UNIVERSAL ROTATE INVERT BUS ENCODING FOR LOW POWER VLSI

Shankaranarayana Bhat M[1] and D. Yogitha Jahnavi[2]

[1]Associate Professor, Department of Electronics and Communication Engineering, Manipal Institute of Technology, Manipal University, Manipal, Karnataka, India
msnbhat@yahoo.com
[2] Student, Department of Electronics and Communication Engineering, Manipal Institute of Technology, Manipal University, Manipal, Karnataka, India
yogithadodda@gmail.com

ABSTRACT

Power dissipation is an important design constraint in today's CMOS VLSI design and is addressed widely by the researchers across the globe. Switching activity is one of the factors that affect dynamic power in a chip and several publications have suggested various techniques to reduce the same. Reduction of switching activity in the busses attains significance as bus width, bus capacitance and the clock are recording continuous uptrend. In this paper, we propose a technique for bus encoding, which, reduces the number of transitions on the bus and performs better than the existing methods such as bus invert coding and shift invert coding for random data in terms of switching activity, without the need for extra overhead in computation and circuit. However, irrespective of the bus width it needs three extra bits and does not assume anything about the nature of the data on the bus.

KEYWORDS

Bus encoding, Switching activity, CMOS VLSI, Power dissipation, Low Power Design.

1. INTRODUCTION

Power dissipation in CMOS circuits is a major concern for VLSI design. The power dissipation in CMOS can be classified in to two types, namely, dynamic and static power dissipation. While the static power dissipation is caused by leakage currents in transistors the major components of dynamic power dissipation is switching power and short circuit power. Switching power is dissipated when there is a transition from 1 to 0 or from 0 to 1. The probability of such transition is called switching activity.

We can express the dynamic power as

$P_d = \propto V_{dd}^2 C_L f_{clk}$

Where, P_d = Dynamic power dissipation, $\propto$ = Switching activity, V_{dd} = Power supply

C_L = Load Capacitance, f_{clk} = Clock Frequency.

The equation for power dissipation in a bus is as follows:

$$P_d bus = \left(\propto V_{dd}^2 C_L f_{clk}\right) N \;;$$

Where N = Width of the bus

Inferring from the above equation, the power dissipation in a bus proportional to the switching activity and the width of the bus.

To reduce power dissipation on a bus the input data is encoded so that the number of transitions between the input data and the data on the bus is reduced. The data on the bus refers to the data which was previously sent over the bus.

Over the years many bus encoding techniques have been proposed and most of them aim at reducing the number of transitions on the bus. Some of them focus on data buses or data address buses while the others focus on instruction address buses.

In instruction address buses the data is sequential, that is, they are consecutive binary numbers. In data buses or data address buses the data is random. We need to have different techniques to encode random data and sequential data in order to reduce the number of transitions.

2. Related Work

There have been many techniques proposed and implemented to achieve reduction of the switching activity in the buses. Following paragraphs briefly describe such existing techniques.

2.1 Bus-Invert Coding [1]

This method uses an extra control bit called invert. The hamming distance (the number of transitions) between the present bus value and the next data value is computed. If this hamming distance is greater than n/2, where 'n' is the width of the bus, then the data value is inverted (bit complemented) and transmitted over the bus. In this case invert=1. If the hamming distance is not greater than n/2, then the data value is transmitted as it is. In this case invert=0.

2.2 Partial Bus-Invert Coding [2]

In PBI coding, we partition a bus into two sub buses based on the behaviour of patterns transferred. For example, we partition a bus B into a selected sub bus S and the remaining sub bus R such that S contains bus lines having higher transition correlation and/or higher transition probability and R contains the remaining bus lines. Because the bus lines in R have low correlation with those in S and low transition activity, inverting those in R may increase rather than decrease the transition activity. Therefore, by applying BI coding only to the sub bus S, we can reduce the hardware overhead as well as decrease the total number of bus transitions.

2.3 Frequent value technique [3]

This technique creates a small table of statistically or dynamically determined common bit patterns. It uses a small look-up table to create one-hot encoding alternatives for the high transition patterns. It XORs the one-hot alternatives with the previous data transmissions. It reduces switching 2-4 times more effectively than bus-invert coding.

