ADAPTIVE BUS ENCODING SCHEMES FOR POWER-EFFICIENT DATA TRANSFER IN DSM ENVIRONMENTS*

Claudia Kretzschmar, Markus Scheithauer, and Dietmar Mueller
Dpt. of Systems and Circuit Design
Chemnitz University of Technology, Germany
clkre@Infotech.tu-chemnitz.de

Abstract: Power dissipated on global DSM buses increasingly influences total chip power consumption. In order to reduce that portion, activity minimizing bus encoding schemes are applied. Thereby adaptive techniques have a higher potential in reducing transitions than static schemes. However, they are susceptible to rarely but occuring transmission errors since the encoding rule on decoder side is calculated from the received data. Once a different encoding rule is calculated there is no re-synchronization. In this paper we present a new approach which eliminates the dependency on the received data. The encoding rule of both, coder and decoder is updated by partial runtime reconfiguration. In combination with a static scheme we could achieve a reduction in activity of up to 26%. In comparism with an adaptive scheme the hardware requirements were reduced by up to 50%.

Keywords: Adaptive bus encoding, DSM, Partial run-time reconfiguration

1. INTRODUCTION

Technology scaling into the deep sub micron range (DSM) allows the realization of embedded systems on a single chip. Due to the extended functionality and the higher integration density both, power dissipation and power distribution over the whole chip are becoming limiting factors [Sylvester and Keutzer, 1998]. Since the reliability of a circuitry can be guaranteed only until a certain thermal threshold the reduction of power dissipation becomes increasingly important.

Embedded systems realized in DSM technologies are proposed to be implemented in a tile-like structure since DSM effects can be almost neglected in

*This work is funded by the Deutsche Forschungsgemeinschaft DFG within the VIVA research initiative under MU 1024/5-4 modules with 50 to 100 k gates [Sylvester and Keutzer, 2001]. They communicate with each other over high capacitive global system buses or extended networks on chip (NoC) [Benini and DeMicheli, 2002] which are dominated by parasitic DSM effects such as capacitive and inductive coupling. As a result the communication becomes error-prone and lossy. The signal integrity on these communication channels can be improved by repeaters which additionally increase the power dissipated on global wires [Sylvester and Keutzer, 2001]. Therefore wires contribute a main and even increasing portion to the overall power dissipation of embedded systems. The power consumed on system buses can be calculated by the well-known formula: $P = \frac{1}{2}V_{dd}^2 f \sum_{i=1}^{n-1} C_{L_i}\alpha_i$ where V_{dd} is the supply voltage, f the frequency, C_i the capacitance and α_i the switching activity of line i, respectively. Although modern technologies allow a high flexibility of the used supply voltage such as voltage islands, both V_{dd} and frequency are determined by performance requirements. The wire capacitance, depends on the distance of the modules and the layout. Only the switching activity can directly be influenced by the designer.

One approach is the application of transition-minimizing bus encoding schemes which either work statically or adaptively. Due to their reactivity to varied statistics adaptive schemes usually outperform static schemes with respect to the reduction in activity. The required adaption of the encoding rule is concurrently performed at coder and decoder based on the uncoded data. Therefore adaptive techniques rely on error-free transmission. However, errors occur on DSM transmission channels which can result in a different, non re-synchronizable decoding rule. In this paper we present a new approach which eliminates the susceptibility to errors. It exploits a new trend in complex ASICs: embedded FPGA (eFPGA) which are tailored on specific functional requirements in order to provide more flexibility [IBM, 2003]. Implemented in eFPGAs, partial dynamic reconfiguration allows the replacement of coder and decoder with a more suited one during operation.

The remainder of the paper is structured as follows. Section 2 gives an overview of the related work. The fundamentals of the proposed scheme are presented in sect. 3. In Sect. 4 the hardware platform and the realization are described while in Sect. 5 experimental results are presented. The paper will be summarized in Sect. 6.

2. RELATED WORK

In the literature a huge number of bus encoding techniques were published. Applying such methods transitions are displaced from high capacitive bus lines into the less-capacitive encoder-decoder (codec) circuitry under exploitation of different characteristics of the data stream.

