Bus Encoding for Simultaneous Delay and Energy Optimization

Jingyi Zhang, Qing Wu, Qinru Qiu
Department of Electrical and Computer Engineering
Binghamton University, State University of New York
Binghamton, New York – 13902, U.S.A
{jzhang5, qwu, qqiu}@binghamton.edu

ABSTRACT
In this paper we propose two bus encoding algorithms that optimize both bus delay and energy dissipation based on the probabilistic characteristics of data on data buses. The first algorithm minimizes the crosstalk transitions by inserting temporal redundancy and achieves optimal energy. The second algorithm reduces crosstalk more aggressively to achieve optimal bus delay by mapping the original data to low-energy opposite-transition-forbidden codes. Experimental results show that they outperform the existing heuristic bus encoding algorithms by 15.7% to 58.8% in average energy dissipation and 11.4% to 58.4% in average delay.

Categories and Subject Descriptors
B.7.1 [**Integrated circuits**]: Integrated Circuits

General Terms: Algorithms, Design, Performance.

Keywords: Adaptive bus encoding, coupling capacitance, data probability distribution peaking, delay optimization, energy optimization, opposite transition forbidden, temporal/spatial redundancy.

1. INTRODUCTION
As technology scales down to deep submicron, the crosstalk energy becomes the major component in bus energy dissipation [1][2]. The bus delay becomes data pattern dependent because of the crosstalk induced delay. For example, opposite transitions (also called "2λ" transitions) on adjacent wires produce especially large propagation delay. Extensive research works have been done for the optimization of bus energy dissipated on the coupling capacitances [1][4]. Nowadays, more and more attentions have been paid on minimizing bus delay [2][3][5][7][8]. However, rarely any work effectively optimizes the delay and energy at the same time. Although reducing the crosstalk effect is the key for both optimizations, as we will show later in the paper, the bus energy dissipation is determined by the overall crosstalk effect across the bus lines while the bus delay is determined by the worst case local crosstalk effect. A bus encoding scheme for minimum delay does not always lead to minimum bus energy and vice versa.

M. Mutyam et al. [8] proposed an approach that eliminates certain types of crosstalk by inserting temporal redundancy while using variable cycle transmission. The performance can be speeded up; meanwhile moderate energy reduction can be achieved. K. Sainarayanan et al. [5] proposed a method that applies bus inverter on every subgroup of bus, and inserts temporal redundancy to indicate the status of bus inverting. The method targets at reducing the worst case bus delay and the amount of improvement depend on the number of shielding lines inserted. N. Satyanarayana et al. [10] proposed two delay minimization techniques by exploiting the similarity of the upper half data, thus the performances highly depend on the data behaviors. They can't guarantee to eliminate the delay-expensive crosstalk type. C. Duan et al. [2] presented a scheme to eliminate part of the high-energy and delay-expensive code patterns. B. victor et al. [3] proposed a one-to-one mapping scheme based on a codebook that doesn't allow opposite direction transitions on adjacent lines. The codebook is obtained by searching for the largest prime clique of either one of the two codewords with alternating 0 and 1. Since the largest clique problem is NP-hard, the calculation of the codebook is computationally intensive for wide bus width. L. Li et al. [7] proposed a variable cycle transmission technique, where a crosstalk analyzer is implemented to dynamically tunes the length of bus clock cycle time. This approach does not try to reduce the crosstalk effect, hence it has no effect on energy dissipation and it does not reduce the worst case delay.

To trade off between the circuit performance and the energy consumption, we propose two novel delay and energy efficient bus coding schemes. The first algorithm exploits both spatial and temporal redundancy to minimize average energy and average delay. The second one generates a one-to-one mapping from the original data to the weighted opposite-transition-forbidden code, which is decided by the probability distribution of the original data. Both are based on weighted code mapping (WCM) [9].

The remaining paper is organized as follows. Section 2 introduces the background on the analytical delay and energy model for DSM bus. In section 3, the data probability distribution peaking method to optimize the performance of the WCM algorithm is introduced. Then the proposed delay and energy efficient coding algorithms are presented. Section 4 presents and analyzes the experimental results in detail. Finally we draw conclusions in section 5.