2.4 Gray code [4]

In gray code only one bit differs from consecutive numbers. If input patterns are consecutive then they are converted from binary to gray code. The data thus encoded using gray code reduces the number of transitions considerably when compared to binary. An additional redundant line is required.

2.5 T0 / Asymptotic zero-transition technique [5]

The T0 method requires an extra redundant line called INC which indicates to the receiver whether the data is consecutive or not. If INC=1 then the data is consecutive. Hence all others lines on the bus are frozen, to avoid unnecessary switching. The new address is computed directly by the receiver. On the other hand, when the two addresses are not consecutive, INC=0 and the remaining bus lines are used as standard binary codes for the new addresses.

2.6 T0-C (T0-Concise) [6]

This code is an extension of T0 code. It improves T0 code in a number of important ways. First of all, it eliminates the redundant bit. Second, it results in higher power saving on the bus.

2.7 T0-BI technique [5]

This is a mixed encoding, where in two types of encoding-T0 and BI- is employed. In this method, two redundant lines INC and INV are used to convey the receiver the type of encoding is being used.

If both INC =0 and INV=0, then data was transmitted as it is.

If INC=0 and INV=1 then the data has been inverted (bit complemented).

If INC=1 and INV=0 then the data is consecutive and the bus is frozen till INC=0. The receiver directly computes the new addresses.

2.8 T0-BI_1 technique [7]

T0_BI_1 design uses only one control line, called INCV, to control both INC and INV functions to transmit a data address, the encoder of T0_BI_1 first detects the continuity of the transferred address sequence. The continuity means that the current address is equal to the sum of the previous address and the stride. Transmission of an address with continuity property is done with an asserted INCV and a frozen address bus.

If continuity test fails, then the encoder checks to see if the address pattern produces bus bit-transitions on more than half of the address bus lines and the inverted address pattern is not equal to the previous bus value, then the INCV is also asserted and the inverted address is sent over the address bus.

Otherwise, the INCV line is de-asserted and the address will be sent directly.

2.9 Dual-T0 technique [5]

In the case of multiplexed address buses two address buses α and ß 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 (sequential).Stream ß, corresponding to data addresses, has almost no in-sequence patterns (random).The control signal SEL (used to de-multiplex the bus) is high for stream α and low for stream ß.

In this technique, T0 code is applied when SEL is high (i.e., stream α) and binary code is applied when SEL is low (i.e., stream ß).

2.10 Dual T0-BI technique [5]

This technique requires an additional control signal called INCV to indicate to the receiver whether the data is inverted or not. When SEL=1 and INCV=1 then the incoming data is consecutive. Hence T0 coding is applied. When SEL=0 and INCV=1 then the received data is inverted (bit complemented). When INCV=0, irrespective of the value of SEL, then the received data is unmodified.

2.11 T0-XOR technique [8]

This technique extends the capabilities of the T0 code by combining it with a xor function:

$$B(t)= \begin{cases} [b(t)\ (XOR)\ (b(t-1) + S)]\ (XOR)\ B(t-1) & t > 0 \\ b(t) & t = 0 \end{cases}$$

where B(t) is the value on the encoded bus lines at time t, 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.

2.12 Shift-Inv technique [9]

In this technique we shift the data bits by one bit position (left-shift and right-shift) if the shifting reduces the number of transitions. Among four operations, namely, data shifted left, shifted right, inverted and sent as unmodified, whichever gives the least number of transitions that data is sent on the bus. To indicate the operation performed it requires two extra bits.

2.13 Transition signaling technique [10]

This technique encodes logic 1 as having a transition and logic 0 as lack of transition (or vice-versa).It is efficient with respect to power consumption when the probability of 0 (or 1) is very high. After encoding the input data using this method, the number of transitions is equal to the number of 1s.

2.14 BITS technique [11]

In this technique, if the number of 1's in the input data is more than half the bus width, then each bit is inverted and then encoded using transition signalling. Otherwise, each bit of the input data is encoded using transition signalling without alteration. An extra bus line is used to signal the inversion.