Depending on the way the encoding rule is implemented into the codec system encoding techniques are either *static* or *adaptive.* The encoding rule of *static* schemes has to be optimized at design time for an application-specific data stream. Examples tailored on the high in-sequence portion of address buses are Gray encoding [X.L.Su et al., 1994] and combined schemes such as T0-BI, Dual-T0 and Dual-T0-BI [Benini et al., 1998]. Instruction- or data bus streams have a different characteristics and require therefore other techniques such as the 1bit redundant Businvert encoding (BI) [Stan and Burleson, 1995], which is the only static scheme that does not require a priori knowledge of statistical parameters of the data stream or the Partial Businvert encoding scheme (PBI) [Shin et al., 1998] encoding only a sub set of bus lines with high activity. A third class of static schemes focuses on the minimization of activity on adjacent lines. Coupling activity contributes due to the higher coupling capacitances in DSM technologies mainly to overall power dissipation. Example schemes are the coupling-driven Businvert [Kim et al., 2000] and the Odd/Even Businvert scheme [Zhang et al., 2002].

Most of the static schemes mentioned so far achieve a high encoding efficiency if the data transmitted corresponds to the streams they are optimized for. If the statistical parameters vary over time as this is usually the case for *real* applications, the encoding efficiency decreases dramatically as shown in [Kretzschmar et al., 2003]. In contrast to that adaptive schemes such as presented in [Benini et al., 1999], the Adaptive Code-book Method [Satoshi Komatsu et al., 2000] and the Adaptive Partial Businvert (APBI) [Kretzschmar et al., 2000] which adapts the set of bus lines to encode, outperform static schemes in terms of encoding efficiency due to the adaption of the encoding rule. In the last few years the number of adaptive schemes increased. Only a few can be mentioned here: the Frequent Value encoding (FV) [Yang and Gupta, 2001], the Adaptive Encoding Scheme ADES [Lv et al., 2002] and the Adaptive Probability Based Mapping (APBM) [Kretzschmar et al., 2003] for data buses and the Address Encoding using Self-Organizing Lists [Mamidipaka et al., 2001] and ETAM++ [Lekatsas and Henkel, 2002] for address buses.

Starting from the identical initial encoding rule, coder and decoder of adaptive schemes concurrently perform an update based on the uncoded or decoded data, respectively. As long as the transmission is error-free both compute the same encoding rule. However, wires in DSM technologies are susceptible to transmission errors which can result in a different encoding rules. Therefore the encoding rule has to be computed at a central place from which it is transferred to coder and decoder. If being transmitted over the bus error correction is compulsory. Furthermore activity is generated and additional bandwidth or wires are required which is infeasible in complex circuits. Therefore we propose the utilization of embedded FPGAs (eFPGA) as presented by XILINX Inc. and IBM Corp. [IBM, 2003]. eFPGAs will be included in complex circuits in order to introduce more flexibility in complex ASICs. The areas are tailored to the needs of the function to be realized. Implementing the codec system on eFPGAs provides the adaptability of the encoding scheme while simultaneously eliminating the dependence on transmission errors. Additionally our approach reduces the overall hardware overhead.

3. FUNDAMENTALS OF THE SCHEME

The principle of the proposed encoding using eFPGAs is shown in Fig. 1. In the depicted example circuit the 3 modules X, Y and Z are connected via a transition minimizing codec system to the bus. The data sent by the source module X is transmitted in activity reduced fashion over the bus. Each of the receiving modules Y and Z is connected to a decoder which recovers the original data. Coder and decoder circuitry are implemented as partial runtime reconfigurable (pRTR) modules into the eFPGAs. At design time a static encoding scheme is selected based on the statistics of an application-specific data stream. The corresponding codec macros are loaded at setup time into the eFPGAs.

