INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Power Consumption in Networks-on-Chip by Encoding Scheme

A. V. Manoj
Department of ECE, S. V. College of Engineering, Tirupati, India
S. Bhavya Sree
Department of ECE, S. V. College of Engineering, Tirupati, India
K. Yuva Kumar
Department of ECE, S. V. College of Engineering, Tirupati, India
V. Purandhar Reddy
Department of ECE, S. V. College of Engineering, Tirupati, India

Abstract:
Power has become an important design criterion in modern system designs, especially in portable battery-driven applications. A significant portion of total power dissipation is due to the transitions on the off-chip address buses. This is because of the large switching capacitances associated with these bus lines. There are many encoding schemes in the literature that achieve a huge reduction in transition activity on the instruction address bus. However, on data and multiplexed address buses, none of the existing schemes consistently achieve significant reduction in transition activity. Also, many of the existing techniques add redundancy in space and/or time. In this paper, novel encoding schemes are proposed that significantly reduce transitions on these buses without adding redundancy in space or time. Also, for applications with tight delay constraints, configurations with minimal delay overhead while still achieving significant reduction in transition activity are proposed. Results show that, for various benchmark programs, these techniques achieve reduction of up to 54% in transition activity on a data address bus. On a multiplexed address bus, there is a reduction of up to 61% using our techniques. The proposed schemes are then compared with the existing schemes. It is seen that on an average, the reductions achieved by our techniques are twice those obtained using the current scheme on a data address bus and 55% more than those for multiplexed address bus.

Key words: *Network on Chip; Low power; Data encoding; Coupling capacitance; Power analysis*

1. Introduction

Continuous improvements in semiconductor technology make it possible that a whole computing system comprising processors, memories, accelerators, peripherals, etc. can now be integrated in a single silicon die. This trend has been favoured thanks to the definition of new design methods which stress the reuse of pre-designed and pre-verified modules in the form of intellectual properties (IPs or cores). As the number of IPs used to implement the functionalities demanded by the current systems-on-a-chip (SoCs) increases, the role played by the on-chip interconnection system becomes more and more important. The International Technology Roadmap for Semiconductors [1] depicts the on-chip communication issues as the limiting factor, for performance and power consumption in current and next generation SoCs. Network-on-Chip (NoC) is generally viewed as the ultimate solution for the design of modular and scalable communication architectures. A NoC-based communication infrastructure promises flexibility in network topology, the support of advanced routing algorithms, flow-control and switching techniques, and the possibility of guarantying quality-of-service requirements. These advantages over bus-based architectures comes at the cost of increase in complexity which pushes the communication system to become one of the main elements of a SoC which strongly impact the cost, power, and performance figures of the overall system. For instance, in the Intel's 80-tiles TeraFLOPS processor [2] over 30% of the chip area is dedicated to the communication system and the communication power accounts for about 28% of the total. At the Massachusetts Institute of Technology RAW chip [3] the NoC is responsible for the 40% os the system power. In this paper we focus on power dissipation and energy consumption issues related to the communication system of a NoC-based SoC. We propose the use of data encoding technique as a viable way to reduce both the power dissipation and energy consumption on the links of a NoC. Several data encoding schemes previously proposed for bus-based architectures were applied in the NoC context [4]. However, the impact of the coupling capacitance, which is dominant, especially in deep-sub-micron (DSM) technologies, had not been taken into account.

Differently from [5] we propose an end-to-end encoding scheme which exploits the wormhole switching technique and which is transparent to the NoC (i.e.,it does not require the re-design of routers and links). We assess the proposed encoding scheme and architecture on a set of representative data streams. We compare the proposed approach with the bus-invert coding [6] and the coupling driven bus invert coding [7] as they have the highest potential for power saving while still represent a feasible implementation for on-chip communication. The comparison embraces not only the power/energy figures, but also the implications of silicon area and delay due to the overhead of the encoder/decoder logic in the network interfaces (NI). We show that the proposed data encoding scheme outperforms the other proposals allowing to save up to 26% in power dissipation and 9% in energy consumption.

2. Overview of the Proposed Scheme

In this section we give a brief overview of the proposed data encoding scheme and its application in the NoC context. Since the proposed scheme leverages on the use of wormhole routing as switching technique used in the NoC, in the next subsection we recall some notions about wormhole routing. Then, we will present the proposed scheme in a sketch.

2.1. Characteristics of Sequential Data

Statistics show that typically, in the execution of a program, 15% of the instructions are branches or jumps [10]. This means that, on the instruction address bus, there will be a change of address sequence 15% of the time and the remaining 85% of the time there will be sequential accesses. Since addresses on the instruction address bus are sequential most of the time, we first analyze the characteristics of a completely sequential set of data. Let L be the length of the sequence data and W be the width of the data (A_{W-1}, A_{W-2} , ….. A_1, A_0). A sample sequential address stream of width 4. It can be noticed that: The low-order bit flips almost 100% of the time, while the probability of a flip drops off geometrically for increasing bit significance. The probability of a flip on bit position i is 2^{-i} (i from 0 to W-1). It can be shown that the ratio of the number of toggles on bit position i to the total number of toggles over the complete sequence of data L to be $\sim 2^{-(i+1)}$, irrespective of the length of sequential data.