2.15 Offset-XOR technique [8]

In this technique the difference between the current value of b(t) and the previous value b(t-1) is transmitted on the bus. When the bus values transmitted on the buses are highly correlated, the value on the encoded bus lines B(t) is reduced, and it is kept constant for in-sequence data. In the Offset-XOR code, we first compute the difference (b(t) - b(t-1)), then we execute a XOR function to decorrelate the encoded bus lines:

$$B(t) = \begin{cases} (b(t) - b(t-1)) \text{ (XOR) } B(t-1) & t > 0 \\ b(t) & t = 0 \end{cases}$$

This technique does not use any extra redundant bits.

2.16 EXNOR technique [12]

The scheme encodes the data by applying either XOR or XNOR operation.

D(t): un-encoded data to be transmitted at time t

L(D(t)): encoded data on the bus at time t

The m-bit bus is divided into subsets with each subset consisting of 4 bits. Then the number of zeroes in the subset of D(t) are computed.

$$L(D(t+1)) = L(D(t)) \text{ (XOR) } D(t+1) \qquad (1)$$

$$L(D(t+1)) = L(D(t)) \text{ (XNOR) } D(t+1) \qquad (2)$$

If the number of 0s is greater than the number of 1s then we use the first equation.

If the number of 0s is less than the number of 1s then we use the second equation.

If the number of 0s and 1s are equal then there are three different cases.

2.17 Adaptive code-book technique [13]

This technique reduces bus signal transitions by referring a code-book that has the transmitted past data. To adapt the code-book to data streams the code-book is updated at every cycle.

1) Code selection in the code-book

2) Data transmission to the bus lines

3) Code-book updating

2.18 T0-CAC [14]

The basic concept is to assign the least signal transition codes to the most frequent patterns. For example, the following case occurs on the instruction address bus without JUMP or BRANCH operations.

$\mathrm{Data}(t) = \mathrm{Data}(t-1) + S$

This method assigns zero transition code to such cases just like T0 code (or T0-C code). As a result, Data(t) is encoded to Bus(t–1). In addition, we use an adaptive codebook which contains the recently accessed destination address of JUMP / BRANCH operation (JUMP/BRANCH destination address history). In case that the address to be transmitted is hit to one of the addresses in the codebook, this method assigns MSB one-hot code to such cases. For example, in case of a loop operation, the same destination addresses of JUMP/BRANCH operations will frequently appear. By using T0CAC code, the number of signal transitions is effectively reduced in such cases.

2.19 OXAC [14]

Basic concept is very similar to T0CAC code in the sense that both encoding methods use adaptive codebooks which contain history of JUMP/BRANCH destination addresses. In the cases that the address does not hit to the codebook, OXAC encoder transmits an offset between Data(t) and Data(t–1) whereas T0CAC encoder transmits a raw address value(Data(t)).

2.20 Beach solution [15]

This technique makes clusters of bus lines based on statistical information of address patterns.

Then it generates an encoding function for each cluster such that the encoded version of each cluster results in fewer transitions.

The technique has proved to be particularly suitable for special purpose machines, where the same portion of embedded code is executed over and over.

The performances measures of any bus encoding techniques used for the reduction of switching activity include mainly the delay caused by the additional computation required, the hardware overhead needed for implementation of encoding and decoding techniques at the sending and the receiving end respectively causing an increases in area, power dissipation and extra bits required to represent the encoding technique.

Most of the methods try to reduce the number of transitions while also simultaneously trying to mitigate the above mentioned problems.

3. METHODOLOGY

In the proposed method, multiple options will be explored and weighed to see which technique will give the minimum number of switching and the technique so selected will be put on the bus.

The type of coding so selected will be communicated to the receiver by three additional bits. The options tried are listed bellow:

- Invert or bit complement
- Rotate left by one bit
- Rotate right by one bit
- Rotate left by one bit and invert
- Rotate right by one bit and invert

Though the first three options were already reported, we add the last two techniques to get an integrated approach and as seen in the later part of discussion, the combination results better switching efficiency.