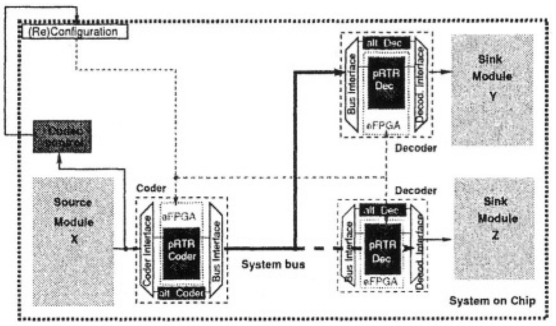

Figure 1. Principle of runtime exchange of encoding techniques using eFPGA

Since the statistics of real-life data streams usually changes over time the encoding efficiency of the selected static technique decreases. Therefore the scheme has to be adapted to varied statistical parameters during operation. The implemented codec control block observes relevant parameters in a window of fixed size and periodically selects a suited scheme. The corresponding coder and decoder(s) are loaded into the eFPGAs by runtime reconfiguration. In contrast to traditional adaptive schemes which concurrently observe the statistics in coder and decoder, our proposed technique implements a single observation block within the circuit which is sufficient even if more than one receiver is connected to the source. Thereby the hardware overhead is significantly reduced in comparism with published adaptive schemes. However, the most important advantage is the decoupling of the decoding rule from the received data. Possible transmission errors will not impact the computation of the decoding rule anymore.

While being reconfigured data can not be processed by the codec system. The required alternative data path for the reconfiguration period can be realized either by wires bypassing the pRTR codec or a simple encoding scheme. Simply bypassing the pRTR codec requires very little hardware overhead. On the other hand the encoding efficiency deteriorates dependent on the reconfiguration duration. In contrast to that providing a different codec system introduces additional hardware overhead which results in a higher total area but improves the overall encoding efficiency. The total power dissipation can be kept low by switching off the alternative codec during periods of "normal" operation. The tradeoff between cost and gain of the two possibilities determines which alternative to use.

For the realization of the proposed adaptive scheme on eFPGAs two different possibilities of adaptation exist. The two variations which differ in their observation and reconfiguration complexity, will be discussed in the following.

3.1 Dynamic exchange of the encoding scheme

This variation is based on experimental results which show that the encoding efficiency of different encoding techniques distinguishes if applied to a number of data streams. In order to achieve the maximum encoding efficiency always the most suited encoding scheme has to be selected for the current data. The encoding scheme is exchanged by replacing it with the new, more efficient one during operation as depicted in Fig. 2. If for instance a data stream of an i.i.d. source is transmitted BI is optimal [Stan and Burleson, 1995] while for other streams PBI or PBM might achieve a more efficient reduction in activity. Since the schemes exploit different characteristics of the data streams

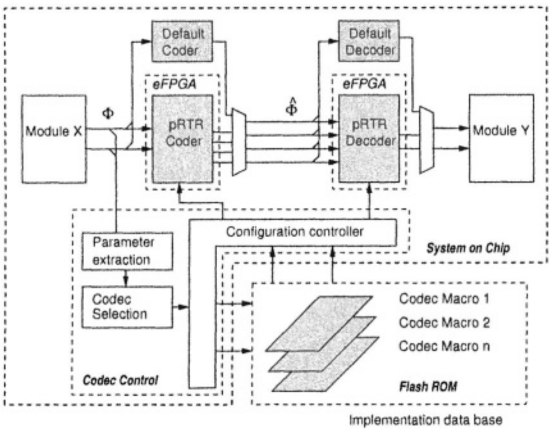

Figure 2. Dynamic replacement of encoding schemes using eFPGA

the codec control block has to observe the parameter set for each of the considered techniques such as line transition activity or data word probability. In a very complex process the codec control block has periodically to evaluate the results of each observed parameter set in order to determine which scheme would achieve the highest encoding efficiency for the current data stream characteristics. After initiating the reconfiguration and switching the data transmission to the alternative data path the configuration controller loads the selected codec macro from the flash ROM into the eFPGA. After the reconfiguration the transmission is switched back to the pRTR codec system by the codec control block.

While this approach allows the highest flexibility with respect to the encoding scheme to be selected and the best adaption to the current data statistics it requires a high effort. The number of schemes to consider is restricted by the size of the flash ROM and the number of parameter sets to be observed and evaluated in the codec control block since it directly influences the area and power requirements. The most decisive drawback of this variation is that a large eFPGA is required in order to provide enough area for each coder and decoder, respectively of the considered static schemes. The size of the eFPGA determines the cost of the circuit as well as the reconfiguration duration. Since during the reconfiguration data is sent either non- or less effective encoded the overall encoding efficiency is directly influenced.

Due to the previously mentioned drawbacks we propose a simplified version of the partial runtime reconfiguration which is described in the next section.