- It follows that bit lines 0, 1, 2 contribute ~87.5% of the total number of toggles that occur in the sequence data.
- Also, the bit lines have recurring patterns, with the recurring pattern length equal to $2^{(i+1)}$, for bit position i.

Further analysis of recurring patterns in sequential data shows that the recurring patterns have a characteristic:

$X_{i+p/2}$ = complement (X_i) = $X_{i-p/2}$ for i > p/2

Where X is the single bit stream and p is the recurring pattern length and X_i denotes i-th data in bit stream X. Now, we propose to encode functions to reduce the transitions that occur on the instruction address bus.

Encoding functions for the instruction address bus:

Typically, data on an instruction address bus is sequential 85% of the time [10]. Hence the characteristics of the sequential data are used to define the following encoding functions to reduce the number of transitions on the bus.

As was seen, the bit lines have a recurring pattern when the data on the bus is sequential. For a recurring pattern of length p, it can be proved that the function, ENC1, of the form $Y_i = X_i \oplus X_{i-1} \oplus X_{i-2} \oplus ... \oplus X_{i-p+1}$ yields the minimum number of toggles. "⊕" represents an Exclusive-OR function [16]. Note that since the recurring pattern lengths of different bit lines of sequential data are different, the encoding functions would be different on each bit line. While this encoding eliminates all transitions on the corresponding bit line if the addresses are sequential, the implementation of this encoding function requires (p-1) storage elements and (p-1) 2- input XOR gates and the same amount of logic in implementing the decoding function. Also the delay induced in the critical path of the encoding and decoding functions increases for longer recurring pattern lengths, which may not be desirable. Fortunately, the recurring patterns are the longest in higher order bit lines of the bus in which the transitions are very few. So this encoding can be applied only on a few low order bit lines that carry most of the transitions.

ENC2: $Y_i = X_i \oplus X_{i-p/2}$

Where, p is the recurring pattern length and is even. Since X_i and $X_{i-p/2}$ are complements of each other (from 1), this encoding function will always result in logic '1' given that the incoming bit stream follows the recurring patterns in the sequence data. This encoding function adds the delay of only one 2-input XOR gate on the critical path irrespective of the length of the recurring pattern. Now we consider the encoder and decoder implementations of both ENC1 and ENC2 for an example, recurring pattern 0011, with recurring pattern length p=4.

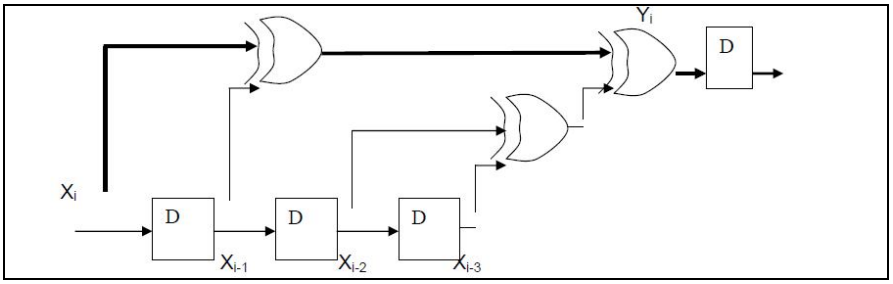

Figure 1: Implementation Structure of the Encoding Logic (ENC1)

Since p=4, ENC1 will be $Y_i = Xi \oplus X_{i-1} \oplus X_{i-2} \oplus X_{i-3}$. The implementation of this encoding function is shown in Figure 1. The corresponding decoding function will be $X_i = Y_i \oplus X_{i-1} \oplus X_{i-2} \oplus X_{i-3}$, the implementation structure being similar to that of the encoder. Similarly the encoding function for recurring pattern 0011 using ENC2 will be $Y_i = X_i \oplus X_{i-2}$ and the implementation is shown in Figure 3.

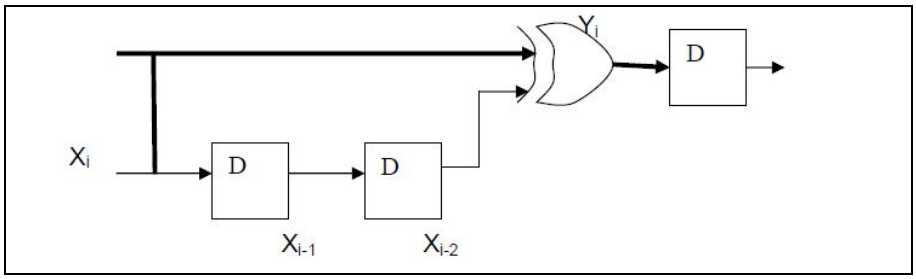

Figure 2: Implementation Structure of the Encoding Logic (ENC2)

The bold lines shown in the Figures 1 and 2 indicate the delay overhead in the critical path. The encoder inserts a one-cycle delay between arrival of address and output of the encoding. As indicated in [5], this extra delay is not an overhead because even if binary code (without encoding) were used, the flip-flop at the output of the bus would be needed because the address would be generated by a very complex logic that produces glitches and misaligned transitions. The flip-flops filter out the glitches and align the edges of the clock, thereby eliminating excessive power dissipation and signal quality deterioration.