The process of encoding is as follows. Unmodified data is encoded using rotate right by one bit and rotate left by one bit codes. The number of transitions between the unmodified data and the data on the bus are computed. This number of transitions is compared with half the bus width (n/2), to check whether it is greater than n/2 or not. If this number of transitions is greater than n/2 then the unmodified data is encoded using the invert code. The number of transitions between this inverted data and the data on the bus is calculated. The number of transitions between the data rotated right by one bit and the data on the bus is computed. This number of transitions is compared with half the bus width (n/2), to check whether it is greater than n/2 or not. If this number of transitions is greater than n/2 then the data rotated right by one bit is inverted. The number of transitions between the data rotated right by one bit and inverted and the data on the bus is calculated. The number of transitions between the data rotated left by one bit and the data on the bus is computed. This number of transitions is compared with half the bus width (n/2), to check whether it is greater than n/2 or not. If this number of transitions is greater than n/2 then the data rotated left by one bit is inverted. The number of transitions between the data rotated left by one bit and inverted and the data on the bus is calculated. Since the numbers of transitions are conditionally computed for any given case, we have three numbers of transitions to be compared. These three numbers of transitions are compared and the least number of transitions is found out. The encoding method corresponding to this least number of transitions is used to encode the data and it is sent over the bus.

At the receiving end, the receiver must be notified as to which method has been used to encode the data being sent over the bus. Since there are six different types of data, a minimum of three extra bits are required to indicate to the receiver which of them is being sent.

The binary codes used to identify the type of encoding selected in this technique are given in table-1.

Table 1. Control bit representation for proposed technique

Bit representation	Method
000	Unmodified data
001	Invert / Bit complement
010	Rotate left by one bit
011	Rotate right by one bit
100	Rotate left by one bit and invert
101	Rotate right by one bit and invert

3.1 Algorithm

The algorithm for the proposed method is as follows:

```
Procedure rotate_inv()
  {
   Input: n_d, d_b, w
   Output: r_i_d
   N= no. of transitions (n_d, d_b)
  If  (N > w/2)
  {
    N_I= no. of transitions (n_d(invert), d_b)
  }
  N_R= no. of transitions (n_d(rotate right), d_b)
  If (N_R > w/2)
  {
    N_R_I= no. of transitions (n_d(rotate right & invert), d_b)
   }
  N_L= no. of transitions (n_d(rotate left), d_b)
  If (N_L > w/2)
  {
     N_L_I= no. of transitions (n_d(rotate left & invert), d_b)
   }
  r_i_d= one of (n_d, n_d(i), n_d(rl), n_d(rr),
          n_d(rli), n_d(rri)) depending on minimum(N, N_I,  N_L,  N_R,  N_L_I,  N_R_I)
}
```

In the given algorithm,

'n_d' is the new input data

'd_b' is the data on the bus

'w' is the width of the bus

'r_i_d' is the encoded data which will be sent on the bus

'N' is the number of transitions between the new input data and the data on the bus

'N_I' is the number of transitions between the inverted data and the data on the bus

'N_L' is the number of transitions between the data rotated left by one bit and the data on the bus

'N_R' is the number of transitions between the data rotated right by one bit and data on the bus

'N_L_I' is the number of transitions between the data rotated left by one bit and inverted and the data on the bus

'N_R_I' is the number of transitions between the data rotated right by one bit and inverted and the data on the bus

The algorithm computes the number of transitions between the data on the bus and the six different types of data conditionally. The type of data which gives the least number of transitions is sent on the bus.

3.2 Flow chart

The flow chart given below shows step by step working of the method.