3.2 Dynamic adaption of the encoding rule

Figure 3 shows how dynamic reconfiguration is used in the simplified variation for the adaption of encoding schemes in integrated DSM circuits. In contrast to the previously described method only the encoding rule of a static scheme is optimized instead of exchanging the encoding technique. That approach allows the optimization of static schemes such as PBI or PBM for the current data stream characteristic similar to APBI or APBM without the susceptibility to transmission errors. Due to the adaptability a high encoding efficiency can be achieved if the statistics varies while simultaneously alleviating some of the drawbacks of the previous variation. In comparism with the dynamic exchange of the encoding scheme the eFPGA area is reduced drastically since only that part of the codec system which implements the encoding rule has to be updated. The rest of the codec system does not change and can therefore be fabricated in conventional ASIC technology which reduces the chip cost while providing enough flexibility to adapt the scheme. The smaller eFPGA area results in a reduced reconfiguration duration which requires less data words to be encoded by the default codec during the reconfiguration period. Therefore the influence of the default encoding method is decreased which increases the overall performance especially if data is not encoded alternatively.

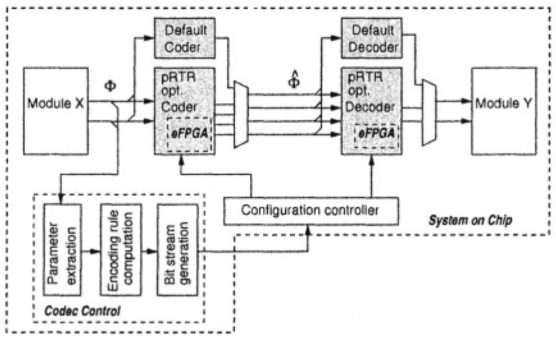

Figure 3. Dynamic adaption of the encoding rule using eFPGA

Also the codec control block is significantly simplified since for the adaption of one scheme only a single parameter set has to be observed.

In order to eliminate the flash ROM which is used to store all possible codec macros we propose to expand the codec control block by a bit-stream generation block. The updated bit-stream containing the new encoding rule is computed and directed to the configuration pins of the eFPGA. This approach allows the self-reconfiguration of the circuit.

In order to keep the hardware overhead and the power dissipation of the codec control block as low as possible we decided to implement the second approach which realizes the adaption of the encoding rule instead of the exchange of the complete encoding technique. As a representative for all possible static schemes we chose PBI and periodically adapted the set of bus lines to include into encoding. We refer to it as PBI-Reconf.

4. TEST PLATFORM

4.1 Hardware

We implemented the proposed partial run time reconfigurable encoding scheme on a XILINX VIRTEX FPGA XCV1000, since we did not have on our disposal integrated circuits containing eFPGAs. XILINX VIRTEX FPGAs consist of configurable logic blocks (CLB) which are arranged in columns and rows. Each CLB contains 2 Slices with 2 logic cells (LC) each. A LC is formed by a flip flop (FF) and a 4-input look up table (LUT) which are used to realize the logic function. In order to (re)configure the FPGA according to the required function a bit-stream is generated from a RTL description and loaded via the configuration pins into the FPGA. Thereby the smallest reconfigurable unit is a frame which spreads over all rows of a certain column.

Due to the bit-stream structure all rows of the concerned column are involved in a reconfiguration. Therefore the modules have to be placed carefully so that a partial run time reconfiguration does not affect logic of other modules.

In [Haase et al., 2002] the authors developed a method for circuit partitioning together with an automated design flow based on XILINX standard tools in order to produce the partial bit streams used to dynamically reconfigure the FPGA during runtime. The exchange of the complete encoding scheme would require synthesis, place and route for each of the considered codecs. Since we restrict ourselves to the adaption of the encoding rule only a single bit-stream has to be generated, from which the part is extracted that contains the encoding rule for the codec system. That partial bit-stream is taken as a basis to periodically generate the updated bit-stream.