Advantages of ENC2 compared to ENC1:

- The delay introduced in the critical path is independent of the length of the recurring pattern
- The delay introduced is very minimal and is just the delay of a 2-input XOR gate.
- If there is a discontinuity in the bit sequence, ENC1 will take p more sequential data inputs to settle down while ENC2 needs only p/2 sequential data inputs.

Disadvantages of ENC2 compared to ENC1:

- While ENC1 can be applied on any recurring pattern, ENC2 has limited applicability. (ENC2, is most suited for instruction address buses.)

In the following sections we propose some adaptive encoding techniques based on some heuristics for reducing the transitions on address buses.

3. Adaptive Encoding for Instruction Address Buses

In our adaptive encoding technique, all possible input symbols are assigned codes. For every input symbol, the corresponding encoding is transmitted and the codes are adapted (updated) based on the current input symbol and current encodings.

3.1. Swap Based Adaptive Encoding

On instruction address buses, since the addresses are mostly sequential, we use a heuristic to send the same code when the addresses are sequential by swapping the code of the current address with the code of the next address in sequence. That is, for every address to be transmitted the corresponding code is put on the bus and the code for this address is swapped with the code of the next address in sequence. So if the addresses are sequential the same code is transmitted, thereby eliminating the transitions on the bus. We illustrate this with an example for a 2-bit address bus.

Let the initial encoding for the possible addresses 0, 1, 2, and 3 be 0, 1, 2, and 3 respectively. Let the actual address sequence be: 0 1 2 3 3 2 3 0 2 3 0. The first incoming symbol is 0. Since the code for 0 in the encoding array is 00 initially, the code transmitted for symbol 0 is 00. Then the codes for symbols in the incoming array are adapted based on the current incoming symbol. Since the next symbol that could come is more likely to be 1 (symbol sequential to 0), the code for 0 is swapped with code for 1 so that if the next incoming symbol happens to be 1, the same code for 0 previously is transmitted thereby reducing the transitions. This is repeated over all the incoming symbols. Note that the code that is transmitted differs from the previous transmitted code only if there is a discontinuity in the incoming symbol sequence. Also, the symbols could be decoded at the receiving end by having a similar encoding array at the other end with the same initialization as the one at the transmitting end. The only difference being that the encoding array at the receiving end is updated based on the symbol that is decoded from the incoming code.

The structure of the implementation of SWAP based adaptive encoding for 2-bit address bus is shown in Figure 3.All the signal lines in the Figure 4 are 2-bit lines. C_{00}, C_{01}, C_{10}, and C_{11} are the current codes for addresses 00, 01, 10, and 11 respectively. N_{00}, N_{01}, N_{10}, and N_{11} are the adapted next encodings that depend on the current input X_0X_1 and current codes C_{00}, C_{01}, C_{10}, and C_{11}. As can be seen the new code for given address is either the same code or is swapped with the neighboring address. Consider the MUX4 in Figure 4. If the inputs are 00 or 01, the code for 11 holds the value ($N_{11} = C_{11}$) since the next address in sequence of neither of these addresses is 11. When the input is 10, the sequential address of 10 is 11, so the code for 11 is swapped with the code for 10. i.e, $N_{11} = C_{10}$ and $N_{10} = C_{11}$. Similarly, when the input is 11, since the next address in sequence for 11 is 00, the code for 00 is swapped with the code for 11 i.e., $N_{00} = C_{11}$ and $N_{11} = C_{00}$. The decoder for the SWAP based adaptive encoding will have a similar structure as the encoder in Figure

4, the only difference being that the select signal to the SEL-MUX will be the encoded address Y_0Y_1 and the output of this SEL-MUX gives the actual address, X_0X_1. Also, the delay element after the SEL-MUX will be absent for the decoder. The delay induced in the critical path in both encoder and the decoder, is simply the delay of the 4-1 multiplexer for 2-bit address bus.
Note that the number of ENC-MUX's, storage elements and the size SEL-MUX increases exponentially with the number of address bits. Also the delay induced in the critical path increases with the number of address bits because of the increasing size of the SEL-MUX. But as we noted earlier, in sequential addresses, the maximum number of transitions occur in the least significant bits. So this encoding could be done only on the last few address bits with significant reduction in the total number of transitions. Our results in Section 7 are presented for SWAP based adaptive encoding on a 32-bit address bus with encoding on least significant 2-bits, 3-bits and 4-bits. Note that all the encoding schemes suggested for the instruction address bus are applied only on the last few address bits. Next we propose heuristics for adaptive encoding on data address buses.

3.2. Adaptive Encoding For Data Address Bus
Unlike the instruction address bus, the addresses on the data address bus are non-sequential most of the time. But still the data addresses follow the spatial and temporal locality principles [10]. That is, it is more likely that there will be an access to a location near the currently accessed location (spatial locality) and it is more likely that the currently accessed location will be accessed again in the near future (temporal locality).