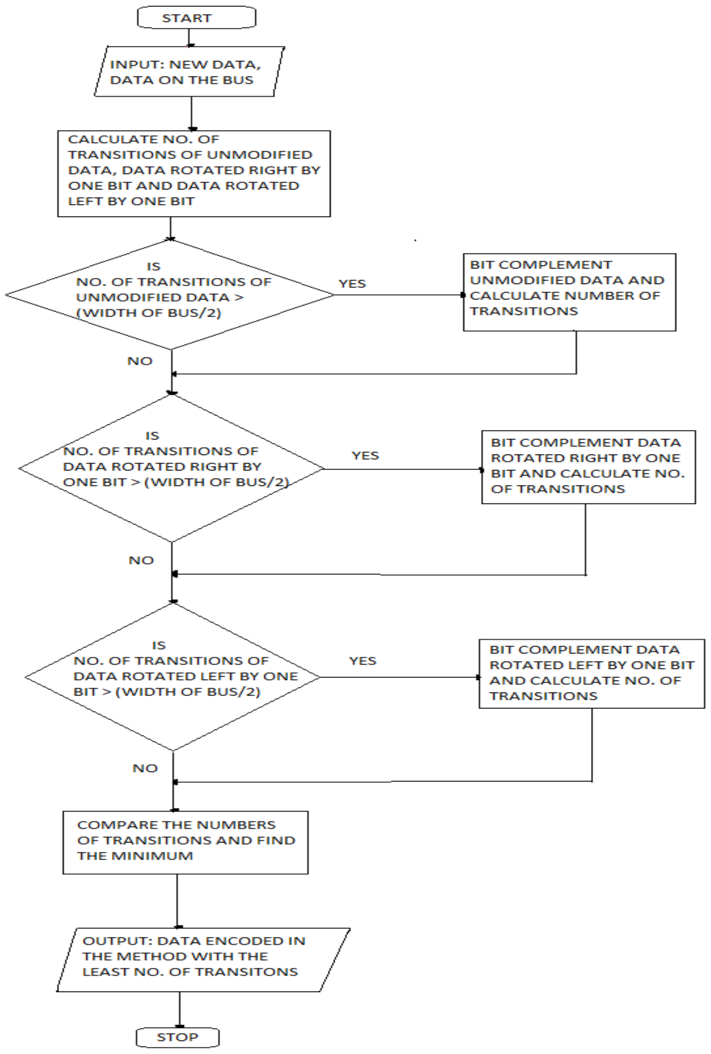
START
INPUT: NEW DATA, DATA ON THE BUS
CALCULATE NO. OF TRANSITIONS OF UNMODIFIED DATA, DATA ROTATED RIGHT BY ONE BIT AND DATA ROTATED LEFT BY ONE BIT
IS NO. OF TRANSITIONS OF UNMODIFIED DATA > (WIDTH OF BUS/2)
YES
BIT COMPLEMENT UNMODIFIED DATA AND CALCULATE NUMBER OF TRANSITIONS
NO
IS NO. OF TRANSITIONS OF DATA ROTATED RIGHT BY ONE BIT > (WIDTH OF BUS/2)
YES
BIT COMPLEMENT DATA ROTATED RIGHT BY ONE BIT AND CALCULATE NO. OF TRANSITIONS
NO
IS NO. OF TRANSITIONS OF DATA ROTATED LEFT BY ONE BIT > (WIDTH OF BUS/2)
YES
BIT COMPLEMENT DATA ROTATED LEFT BY ONE BIT AND CALCULATE NO. OF TRANSITIONS
NO
COMPARE THE NUMBERS OF TRANSITIONS AND FIND THE MINIMUM
OUTPUT: DATA ENCODED IN THE METHOD WITH THE LEAST NO. OF TRANSITONS
STOP

Figure 1. Flow chart describing the proposed technique

- The inputs to the flow chart are the new data (unmodified data) and the data on the bus.
- This unmodified data is rotated right by one bit and rotated left by one. The respective numbers of transitions are calculated by comparing these three types of data with the data on the bus.
- Then we decide whether the number of transitions of the unmodified data is greater than half the width of the bus or not.
- If YES, we invert or bit complement the unmodified data and compute the number of transitions between the inverted data and the data on the bus.

- If NO, then we move on to the next decision block. The output of the above mentioned block is also connected to this decision block.
- In this decision block we decide whether the number of transitions of the data rotated right by one bit is greater than half the width of the bus or not.
- If YES, we invert the data rotated right by one bit and calculate the number of transitions between the data rotated right by one bit and inverted and the data on the bus.
- If NO, then we move on to the next decision block. The output of the above mentioned block is also connected to this decision block.
- In this decision block we decide whether the number of transitions of the data rotated left by one bit is greater than half the width of the bus or not.
- If YES, we invert the data rotated left by one bit and calculate the number of transitions between the data rotated left by one bit and inverted and the data on the bus.
- If NO, then we move on to the next block. The output of the above mentioned block is also connected to this block.
- For any given case we need to compare only three numbers of transitions because they are computed conditionally.