4.2 Implementation

PBI can be adapted to the statistics of the current data stream if the sub bus to be encoded is periodically selected. Included lines are transmitted inverted if more than a half of the considered lines would switch. It means that transitions are only detected for selected bus lines and the value to which the number of switching lines is compared has to be updated for each configuration. Furthermore only lines which are included into encoding are inverted while non-included lines are transmitted uncoded independent of the number of transitions. Corresponding lines have to be masked. On decoder side the inversion is selectively applied according to a mask. For the PBI-Reconf coder we placed the 2 required LUTs per bus line in the reconfigurable area in order to influence the functionality as described. According to the configured function of the LUT the bits are correspondingly processed. The inversion threshold is formed by the outputs of $log(\frac{buswidth+1}{2})$ LUTs. The decoder implements one reconfigurable LUT per line which does the selective inversion. For a 32 bit bus it results in 69 reconfigurable LUTs for coder and 32 for decoder. A column in the XCV1000 contains 64 rows which corresponds to 128 LUTs per slice. The reconfiguration of the experimental environment can not be restricted to the utilized LUTs. Instead all LUTs of the corresponding slice of the column are covered. In contrast to that only used resources are reconfigured in the eFPGA. In order to converge the reconfiguration time of the two approaches all the reconfigurable LUTs of coder and decoder were placed in a single column of the VIRTEX FPGA.

The codec control block consists according to Fig. 3 of a parameter observation block, a mask computation logic and a bit-stream generation unit. The first two blocks correspond to the mask computation block of APBI but are required only once per PBI-Reconf circuit. The bit-stream generation unit alters the LUT bits of the partial bit-stream according to the new mask. Each of the 16 bits of a LUT is situated in a different frame. Therefore 16 Frames with 39 words have to be reconfigured. Since the LUT bits are placed very regularly in the bit-stream a few counters suffice to find their positions. According to the new subset of bus lines the bit-stream is altered and routed to the configuration pins of the FPGA. The codec control block switches data transmission to the alternative data path during the reconfiguration and back to the adapted PBI-Reconf codec system afterwards.

5. EXPERIMENTAL RESULTS

We implemented the proposed PBI-Reconf encoding for a bus width of 32 lines in the FPGA test environment and conducted experiments with the following set of test data streams:

- **art:** A random, segmented data stream with a varying distribution of activity in each segment.
- **eps:** An ASCII file in encapsulated postscript format.
- **gzip:** Gzip binary (example for an executable file).
- **ppm:** A composed PPM image stream consisting of 4 different images with varying statistics.
- **gauss:** White Gaussian noise.

The statistical parameters were observed in a window of fixed size. Subsequently a new mask was computed. For the experiments we first read the partial configuration bit-stream using the reconfiguration pins of the FPGA before it was altered according to the new mask and reloaded. The reconfiguration was done in the SelectMap mode with a reconfiguration clock of 45 MHz. During the reconfiguration period continuously data was transmitted either with no encoding (PBI+NE) or BI encoding (PBI+BI).

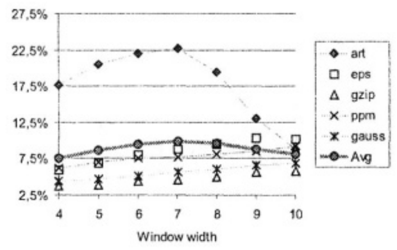

Figure 4. Relative activity reduction as function of window width: PBI-Reconf with no default encoding (PBI+NE)

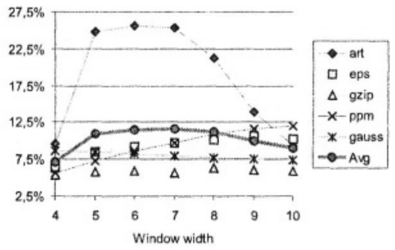

Figure 5. Relative activity reduction as function of window width [bit]: PBI-Reconf with BI as default codec (PBI+BI)

In a first experiment we determined the influence of the window width which is used for parameter observation on the encoding efficiency. The relative savings for the test data streams are depicted as a function of *window width* given in *bit* for no alternative encoding in Fig. 4 and for BI encoding during the reconfiguration period in Fig 5. Although the encoding efficiency is as expected influenced by the alternative encoding both plots show a dependency on the window width. The most efficient reduction is achieved for *art* since adaptive schemes can exploit the strongly over time varying activity profile. The efficiency varies for the streams. In order to achieve the maximum reduction for each data stream different window widths would be required. Since we focus on a general-purpose scheme the average performance *Avg* is regarded. The highest average reduction is achieved for window widths between 2^5 (window width 5) and 2^7 (window width 7).

