Journal of Computing and Electronic Information Management

ISSN: 2413-1660

A Low Bit Switching Activity Algorithm for Data Bus

Mingquan Zhang
School of Control and Computer Engineering, North China Electric Power University, Baoding 071003, China

Abstract: With the reduction of the technology size the bus dynamic energy consumption, which is brought by the coupling switching activity (SA) is increasing, and the effect of one single bus encoding on the bus energy saving is not significant. To settle the problems, the author proposed a method named LBSA (Low Bit Switching Activity) algorithm that reduces the total number of bit switching activity, which is based on bit SA perception, and four bus encoding schemes are introduced. In the algorithm, the number of bit SA in each encoding scheme is perceived, and the encoding scheme with the minimum SA number is automatically selected to encode the value to be transferred. The simulation results show that the method can effectively reduce bit SA of on-chip data bus.

Keywords: Bus energy, bit switching activity, data bus.

1. Introduction

Switching activity (SA) on data transmission gives the number of times the bus lines are discharged and recharged between 0 and 1, and is directly responsible for the dynamic energy consumption on the data bus. Power consumption for on-chip driving can reach up to 30% of the total chip power, where the bit SA is the most dominant factor [1]. Therefore, the reduction of bus SA has a large impact on the reduction of the system energy consumption.

The low power encoding techniques [2, 3] are widely used to reduce SA for dynamic energy consumption and the effects of crosstalk (signal noise, delay) during data transmission on buses. They aim to transform the data being transmitted on buses in such a manner so that the self-SA and coupling SA on buses are reduced. The bus encoding schemes can be classified three types [4]: Algebraic encoding, which refers to the transformation of the original code into other forms of code, such as bus invert encoding; Permutation encoding, which is the permutation of the original code; Probability encoding, which is using statistical analysis of the program, getting the

probability of continuous data or instructions, then encoding the data or instruction with high probability to make the number of SA as small as possible.

With the rapid increase in the complexity and speed of integrated circuits and the popularity of portable embedded systems, power consumption has become a critical design criterion [5]. In today's processors, a large number of data needs to be sent through high-speed data bus. Compared to a general-purpose high-performance processor, an embedded processor has much fewer transistors integrated on the chip. Therefore, the amount of the energy dissipated on the data bus of an embedded processor is significant when compared with the total energy consumption of the general-purpose processor. On the other hand, there is a lot of value locality of the data for transmission on the bus. Consequently, low-power bus encoding techniques are often more attractive in the context of embedded processors. It is desirable to encode the values sent over these buses to decrease the SA and thereby reducing the bus energy consumption. An encoder on the sender side does this encoding, whereas a decoder on the receiver side is required to restore the original values.

In this paper, we introduce low overhead encoding methods targeting on-chip data bus. Our methods are irredundant, meaning that they do not require any additional lines to be added to the bus.

2. Dynamic energy model and SA distance

The DSM bus capacitance model is shown in Fig. 1, where C_L is the self-capacitance between a bit line and ground, and C_I is the coupling capacitance between adjacent bit lines. The capacitance factor λ is defined as the ratio of the coupling capacitance to the self-capacitance; that is, λ= CI/CL, it highly depends on the manufacturing and layout details. The value of λ becomes high when the technology shrinks. As the technology continuing scaled-down, the value of λ will become larger in the future [6]. In this paper, we investigate the bus with 70nm technology and the λ is about 5.

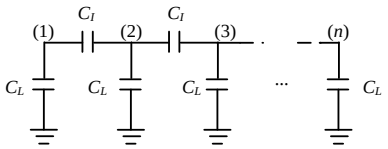

Fig.1 Capacitance model of a DSM

As mentioned earlier SA makes the bus lines discharge and recharge between 0 and 1, and is responsible for the bus dynamic energy consumption. In this paper we mainly use the value optimization of data transmission to optimize the on-chip bus data dynamic energy consumption. The SA, which is the source of dynamic energy consumption, mainly consists of two parts: the self-SA (caused by self-capacitance) and the coupling SA (caused by coupling capacitance). The on-chip data bus dynamic energy consumption E can be calculated by (1). Equations (2) and (3) describe the