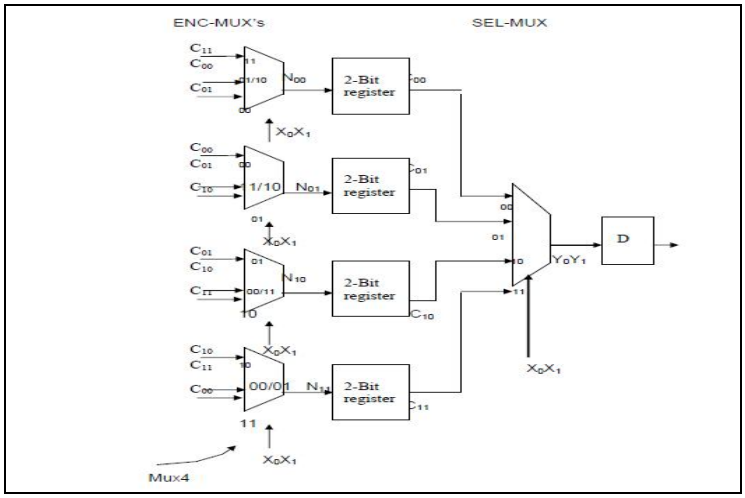

Figure 3: Implementation of Encoder for SWAP based adaptive encoding

In this section we define adaptive encoding techniques based on the heuristics associated with these principles of localities for reducing the transitions on the data address bus. The principle of locality states that most programs do not access all code and data uniformly [10]. We will reduce the number of transitions between the most frequently accessed address ranges by assigning them the codes with minimal Hamming distance. To achieve this, we use Move-To-Front (MTF) and Transpose (TR) methods in self-organizing lists [14] for assigning codes so as to reduce the transitions on the address bus. Move-To-Front (MTF) is a transformation algorithm that, instead of outputting the input symbol, outputs a code that refers to the position of the symbol in a table with all the symbols. Thus the length of the code is the same as the length of the symbol. Both the encoder and decoder initialize the table with the same symbols in the same positions. Once a symbol is processed, the encoder outputs its position in the table and then the symbol is shifted to the top of the table (position 0). All the codes that from the position 0 until the position of the symbol being coded, are moved to the next higher position. This simple scheme assigns codes with lower values for more redundant symbols (symbols which appear more frequently). We illustrate this with the following input data sequence: 0 1 0 0 2 0 1 0 3. Shows encoding and decoding of the data using MTF. The Transpose (TR) algorithm is similar to MTF in the way the code assigned to the symbol being the position of the symbol, but instead of moving the symbol to the front, the symbol is exchanged in position with the symbol just preceding it. If the symbol is at the beginning of the list, it is left in the same position. Note that, in both MTF and TR, the most frequent incoming symbols are at the beginning of the list and the Hamming distance associated with these symbols is smaller. So, these heuristics are very useful in data address buses in which there is a greater likelihood of two different address sequences being sent on the bus (two arrays being accessed alternatively, reads from an address space and writes to a different address space, etc.). In such cases, we would like to keep the encoding of these addresses as close as possible i.e., with minimal Hamming distance. The Move-To-Front (MTF) and TRANSPOSE heuristics achieve the goal. Figure 4 shows the implementation of the encoder for MTF/TRANSPOSE based adaptive encoding for a 2-bit bus. A straightforward implementation of the encoding method as suggested in the algorithm would be impractical because searching for the symbol in the array and sending the index of the array would add a huge delay overhead on the critical path. A better way for implementing this would be to keep the location of the symbol fixed and for every incoming symbol, update the codes of the symbols. Figure 4 shows the implementation in which the symbol location is fixed and the code for the symbols is changed based on the current input symbol and the current code of the symbol. The SEL-MUX does the job of selecting the

corresponding code for X_1X_0. The combinatorial logic in front of the registers does the job of updating the codes depending on the current codes of these symbols and the output code.

For MTF, the combinatorial logic will have the functionality in the following way:

N_{xx} = C_{xx} if $Y_0Y_1 < C_{xx}$

= $C_{xx} + 1$ if $Y_0Y_1 > C_{xx}$

= 00 if $Y_0Y_1 = C_{xx}$

For Transpose, the combinatorial logic will have the functionality as given below:

N_{xx} = $C_{xx} - 1$ if ($Y_0Y_1 = C_{xx}$) and ($C_{xx} \neq 0$) = $C_{xx} + 1$ if ($Y_0Y_1 = C_{xx} + 1$) = C_{xx}

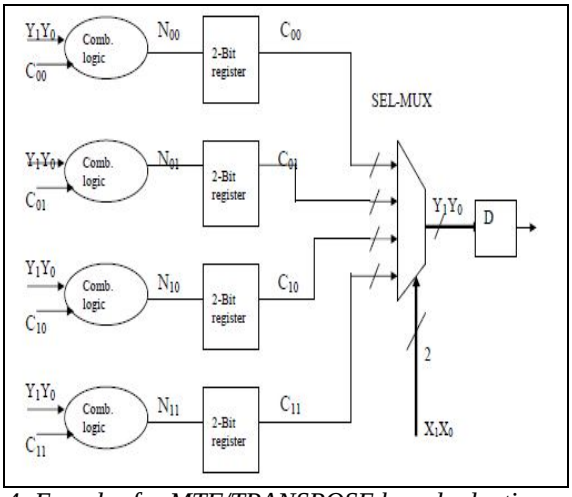

Figure 4: Encoder for MTF/TRANSPOSE based adaptive encoding

Note that, by using this implementation structure, in the critical path only a 4-1 multiplexer delay is being introduced for a 2-bit address bus. Similar to the SWAP based adaptive encoding, the number of storage elements needed and the size of SEL-MUX increase exponentially with the number of address bits. So we use a standard method of splitting the address bus into smaller buses and then applying this encoding on each of these smaller buses independently. For example, a 32-bit address bus can be split into 16 smaller buses each with 2-bits. The encoding can be applied independently on each of these 2-bit buses. The results in the next section are shown for a 32-bit address bus and splitting it into different smaller bus sizes.