3.3 Basic hardware model

Figure 2. shows the basic hardware model for the proposed method.

The first set of blocks in the above diagram indicates the six methods involved in the technique. The common input to these blocks is the n-bit, un-encoded new input data. The output of these blocks is the respective encoded n-bit data.

The blocks invert, rotate right and invert and rotate left and invert have an extra input which is the number of transitions of unmodified data, data rotate right and data rotated left respectively. In these blocks the given number of transitions is compared to half the bus width. If it is greater than half the bus width then the data is inverted. Otherwise it is not inverted.

The XOR-ADD blocks compute the number of transitions between the encoded data and the data on the bus. The inputs to theses blocks are the n-bit data on the bus, the 3-bit representation of the different methods and the encoded data. The data on the bus and the encoded data are XORed. The result of this XOR operation will also be an n-bit data. The numbers of ones (high) in the output of the XOR operation gives the number of transitions. Hence an adder counts the number of ones in the output of the XOR operation. The outputs of these blocks are the encoded data, its three bit representation and its number of transitions.

The comparator compares the number of transitions of the different methods and finds out the least among them. Since we are conditionally computing the number of transitions, for any case the comparator has to compare only three values. The inputs to this block are the encoded data along with its bit representation and its corresponding number of transitions. The output of this block is the encoded data with the least number of transitions and its 3-bit representation. This encoded data is sent over the bus along with its 3-bit representation.

At the receiver, the 3-bit representation identifies the method used to encode the data and the data is decoded accordingly.

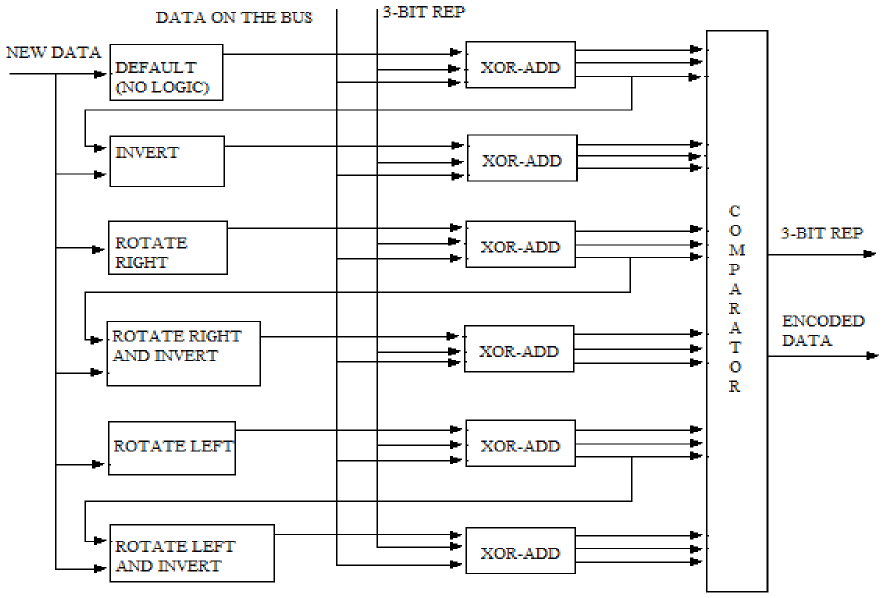

Figure 2. Basic hardware model for the proposed technique

We have coded the hardware using VHDL and a simulated using Modelsim for functionality verification. The number of transitions are counted and compared to select the coding that gives minimum number of switching. According to the type of encoding method that is selected an unique three bit identifier is generated.

4. RESULT ANALYSIS

The proposed algorithm has been implemented using C++ and numbers of transitions were verified. It is found that they are reduced considerably.

Assuming an 8-bit data 01010011 on the bus, Table 2. illustrates how the proposed technique reduces the number of transitions compared to the existing techniques.

Table 2. Illustration of the working principle of the proposed technique

Operation	Modified Data	No. of transitions
New Data on the bus	01011100	4
Data Inverted	10100011	4
Data Rotated right	00101110	6
Data Rotated left	10111000	6
Data Rotated right and inverted	11010001	2
Data Rotated left and inverted	01000111	2

In this case data encoded using any one of the last two methods is sent on the bus since the number of transitions is two which is the least.