2. BACKGROUND
The analytical models for the propagation delay [6] and energy consumption [4] in deep sub-micron buses were proposed by Sotiriadis et al. Let C_I be the total inter-line capacitance, while C_L be the line-to-ground capacitance. Let λ be the *capacitance factor* which is calculated as $\lambda = C_I / C_L$. The crosstalks are categorized

ISLPED'08, August 11–13, 2008, Bangalore, India.

into six classes by their corresponding delay at the intermediate lines, as shown in Table 1. The symbols ↑, ↓, − are used to indicate 0→1, 1→0 and 1→1 (or) 0→0 bit transitions respectively. It is beneficial from delay perspective to minimize the occurrence or even remove crosstalk class 4, 5 and 6 transitions. However, energy dose not necessarily change in the same trend as delay. For example, transition from 000 to 101 (↑−↑) won't cause any delay at the middle line, but will introduce in $C_L R_T \lambda$ energy, because the first charging line and the middle unchanging line form a conducting path for the inter capacitor between them. Therefore it is important for bus encoding schemes to take both delay and energy into consideration.

Table 1. Transition patterns and delay.

CC	Delay	Transition Patterns
1	0	↑−↑, ↓−↓, ↑−↓, ↓−↑, ↑− −, − −↓, − − −, − −↑, ↓− −
2	$C_L R_T$	↑↑↑, ↓↓↓
3	$C_L R_T(1+\lambda)$	↑↑−, ↓↓−, −↑↑, −↓↓
4	$C_L R_T(1+2\lambda)$	−↑−, −↓−, ↓↓↑, ↑↑↓, ↓↑↑, ↑↓↓
5	$C_L R_T(1+3\lambda)$	−↓↑, −↑↓, ↑↓−, ↓↑−
6	$C_L R_T(1+4\lambda)$	↓↑↓, ↑↓↑

3. ENERGY AND DELAY EFFICIENT ALGORITHMS

3.1 Probability Distribution Peaking

The optimality of the WCM algorithm relies on the information of the probability distribution of the data stream because it maps the data with higher probability to the code with smaller *ivs*[9]. Therefore, the data set with probability distributions that have sharper "peak" has higher potential for energy saving. In probability theory, the "peakedness" of the probability can be measured by kurtosis. Data distribution with higher kurtosis has a sharper "peak", that is, a higher probability of values near its mean. It is observed that exclusive-or (XOR) function is able to skew most data distributions toward zero. Thus, performing XOR at adjacent vectors of the input data may be the easiest way to effectively sharpen the peak of the probability distribution. An *m*-bit-wide bus needs *m* XOR gates at the encoder side; the original data can be recovered at the decoder side by performing the same XOR operation. We performed experiments on WCM algorithm with various multimedia and random distributed data benchmarks. The results show that xor-ed data have an average improvement of 27% in energy and 26% in delay over the original data.

3.2 Variable Cycle Transmission with Hybrid Spatial Temporal Redundancy

Even though the WCM algorithm works effectively in reducing energy consumption, there still are "2λ" transitions, leading to more energy dissipation as well as larger delay. We designed a hybrid bus encoding algorithm that applies WCM and also inserts temporal redundant pattern of all-ones or all-zeros between the old and the new codewords. This technique is referred to as Hybrid Spatial Temporal Redundancy encoding (HST).