3.3 Adaptive Encoding for Multiplexed Address Buses

On the multiplexed address bus, both instruction and data addresses are sent on the same bus. So a significant percentage of addresses on the multiplexed address bus would still be sequential. Also, these addresses still follow the principle of locality. We propose a heuristic to combine the techniques proposed for instruction and data address buses on the multiplexed address bus. The proposal is to apply encoding schemes discussed for instruction address bus on the least significant bits and those for data address bus on the higher address bus bits. When the addresses of multiplexed bus are sequential, most transitions occur on least significant bits. The techniques for instruction address bus on least significant bits minimize the transitions in such cases. Also, the addresses follow the principle of locality. So the schemes for data, address bus applied to higher significant bits give further reduction in transition activity. Results have been presented in Section 7 for various combinations of instruction and data address bus encoding techniques applied on multiplexed bus.

4. Results

In this section, we show the reduction in transition activity obtained by applying the techniques discussed in previous sections of address streams of several programs. We then compare these results with those obtained with existing techniques. We also compare the delay overheads of these techniques. The address bus traces of the programs were obtained by running them on an instruction-level simulator, SHADE [15] on a SUN Ultra-5 workstation. The comparison is made in terms of the total number of toggles on the bus before and after the encoding is applied.

Data Flow Diagrams

Data Flow Diagrams For NOC Project

Control Data Pins

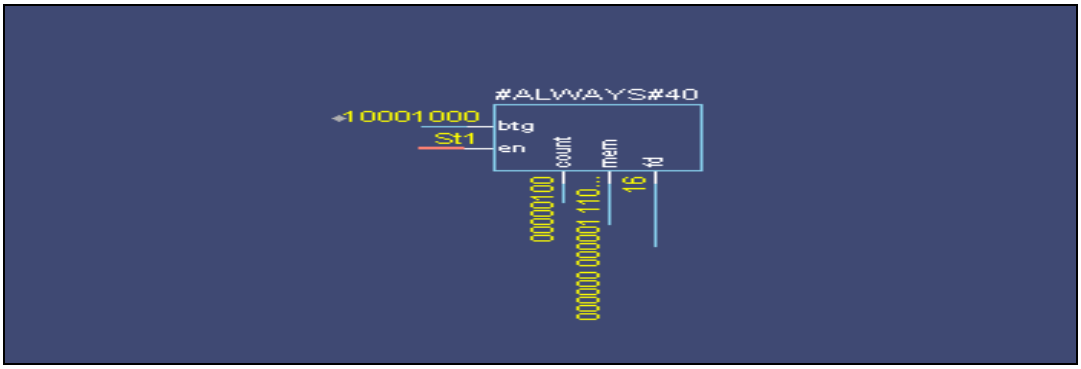

Figure 5

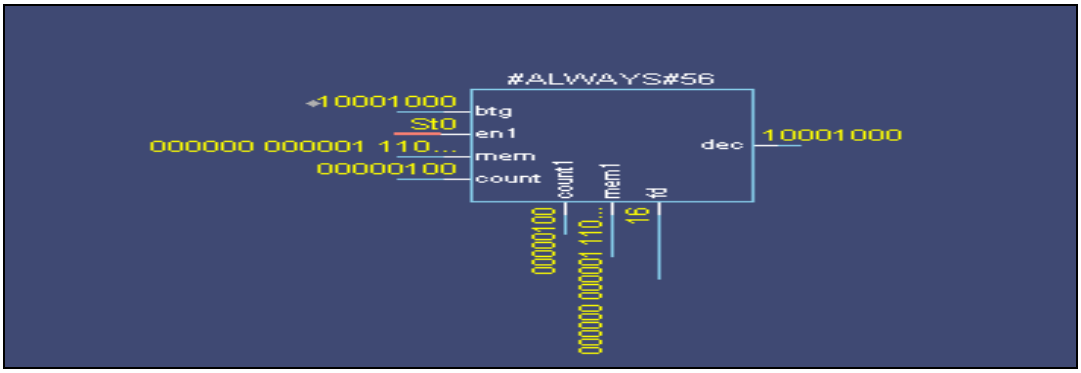

Figure 6

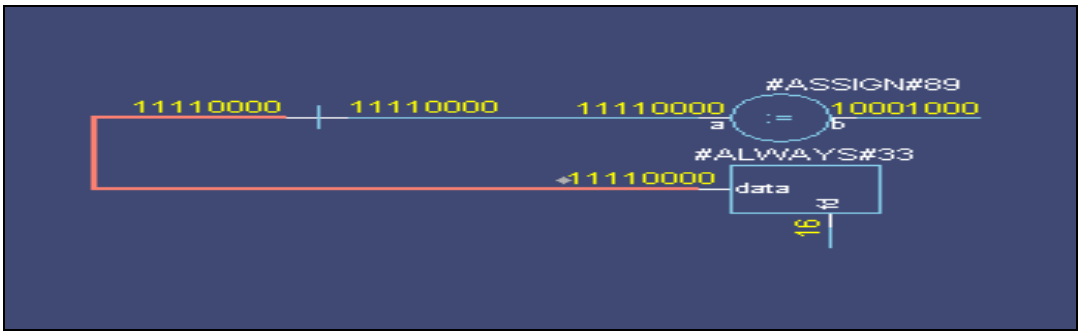

Figure 7: Input Data

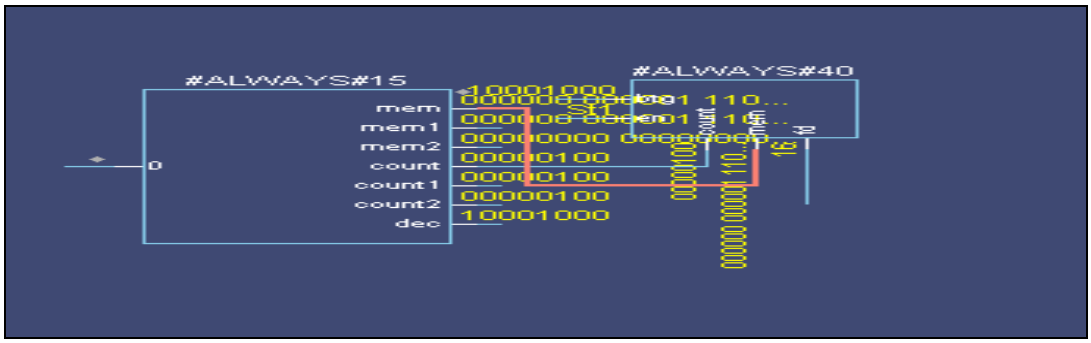

Figure 8: Encoded Data

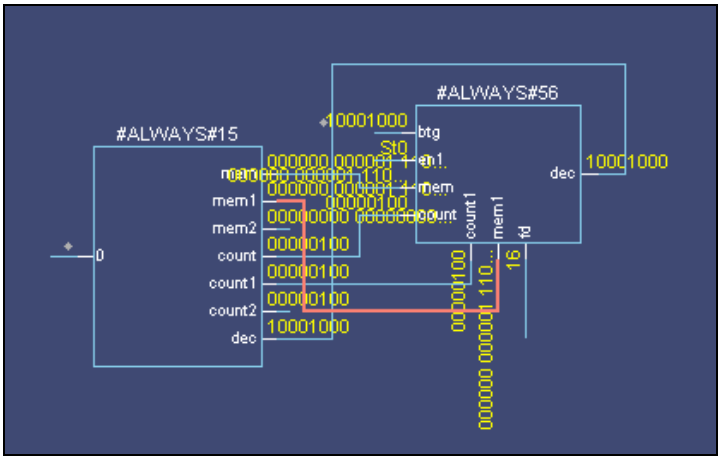

Figure 9: Decoded Data

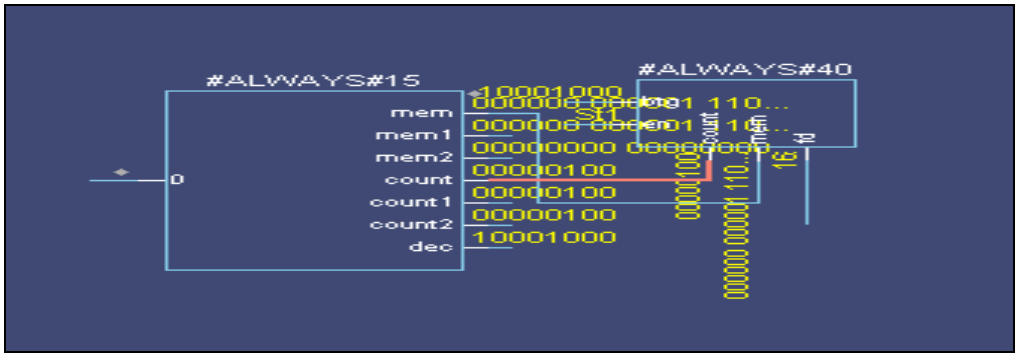

