort`	67.3	0.0	67.3	83.1	73.1	45.8	75.2	72.1	0.0
`date`	58.3	0.0	55.9	71.2	51.7	40.3	59.2	60.5	0.0
`gcc`	54.9	0.1	53.7	68.9	46.2	38.8	55.5	56.5	0.4
`gzip`	74.5	0.0	74.4	81.9	76.7	47.5	79.0	77.7	0.0
`go`	68.1	0.0	64.9	78.6	63.5	43.0	67.7	69.5	0.1
`hello`	59.1	0.0	59.1	72.2	53.8	41.0	60.5	60.9	0.0
`latex`	60.5	0.3	56.0	73.3	53.7	40.6	60.0	62.1	0.7
`vi`	52.7	0.2	51.2	65.0	39.6	39.6	51.0	54.0	0.6
`ghost`	57.8	0.0	56.4	71.6	50.2	39.0	57.4	62.0	0.0
`xfig`	59.5	0.0	56.6	70.4	47.5	36.4	58.8	61.5	0.2
`uncompress`	65.3	0.0	65.3	79.0	64.3	42.9	64.3	69.2	0.0
Average	61.64	0.05	60.07	74.11	56.39	41.35	62.60	64.18	0.18

Table 3: Instruction address streams: Percentage transition savings with respect to binary encoding for the benchmark set.

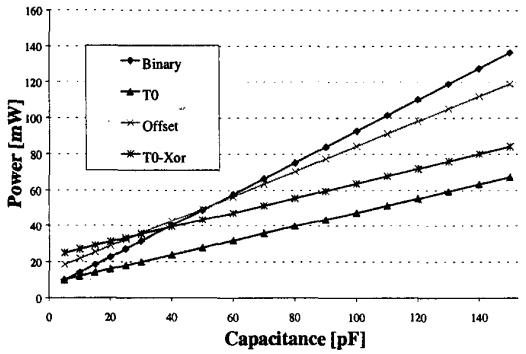

Figure 5: Power dissipation vs. off-chip capacitances for several encoding techniques.

for binary encoders). For each code scheme, a minimum capacitance value $\overline{C}$ exists for which the corresponding coding scheme results convenient with respect to binary encoding. Table 4 summarizes the $\overline{C}$ values for the *T0*, *Offset*, and *T0-Xor* schemes with respect to both binary encoders.

$\overline{C}$	$T0$	$Offset$	$T0-Xor$
Binary (regs)	10 pF	60 pF	40 pF
Binary (no regs)	10 pF	70 pF	50 pF

Table 4: Minimum capacitances $\overline{C}$ for which the selected codes consume less power than binary encoders.

5. FUTURE WORK

As future evolution of the work, we are devising a methodology to evaluate the benefits of encoding schemes on the power consumption of system-level buses when the target system architecture includes multi-level cache memories and different bus topologies. The cache model considers any cache configuration in terms of size, associativity, and cache line size. Moreover, we are investigating system architectures based on *VLIW ASIP* processor cores. The system-level framework can be effectively adopted to appropriately configure the memory sub-system and system bus architecture from the power standpoint.

6. REFERENCES

[1] S. Ramprasad, N. R. Shanbhag, I. N. Hajj, "A Coding Framework for Low-Power Address and Data Busses," *IEEE Trans. On Very Large Scale Integration (VLSI) Systems*, Vol. 7, No. 2, June 1999, pp. 212-221.

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

[3] M. R. Stan, W. P. Burleson, "Low-Power Encodings for Global Communication in CMOS VLSI," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 5, No. 4, pp. 444-455, Dec. 1997.

[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, IL, March 1997.

[7] L. Benini, G. De Micheli, E. Macii, D. Sciuto, C. Silvano, "Address Bus Encoding Techniques for System-Level Power Optimization," *DATE-98*, pp. 861-866, Feb. 1998.

[8] P. R. Panda, N. D. Dutt, " Low-Power Memory Mapping Through Reducing Address Bus Activity," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 7, No. 3, pp. 309-320, Sep. 1999.

[9] B. Cmelik, D. Keppel, " Shade: A Fast Instruction-Set Simulator for Execution Profiling," *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, 1994.