Using C++ the complete binary sequence for 4-bit, 6-bit, 7-bit and 8-bit has been generated. Exhaustive checking has been done, that is, every possible combination of two numbers taken from the respective sequences has been checked.

Bus-invert[1] and Shift-Inv[9] are the existing methods which are used on random data. The algorithms of these existing methods have also been implemented in C++. The proposed method has been compared with these existing methods.

To compare the methods we have calculated the percentage of reduced switching activity based on the ratio of number of switching after encoding to the number of switching in the unmodified data.

Table 3. shows the percentage of reduced switching activity of the two methods for data of varying bus widths.

Table 3. Percentage reduction in transitions for various bus widths in different bus encoding styles.

Bus width	Bus-Invert method	Shift-Inv method	Proposed method
4-bit	37.5%	46.9%	69%
6-bit	31.25%	40.6%	57%
7-bit	31.25%	41.02%	52.13%
8-bit	27.34%	36.11%	48.89%

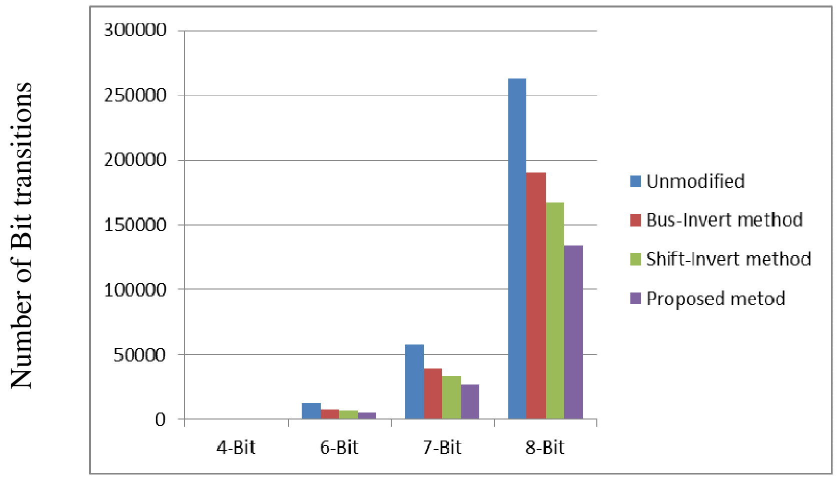

Figure 3. Number of bit transitions for various bus widths different bus encoding styles.

As we can see the percentage of reduced switching activity of the proposed method is much higher than the existing methods. Hence the proposed method is a lot more power efficient than the existing methods.

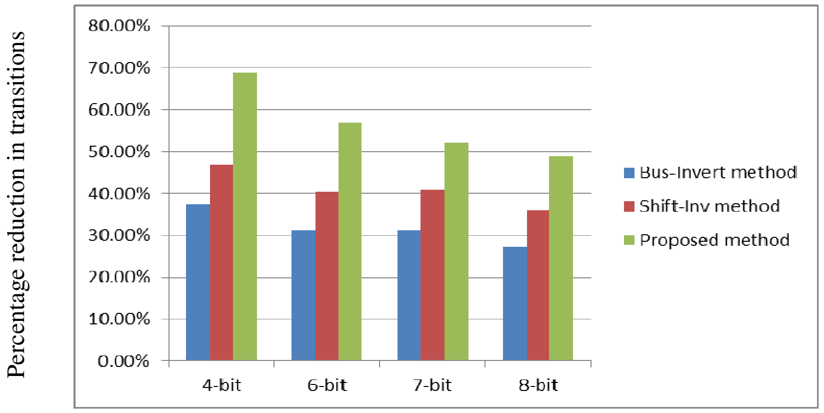

Figure 4. Percentage reduction in transitions for various bus widths in different bus encoding styles.

5. CONCLUSION

The proposed technique demonstrates significant reduction in switching activity irrespective of the bus width, and is therefore more power efficient compared to the similar existing schemes such as Bus-Invert and Shift-Invert techniques. The technique can be used for buses of any width and it requires only three extra bits irrespective of the width of the bus. The proposed method does not require extra computation and hardware overheads when compared to the existing methods. As shown in the flow chart, the technique conditionally encodes the data and hence its usage of the hardware is optimized. The reduction in the number of transitions and hence the reduction in dynamic power by this method is very significant.