energy consumption due to self-SA (E_S) and coupling SA (E_I), respectively. In (2), X (named vertical distance) is the corresponding total number of self-SA, which can be calculated by summing up $X_{i,j}$. Thus, X is given by: $X = \sum_{i=1}^{n}\sum_{j=1}^{t-1} X_{i,j}$,where $X_{i,j}$ represents the SA for bit line i from cycle j to cycle j + 1. It is equal to one if there is a 0 to 1 switching, otherwise, $X_{i,j}$ is equal to zero; t is the total number of clock cycles needed for the data transmission, n is the bit-width of the bus and VDD is the supply voltage. In (3), Y (named horizontal distance) is the total number of coupling SA, which can be calculated by summing up $Y_{(i,i+1),j}$. Thus, Y is given by: $Y = \sum_{i=1}^{n-1}\sum_{j=1}^{t-1} Y_{(i,i+1),j}$, where $Y_{(i,i+1),j}$ represents the coupling SA from cycle j to cycle j + 1 between bit line i and its adjacent bit line i+1. We use the similar coupling bus model employed in [14], which is summarized in Table 1.

$$E = E_S + E_I \tag{1}$$

$$E_S = C_L \cdot V_{DD}^2 \cdot X \tag{2}$$

$$E_I = C_L \cdot V_{DD}^2 \cdot Y \tag{3}$$

Put (2), (3) and $\lambda = C_I/C_L$ into (1), we can get

$$E = (X + \lambda \cdot Y) \cdot C_L \cdot V_{DD}^2 \tag{4}$$

We denote D_{enc} is the total SA distance and D_{enc} is given by

$$D_{enc} = X + \lambda \cdot Y \tag{5}$$

From (4) and (5), to obtain a minimum value of E, we just make the value of D_{enc} minimum.

Table1. The value of Y(i,i+1)

Y(i,i+1)	bit line i at time j->j+1				
bit line i+1 at time j->j+1	SA	0->0	0->1	1->0	1->1
	0->0	0	1	0	0
	0->1	1	0	2	0
	1->0	0	2	0	1
	1->1	0	0	1	0

3. The Proposed LBSA algorithm

3.1 LBSA encoding algorithm

We denote B^j is the value to be sent on the n-bit width data bus at cycle j, and it can be expressed as $B^j = (b_n^j, b_{n-1}^j, b_{n-2}^j, \cdots, b_1^j)$, $B^{(j-1)enc}$ represents the encoded value at cycle j-1. Four encoding schemes of the scheme are expressed as: Swap $B^{(swap)}$, Invert

$B^{(inv)}$, Odd Invert $B^{(odd)}$, Even Invert $B^{(even)}$ with the coding number 00, 11, 01, 10, respectively. Swap the adjacent bits of B^i and append its coding number, we get the encoded value $B^{i(\text{swap})}$.

The encoded value that all the bits of B^i are inverted then appended its coding number is defined as $B^{i(inv)}$. The encoded value that the odd bits of B^i are inverted then appended its coding number is defined as $B^{i(odd)}$. The encoded value that the even bits of B^i are inverted then appended its coding number is defined as $B^{i(even)}$. The coding number indicates which encoding scheme is used to encode the sending value. The function ST_n(d1; d2) is used to calculate the vertical distance X between the two n-bit width values, the function CT_n(data) is used to calculate the horizontal distance Y of n-bit width value, and D_{total} represents the minimum total SA distance of the encoded value.

Algorithm 1 is LBSA encoding algorithm, which demonstrates how the encoder selects the encoding scheme with minimum total SA distance according to (5) and returns the encoded data value and its coding number. The input value with n-bit width is entered the four encoding components at the same time and they produce four types (n+2)-bit encoded values. In addition, subsequently, the encoded values are sent to distance estimator, in which obtained their total SA distance, respectively. In order to reduce the delay, each encoding process and distance evaluation are treated with at the same time. Finally, the distance comparator is responsible for selecting the encoded value $B^{i(enc)}$ which has the minimum SA total distance and producing coding number codenum.

Algorithm 1 LBSA Encoding Algorithm

Input: *value , n*;

Output: *encoded_value, codenum*.