Let $D(i, j)$ present the transition delay from the code i to the new code j. It is observed that the sum of $D(w_i,0)$ and $D(0,w_j)$ (referred as *sum_zeros*), or the sum of $D(w_i,(2^m-1))$ and $D((2^m-1),w_j)$ (referred as *sum_ones*), is smaller than or equal to the original entry $D(w_i,w_j)$. In another word, inserting a temporal redundant data 0 or 2^m-1 between two neighboring data will result in smaller delay. It is also beneficial to reduce energy consumption, since transitions from or to all-zero vector and all-one vector draws relatively fewer or even no energy from the power supply. The pseudo code of the HST algorithm is given in figure 1.

```
Input: present data d_t, next data d_{t+1} and present code w_t
Output: encoded next data w'_t and w_{t+1}
1. map xored_d_{t+1} to w_{t+1} using Weighted Code Mapping;
2. if (crosstalk type(w_t ,w_{t+1}) > 4) {
3.         if (crosstalk type(w_t ,w_{t+1}) = 6)
4.             w_{t+1} = inverted w_{t+1};
5.         if (min{sum_zeros, sum_ones} < D(w_i, w_j)) {
6.             if (sum_zeros ≤ sum_ones)
7.                 w'_t = 2^m − 1;
8.             else
9.                 w'_t = 0; } }
```

Figure 1. The hybrid spatial temporal algorithm.

We then combine our algorithm with the variable cycle transmission scheme. Same as in the VCT scheme [7], a ready_out signal is added and activated dynamically depending on the transition patterns. Unlike other temporal redundant bus algorithms, our decoder doesn't care whether a temporal redundant code w'_t is inserted or which pattern is inserted. As long as it detects an active ready_out, the original data can be decoded from the current code w_{t+1}, and previous code w_t. The algorithm requires three extra lines: the first one is used to form the WCM codebook, the second one to indicate whether the bus is inverted, and the last is the ready_out signal, disregarding the space used to shield the ready_out, invert indicator and the actual code. Example 1 shows the working of our algorithm.

***Example1*:** Consider m = 8, a = 1, λ = 5. As explained in precious section, the maximum *ivs* of the selected WCM code is 4. Assume there are two data sequentially mapped to WCM code: w_t = 101111000, w_{t+1} = 010001000. The original delay of the transition is 1+4λ. The algorithm detects a crosstalk type 6 transition, so w_{t+1} is first flip-flopped; then the summation of $D(w_t,0)$ and $D(0,w_{t+1})$ is calculated and found to be smaller than the original $D(w_t,w_{t+1})$. After encoding, the actual sequence transmitted on the bus is {101111000, 000000000, 101110111}. The total propagation delay and energy consumption for transmitting the sequence are 12 (i.e. 2+2λ), and 27 respectively, compared to the delay and energy of 21 (i.e. 1+4λ), and 26 for the original WCM algorithm with 9-bit wide bus.

The reason that HST algorithm outstands from many of other works lies in that it only requires a small number of extra bus lines. The property of *ivs* code allows us to reduce the crosstalk 5 and 6 delay to 2(1+λ) for most of the transition patterns, leading to much smaller average delay. Furthermore, our approach encodes data sequence by assigning the higher mapping priority to the code with lower *ivs* achieving significant energy reduction.

3.3 Opposite-Transition-Forbidden Weighted Code Mapping (OTF_WCM)

The original WCM algorithm [9] cannot guarantee to eliminate crosstalk 5 and 6. We proposed the Opposite-Transition-Forbidden Weighted Code Mapping (OTF_WCM) algorithm. The new technique generates a WCM codebook that doesn't have opposite transition patterns, therefore, all crosstalk type 5 and 6 as well as part of crosstalk type 4 are eliminated.

Let W denote the set of codewords of n bits: $W=\{w \mid w = a_1 a_2 \ldots a_i a_{i+1} \ldots a_n, a_i = 0,1\}$, where i denotes the index of the bit line of the codeword.

Definition: An opposite transition forbidden (OTF) codebook W is a set of codewords that don't have "2λ" transitions with each other.

Theorem 1: A codebook W is OTF, if $\{(-1)^i \times a_{i+1} \le (-1)^i \times a_i, \forall i, 1 \le i \le n\}$, $\forall w \in W$

Theorem 2: the number of the k-bit OTF codewords is a Fibonacci number. A closed-form formula for the total number of the valid k-bit codes is given by $\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{k+2} - \left(\frac{1-\sqrt{5}}{2}\right)^{k+2}\right]$. It can be used to calculate the number of required OTF code bits for any bus width.