While retaining the merit of the proposed method – reduction in switching activity – we need to explore the possibility for reduction of hardware overhead. We have plans to implement area and power efficient circuit for the encoder and decoder.

The technique proposed consists of six options for reducing the number of transitions on the bus. We can also add T0 coding to make it truly universal. It can be done without adding extra bit for identification although requires some extra hardware.

REFERENCES

[1] M. R. Stan and W. P. Burleson, “Bus-Invert coding for low-power I/O”, IEEE Trans. on VLSI, vol. 3, pp. 49-58, March 1995.

[2] Youngsoo Shin, Soo-Ik Chae, and Kiyoung Choi, “Partial Bus-Invert Coding for Power optimization of Application-Specific Systems”, IEEE Trans. on VLSI, vol. 9, pp. 377-383, April 2001.

[3] Jun Yang, Rajiv Gupta, Chuanjun Zhang, “Frequent Value Encoding for Low Power Data Buses”, ACM Transactions on Design Automation of Electronic Systems, Vol. 9, No. 3, July 2004, Pages 354–384.

[4] C. L. Su, C. Y. Tsui, and 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, 1994.

[5] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, “Asymptotic zero-transition activity encoding for address buses in low-power microprocessor-based systems”, Great Lakes VLSI Symposium, pp. 77-82, Urbana IL, March 1997.

[6] Y. Aghaghiri, F. Fallah, M. Pedram, “Irredundant Address Bus Encoding for Low Power,” International Symposium on Low Power Electronics and Design ’01, pp. 182–187, 2001.

[7] Tsung-Hsi Weng, Wei-Hao Chiao, Jean Jyh-Jiun Shann, Chung-Ping Chung, Jimmy Lu, "Low-Power Data Address Bus Encoding Method".

[8] W. Fornaciari, M. Polentarutti, D.Sciuto, and C. Silvano, "Power Optimization of System-Level Address Buses Based on Software Profiling," CODES, pp. 29-33, 2000.

[9] Jayapreetha Natesan and Damu Radhakrishnan, "A Novel Bus Encoding Technique for Low Power VLSI".

[10] "Limited-weight codes for low-power I/O," in Proc. Int.Workshop Low-Power Design, Napa, CA, Apr. 1994, pp. 209–214.

[11] Youngsoo Shin, Kiyoung Choi, and Young-Hoon Chang, "Narrow Bus Encoding for Low-Power DSP Systems", IEEE Trans. on VLSI, vol. 9,no.5, pp. 656-660, October 2001.

[12] Ahmed Elkammar, Srinivasa Vemuru, Norman Scheinberg, "A Bus Encoding Scheme to Reduce Power Consuming Signal Transitions".

[13] S. Komatsu, M. Ikeda, K. Asada, "Bus Data Encoding with Couplingdriven Adaptive Code-book Method for Low Power Data Transmission," 27th European Solid-State Circuits Conference, pp. 312-315, Sep. 2001.

[14] Satoshi Komatsu Masahiro Fujita, "Irredundant Address Bus Encoding Techniques based on Adaptive Codebooks for Low Power".

[15] L. Benini, G. De Micheli, E. Macii, M. Poncino, and S. Quer, "System-level power optimization of special purpose applications: The beach solution", Proc. Int. Symp. Low Power Electronics Design, pp. 24-29, August 1997.

Authors

Shnakaranarayana Bhat M obtained his M.Tech from IISc, Bangalore and is currently working as Associate professor in the department of Electronics and communication Engineering, M.I.T Manipal. In addition to contributing as resource person for technical workshops and conferences, he is also a recognised Faculty trainer and successfully conducted faculty training in various Engineering colleges in India. He has chaired many national and International conferences and his interests include Low Power VLSI Design and Processor architecture in addition to Engineering Education and Soft skills.

D. Yogitha Jahnavi has completed her Bachelor of Engineering (Electronics and communications) degree from Manipal Institute of Technology whichis a constituent institute of Manipal University, in May 2012. She is currently working in Cybage Software Pvt. Ltd.