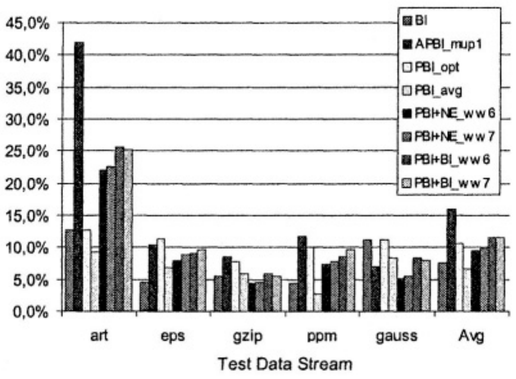

Figure 6. Comparism of the relative reduction in activity

In a second experiment the savings were compared to other encoding schemes: APBI, BI and PBI. PBI was optimized for each of the streams which is indicated by the subscript *opt.* Additionally we investigated in real-life conditions when PBI encodes streams with different statistical parameters. The results are marked by the subscript *avg.* The most efficient reduction in activity was achieved by APBI as depicted in Fig. 6 since the optimization can be applied instantaneously. Nevertheless, that opportunity does not exist in DSM technologies anymore since the scheme is susceptible to transmission errors. The optimized version of PBI is very efficient as long as the data to be transmitted corresponds to the data the scheme is optimized for. Otherwise the efficiency decreases as shown by $\mathbf{PBI}_{avg}$. The proposed PBI-Reconf performs in both variations (BI and NE) comparable to $\mathbf{PBI}_{opt}$. In both variations it outperforms BI by 25 to 50 percent.

In a next experiment we investigated in the hardware requirements. The codec control block which comprises of parameter extraction, encoding rule computation and bit-stream generation unit used for manipulating the partial bit-stream accordingly was described in synthesizable VHDL. We synthesized the block for our target FPGA. The hardware requirements were compared with that of APBI and BI. The results are presented in Tab. 1. As the figures indicate APBI requires compared to BI, a very high overhead in order to adapt the encoding rule periodically to the statistics of the data stream. The main

Table 1. **Hardware requirements on XILINX VIRTEX FPGA**

Scheme		*FG*	*CY*	*DFF*
APBI_{mup1}	*cod*	*725*	*9*	*344*
	dec	*569*	*9*	*314*
BI	*cod*	*115*	*0*	*98*
	dec	*32*	*0*	*65*
$\mathrm{CC}_{PBI,Reconf}$		*732*	*26*	*308*
1Source, 1Sink	*APBI*	*1294*	*18*	*658*
	PBI-Reconf	*879*	*26*	*471*
1Source, 2Sink	*APBI*	*1863*	*27*	*972*
	PBI-Reconf	*911*	*26*	*536*

portion of the increased hardware overhead is caused by the mask computation block (MC) which observes the statistics and computes a new encoding rule. In contrast to that our proposed PBI-Reconf scheme requires a single observation block per circuit and a PBI codec system with the complexity of BI which is periodically adapted by partial reconfiguration. The second part of Tab. 1 shows the requirements for an adaptive codec system with one or two sinks, respectively. In comparism with APBI our approach reduces the hardware requirements by about 30% if a single sink is connected to the bus. If more receivers are implemented the savings increase even more. For 2 sinks the hardware can be reduced to about 50% of a comparable APBI codec while the adaption is independent of any transmission errors on the bus.

6. CONCLUSIONS

The experimental results confirm the efficiency of our proposed scheme. We achieved a higher encoding efficiency than BI and the average PBI with both variations of alternative encoding. Since the reconfiguration process lasts longer than the application of the new mask in APBI the reduction in activity was lower than for APBI. The most important advantage is the independence of transmission errors. Unlike APBI the encoding rule of the decoder is not derived from the received data but computed in the central codec control block and applied by partial reconfiguration. Therefore the scheme is suited for application in DSM environments where, due to coupling effects, transmission errors on system buses occur from time to time. In contrast to that the encoding rules of APBI diverge from each other as soon as a transmission error results in a different encoding rule on decoder side. The investigations in the hardware overhead showed that in comparism to APBI the overall hardware overhead including the bit-stream generation could be reduced by about 30 %. That benefit even increases if data is transmitted to more than one receiver. Furthermore the approach is very flexible regarding the scheme which is adapted. Instead of PBI which we chose as an example also one of the schemes which takes into account coupling activity such as coupling-driven BI or odd/even BI can be applied.