1. $B^{(\text{swap})}$ ←swap(*value*);//Swap code ;
2. $B^{(\text{inv})}$ ← invert(*value*);//Invert code;
3. $B^{(\text{odd})}$ ← odd invert(*value*);// Odd invert code;
4. $B^{(\text{even})}$←even invert(*value*);//Even invert code ;
5. $D_{\text{swap}} \leftarrow \text{ST}_{-n}(B^{(\text{swap})}, B^{(j-1)\text{enc}}) + \lambda * \text{CT}_{-n}(B^{(\text{swap})})$;
6. $D_{\text{inv}} \leftarrow \text{ST}_{-n}(B^{(\text{inv})}, B^{(j-1)\text{enc}}) + \lambda * \text{CT}_{-n}(B^{(\text{inv})})$;
7. $D_{\text{odd}} \leftarrow \text{ST}_{-n}(B^{(\text{odd})}, B^{(j-1)\text{enc}}) + \lambda * \text{CT}_{-n}(B^{(\text{odd})})$;
8. $D_{\text{even}} \leftarrow \text{ST}_{-n}(B^{(\text{even})}, B^{(j-1)\text{enc}}) + \lambda * \text{CT}_{-n}(B^{(\text{even})})$;
9. $D_{total} \leftarrow \text{Min}(D_{\text{swap}}, D_{\text{inv}}, D_{\text{odd}}, D_{\text{even}})$;
10. IF $D_{total} == D_{\text{swap}}$ THEN
11. *encoded_value* $= B^{(\text{swap})}$;
12. *codenum*←00;
13. END IF
14 IF $D_{total} == D_{\text{odd}}$ THEN

15. *encoded_value* $=B^{(\mathrm{odd})}$;
16. *codenum*←01;
17. END IF
18. IF $D_{total} == D_{\mathrm{even}}$ THEN
19. *encoded_value* $=B^{(\mathrm{even})}$;
20. *codenum*←10;
21. END IF
22. IF $D_{total} == D_{\mathrm{inv}}$ THEN
23. *encoded_value* $=B^{(\mathrm{inv})}$;
24. *codenum*←11;
25. END IF
26. RETURN *encoded_value*, *codenum*.

3.2 LBSA decoding algorithm

Algorithm 2 is SAPME decoding algorithm, which demonstrates how the decoder returns the original value using coding number. The input of the 2-4 decoder is the control signal coding numbers a and b. The output of the 2-4 decoder produces one of the four decoding signals controlling one following decoding component to be valid. The valid component decodes to obtain the *n*-bit original value B^i. The four decoding components are: Swapper for swapping adjacent bits of the encoded value $B^{i(swap)}$, Inverter for inverting all the bits of the encoded value $B^{i(inv),}$ Odd inverter for inverting all the odd bits of the encoded value $B^{i(odd)}$ and Even inverter for inverting all the even bits of the encoded value $B^{i(even)}$.

Algorithm 2 LBSA Decoding Algorithm

Input: encoded_value, codenum;
Output: original_value.
1. IF *codenum*==00 THEN
2. original_value <-- swap(encoded_*value*);
3. END IF
4. IF *codenum* ==01 THEN
5. original_value <-- odd invert(encoded_*value*);
6. END IF
7. IF *codenum* ==10 THEN
8. original_value <-- even invert(encoded_*value*);
9. END IF
10. IF *codenum* ==11 THEN
11. original_value <-- invert(encoded_*value*);
12. END IF
13. RETURN original_value.

4. Experiment and Results Analysis

Our multi-core structure based on bus is four-core CMP structure, each core has 4-way set associative private first level of instruction cache (IL1) and data cache (DL1) with a size of 32KB, and 16-way shared second level cache (L2) with a size of 1MB. The cache row size at all levels is 64B, and it is included cache. The specific parameters are shown in Table 2. Each of the private L1 split caches is write-through. The shared L2 cache is write-back and maintains inclusion with respect to the L1 cache. In our multi-core system, the cores and L2 cache are connected by bus. The data bus is alternately used by different cores, so as to achieve the purpose of access the shared L2 cache.

To verify the efficiency of bus energy saving, some of the parameters are adjusted in the experiments. We select three storage intensive programs from Olden[7] and CPU2006[8] for performance evaluation benchmarks: mst_ht, em3d_ht and 429.mcf_ht. The programs are cross-compiled with GCC for MIPSII executable files. In order to reduce the impact of code optimization on application performance during program design, and make the program reach its peak as far as possible when the program runs, the GCC optimization option is the best optimized -O3.