Figure 10: Counter Values

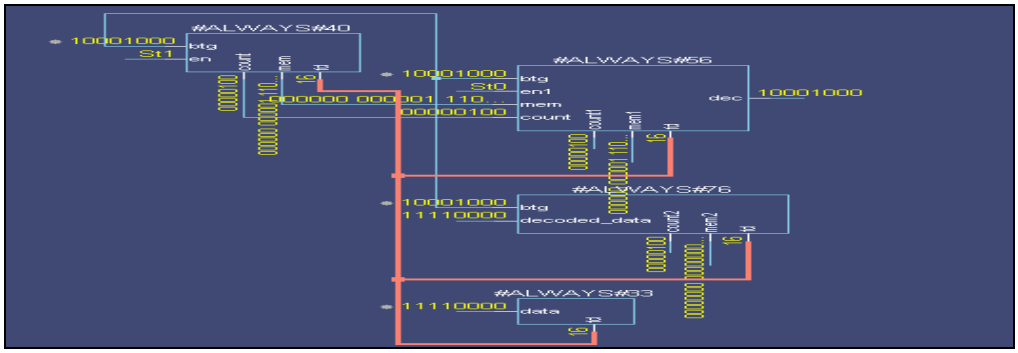

Figure 11: Networking Datas

Simulation Results

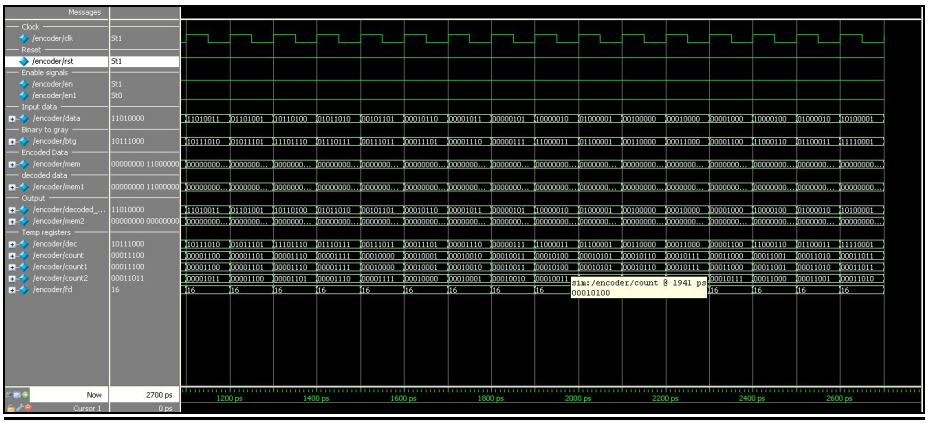

Figure 12: Shows the Simulation Result of the Final Encoded Data

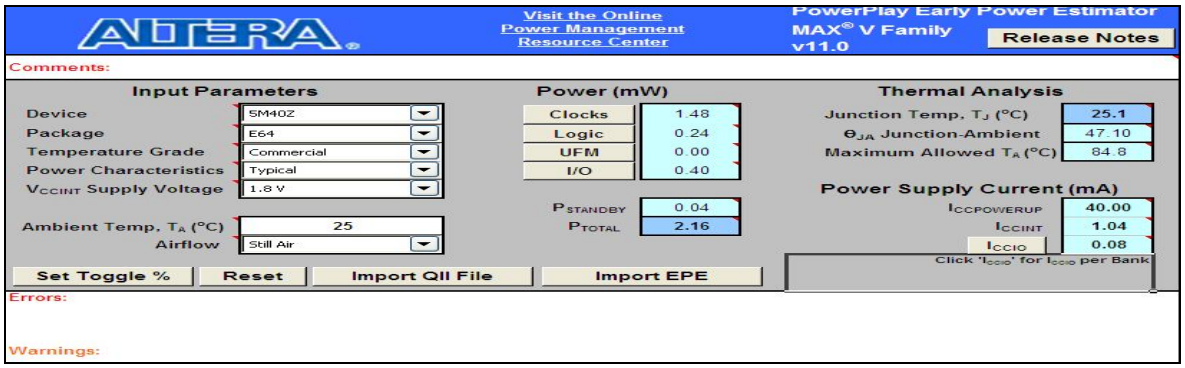

Figure 13: Power Result of Phase 1 methodology

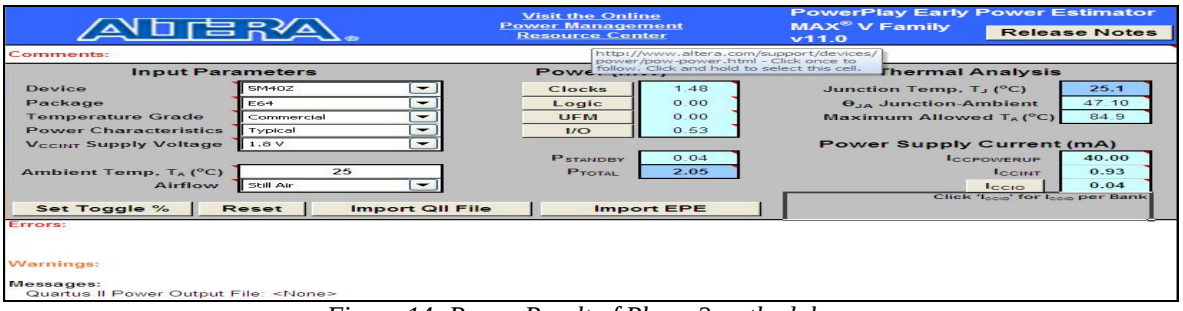

Figure 14: Power Result of Phase 2 methodology

5. Conclusion

We have proposed several encoding techniques for the address buses. For instruction address buses, two encoding functions ENC1 and ENC2 and an adaptive encoding technique, SWAP is proposed. For data address buses, MTF and TRANSPOSE, adaptive encoding techniques based on self-organizing lists, have been proposed. For multiplexed address bus, a combination of encoding techniques has been proposed. The techniques proposed for instruction address bus are applied only on few least significant bits. This enables the usage of these techniques in the multiplexed address bus along with the techniques proposed for data address bus.