REFERENCES

Benini, L. and DeMicheli, G. (2002). Networks on chips: A new soc paradigm. *IEEE Jounal on Computer,* pages 70–78.

Benini, L., Macii, A., Macii, E., Poncino, M., and Scarsi, R. (1999). Synthesis of Low-Overhead Interfaces for Power-Efficient Communication over Wide Buses. In *36th DAC.*

Benini, L., Micheli, G. De, Macii, E., Sciuto, D., and Silvano, C. (1998). Address Bus Encoding Techniques for System-Level Power Optimization. In *DATE.*

Haase, A., Kretzschmar, C., Siegmund, R., Mueller, D., Schneider, J., Boden, M., and Langer, M. (2002). Design of a Reed Solomon Decoder using Partial Dynamic Reconfiguration of XILINX Virtex FPGAs - A Case Study. In *DATE,* Paris, France.

IBM (2003). Embedded fpga cores. http://www-3.ibm.com/chips/products/asics/products/cores/efpga.html.

Kim, Ki-Wook, Baek, Kwang-Hyun, Shanbhag, Naresh, Liu, C.L., and Kang, Sung-Mo (2000). Coupling-driven signal encoding scheme for low-power interface design. In *ICCAD.*

Kretzschmar, C., Siegmund, R., and Mueller, D. (2000). Adaptive Bus Encoding Technique for Switching Activity Reduced Data Transfer over Wide System Buses. In *PATMOS 2000,* pages 66–75, Goettingen, Germany. Springer.

Kretzschmar, C., Siegmund, R., and Mueller, D. (2003). Low Power Encoding Techniques for Dynamically Reconfigurable Hardware. *Special issue of the Kluwer Journal of Supercomputing,* 26(2):185–203.

Lekatsas, H. and Henkel, J. (2002). ETAM++: Extended Transition Activity Measure for Low Power Address Bus Designs. In *ASP-DAC,* pages 113–120.

Lv, T, Wolf, W., Henkel, J., and Lekatsas, H. (2002). An adaptive dictionary encoding scheme for soc data buses. In *Proceedings of DATE,* page 1059. IEEE Computer Society.

Mamidipaka, M., Hirschberg, D., and Dutt, N. (2001). Low power address encoding using self-organizing lists. In *ISLPED,* pages 188–193. ACM Press.

Satoshi Komatsu, Makoto Ikeda, and Kunihiro Asada (2000). Bus Data Encoding with Adaptive Code-book Method for Low Power IP Based Design. In *International Workshop on IP based design and Synthesis,* pages 77–81.

Shin, Youngsoo, Chae, Soo-Ik, and Choi, Kiyoung (1998). Partial Bus-Invert Coding for Power Optimization of System Level Bus. In *ISLPED,* pages 127–129.

Stan, Mircea R. and Burleson, Wayne P. (1995). Bus-Invert Coding for Low-Power I/O. In *Transactions on VLSI Systems,* volume 3, pages 49–58.

Sylvester, D. and Keutzer, K. (1998). Getting to the Bottom of Deep Submicron. In *ICCAD.*

Sylvester, D. and Keutzer, K. (2001). Impact of Small Process Geometries on Microarchitectures in Systems on a Chip. *Proceedings of the IEEE,* 89(4):467–489.

X.L.Su, X.Y.Tsui, and Despain, A.M. (1994). Saving Power in the Control Path of Embedded Processors. *IEEE Design and Test of Computers,* 11:24–30.

Yang, J. and Gupta, R. (2001). Fv encoding for low-power data i/o. In *International symposium on Low power electronics and designISLPED,* pages 84–87. ACM Press.

Zhang, Yan, Lach, John, Skadron, Kevin, and Stan, M. (2002). Odd/even bus invert with two-phase transfer for buses with coupling. In *ISLPED,* pages 80 – 83. ACM.