Table 2. The specific parameters

parameters	value
process technology	70nm
cores	4
IL1/DL1 size	32KB
L1 associativity	4-way
bus width	32+2 lines
bus energy/access	11.6pJ/line
Coupling factor λ	5

After the different measures are adopted, the number of SA on the bus lines are effectively reduced. Fig.2 shows the reduced ratio of the number of bit SA with different measures to the number of original bit SA on the bus. SWAP, ODD, EVEN, INV indicates the reduced ratio with using the SWAP coding, ODD invert, EVEN invert and INVERT coding alone, respectively. LBSW indicates the reduced ratio with using the LBSW algorithm. The number of bit SA is reduced by an average of about 1.1%, 5.9%, 6.0%, 7.2% by using SWAP, ODD invert, EVEN invert and INVERT coding alone, respectively. When the LBSW algorithm is used the reduced ratio of SA is about 11.6%. This is because the LBSA algorithm can perceive the number of bit SA in each encoding scheme, and the encoding scheme with the minimum SA number is automatically selected to encode the value to be transferred. As is known to all, bit SA directly

determines the on-chip bus energy consumption. After using our method, the number of bit switching activity of the benchmarks is reduced in varying degrees, which brings advantages to bus energy saving.

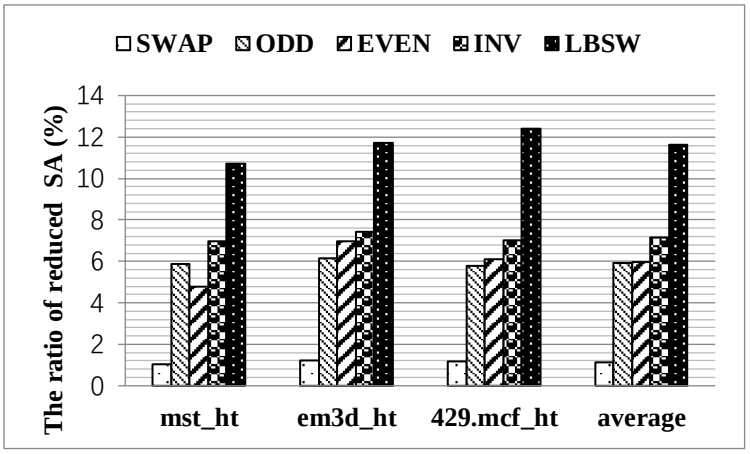

Fig.2 The ratio of reduced bus SA with different measures

5. Conclusion

In this paper we proposed an algorithm to reduce the bus SA, which efficiently reduces the number of bus SA. The LBSA algorithm is particularly suitable for memory intensive special-purpose applications. However, the method is general enough to be used in other types of buses. Experimental results show that the proposed algorithm is efficient.

Acknowledgements

This paper was financially supported by "the Fundamental Research Funds for the Central Universities (2018MS073)".

References

[1]Chiu, Ching-Te, Huang, Wen-Chih, Lin, Chih-Hsing, et al. Embedded transition inversion coding with low switching activity for serial links [J]. IEEE Transactions on Very Large Scale Integration (VlSI) Systems, 2013, Vol. 21(10):p1797-1810

[2]Sankaran, H.,Katkoori, S. Simultaneous Scheduling, Allocation, Binding, Re-Ordering, and Encoding for Crosstalk Pattern Minimization During High-Level Synthesis[J]. IEEE Transactions on Very Large Scale Integration Systems, 2011, Vol. 19(2):p217-226

[3]Kaushik, Brajesh Kumar, Agarwal, Deepika,Babu, Nagendra G. Bus encoder design for reduced crosstalk, power and area in coupled VLSI interconnects[J]. Microelectronics Journal, 2013, Vol. 44(9):p827-833

[4]Verma S. K., Kaushik B. K., Novel bus encoding scheme for RC, coupled VLSI interconnects, Trends in Network and Communications, Springer Berlin Heidelberg,2011, 197, p435-444

[5]Zhang M, Gan Z, Zhang J, et al. "Joint Hybrid Frequent Value Cache and Multi-Coding for Data Bus Energy Saving", IEEE, International Conference on Parallel and Distributed Systems. IEEE, 2017, p869-876

[6]International Technology Roadmap for Semiconductors. http://www.itrs.net.

[7]Rogers, A., Carlisle, M. C., Reppy, J. H., Hendren, L. J., Supporting dynamic datastructures on distributed-memory machines[J], ACM Transactions on Programming Languagesand Systems,1995, Vol. 17(2), 233-263
[8]Henning, John L., SPEC CPU2006 benchmark descriptions [J], SIGARCH Computer Architecture News,2006, Vol. 34(4), p1-17