While the INC-XOR could be used for encoding on the instruction address bus, our techniques could be used for data and multiplexed address bus. The techniques proposed for the data address bus and multiplexed address bus, outperform the existing techniques. Results show that 4-bit MTF with transition signaling applied to various data address streams gives up to 51% reduction in transition activity. On the multiplexed address bus, the 4-bit SWAP + MTF on various address streams yields a reduction of up to 61%. We also showed the configurations that have very little delay overhead but still give significant reduction in transition activity.

None of the proposed techniques add redundancy in space or time. In some applications, redundancy in space in time might be tolerable. We are trying to develop techniques, which give better reduction in transition activity for such applications, by adding some redundancy in space or time. Also, we are looking at how the proposed techniques could be applied on data of the data buses if the characteristics of the data are known a priori.

6. References

1. "International thechnology roadmap for semiconductors – interconnect,"Semiconductor Industry Association, 2006.
2. S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan,A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote,N. Borkar, and S. Borkar, "An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," IEEE Journal of Solid-State Circuits, vol. 43, no. 1,pp. 29–41, Jan. 2008.
3. M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The raw microprocessor: a computational fabric for software circuits andgeneral-purpose programs," IEEE Micro, vol. 22, no. 2, pp. 25–35, 2002.
4. J. C. S. Palma, L. S. Indrusiak, F. G. Moraes, A. G. Ortiz, M. Glesner,and R. A. L. Reis, "Inserting data encoding techniques into NoC-based systems," in IEEE Computer Society Annual Symposium on VLSI, Mar.2007, pp. 299–304.
5. A. Jantsch, R. Lauter, and A. Vitkowski, "Power analysis of link level and end-to-end data protection in networks on chip," in IEEE International Symposium on Circuits and Systems, vol. 2, May 2005, pp. 1770–1773.
6. M. R. Stan and W. P. Burleson, "Bus invert coding for low power I/O," IEEE Transactions on Very Large Scale Integration Systems, vol. 3, pp.49–58, Mar. 1995.
7. K. W. Kim, K. H. Baek, N. Shanbhag, C. L. Liu, and S. M. Kang,"Coupling-driven signal encoding scheme for low-power interface design," in IEEE/ACM International Conference on Computer-aided Design, 2000, pp. 318–321.

8. L. M. Ni and P. K. McKinley, “A survey of wormhole routing techniques in direct networks,” IEEE Computer, vol. 26, pp. 62–76, Feb. 1993.
9. D. Bertozzi and L. Benini, “Xpipes: a network-on-chip architecture for gigascale systems-on-chip,” IEEE Circuits and Systems Magazine, vol. 4, no. 2, pp. 18–31, 2004.
10. Debra A. Lelewer and Daniel S. Hirschberg, Data Compression. 1987.
11. Xilinx. www.xilinx.com. [Online]. http:// www .xilinx. Com/support/documentation/data_ sheets/ds100. pdf
12. Ian D. L. Anderson and Mohammed A. S. Khalid Jason G. Tong, "Soft-Core Processors for Embedded Systems," in 18th International Conference on Microelectronics, 2006.
13. xilinx., http:// www.xilinx.com/ise/embedded/emb_r ef_guide.pdf
14. www.xilinx.com. [Online]. http:// www.xilinx.com /microblaze/
15. www.xilinx.com. [Online]. http://www.xilinx. com/ support/ documentation/ sw_manuals / edk92i_ctt.pdf
16. Khalid Sayood, Introduction to Data Compression.
17. International Technology Roadmap for Semiconductors (ITRS) Working Group, \International Technology Roadmap for Semiconductors (ITRS), 2009 Edition. "http://www.itrs.net/Links/2009ITRS/Home2009.htm.
18. W. J. Dally and B. Towles, \Route Packets, Not Wires:On-Chip Interconnection Networks," in The 38th InternationalDesign Automation Conference (DAC), 2001.
19. Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, \A5-GHz Mesh Interconnect for a Teraps Processor," IEEEMicro, vol. 27, 2007.
20. M. Taylor, M. B. Taylor, W. Lee, S. Amarasinghe, and A. Agarwal, \Scalar Operand Networks: On-chip Interconnect for ILP in Partitioned Architectures," in The IEEE International Symposium on High Performance ComputerArchitecture (HPCA), 2002.
21. D. Molka, D. Hackenberg, R. Schone, and M. S. Muller, \Memory Performance and Cache Coherency Eects on an Intel Nehalem Multiprocessor System," in The 18th International Conference on Parallel Architectures and CompilationbTechniques (PACT), 2009.
22. Advanced Micro Devices (AMD) Inc., \AMD Opteron Processors for Servers: AMD64-Based Server Solutions for x86 Computing." http://www.amd.com/us-en/ Processors/ Product Information /0 30 118 8796, 00.html.
23. P. Pujara and A. Aggarwal, \Cache Noise Prediction," IEEE Transactions on Computers