With the OTF codebook, we are able to eliminate the "2λ" transitions, and hence keep the worst-case delay to 1+2λ. With the WCM mapping, we are able to reduce the average bus energy by maximizing the occurrence of low IVS codewords. Figure 2 gives the pseudo code of the OTF_WCM algorithm. The valid OTF codeword is selected by simply scanning through all the bits and check if it satisfies theorem 1.

```
1. Set codeword array A[2^m]=0;  ivs=0; C=0;  id = 0;
2. While (|A|)<2^m {
3.   if (C== 2* C_{n-1}^{ivs} ) {
4.       ivs++;
5.       C=0;}
6.   Generate the next binary vector w with IVS(w)=ivs;
7  if (w is OTF code) {
8.     A[id++] = w;
9.     C++;} }
10. Sort the input xored_data based on their probability;
11. Map the highest probable data to the code with the smallest IVS;
```

Figure 2. The OTF_WCM algorithm.

Although the idea of utilizing the OTF code is similar as the crosstalk preventing coding method [3] and the forbidden pattern algorithm [2], here we apply a weighted code mapping to maximize the occurrence of low IVS codewords. Therefore, the OTF_WCM outperforms the previous two works in average energy consumption. Furthermore, instead of solving the maximum clique problem, our OTF codebook generation is based on Theorem 1 and it requires only one scan of the codewords. Therefore, it has a much less complexity.

4. EXPERIMENTAL RESULTS

Total of 11 data sequences are tested to evaluate the efficiency of our coding algorithms, 6 of which are artificially generated and the others are extracted from multimedia applications. The artificial data sequences are generated following three different types of random distributions: triangular (T), uniform (U) and normal (N). The "Im8" is an 8-bit sequence with subsequences selected from four different images. The "Au8", and "Vi8" are 8-bit audio and video sequences. Each data sequence is 10-20Mbits long. Twelve different coding algorithms have been compared, which include: the original hybrid WCM (HYB)[9], the HYB with xored-data (HYB_xor), the proposed HST and OTF_WCM algorithms, the original and the modified spatial temporal redundancy algorithms [6] (denoted as ST and MST respectively), the forbidden pattern coding (FP) [2], the crosstalk preventing coding (CPC) [3], the variable cycle transmission algorithm (VCT) [7], the variable cycle transmission with temporal redundancy (VCTR) [8], the data packing (DPack) and data permutation (DPerm) coding methods [10], as well as our proposed algorithms combined with the window-base adaptive scheme [9], where a sliding window is used to find out the probability distribution of the input data (denoted as AHST and AOTF_WCM). The simulation results for 8-bit data benchmarks are shown in Table 2 (see next page). It compares the required bus width for every algorithm (# of lines), the worst-case delay without implementing the VCT technique (worst-case delay), the percentage improvement over un-coded data in average energy (E %) and the percentage improvement over un-coded data in average delay (D %) when the VCT technique is employed. We then compare the percentage improvement of our algorithms over the average of others' works (Average % Improv of AHST and Average % Improv of AOTFW). The last two rows of the table present the percentage improvement in average energy or average delay over the best of others' works, namely the worst-case percentage improvement of AHST (Wst-case % Improv of AHST), and the worst-case percentage improvement of AOTF_WCM (Wst-case % Improv of AOTFW). Table 3 shows the corresponding experimental results for 16-bit data benchmarks.

The results clearly show that AHST has more optimal average energy while AOTF_WCM has more optimal average delay. Both are superior to any of the other works in reducing bus energy. It is also noticed that for random distributed data benchmarks, MST is superior to AHST in reducing average delay. However, when it comes to real application benchmarks, MST can't compete with our algorithms in minimizing delay. Besides, each inserted shielding line for MST method introduces in up to 2λ extra energy. We summarized table 2 and 3 by comparing the average of the results for all the data benchmarks. AHST ends up with 40.91% (8-bit) and 28.19% (16-bit) improvement in energy over the best of the other works. AOTF_WCM shows 21.79% (8-bit) and 12.77% (16-bit) improvement in energy over the best of other works. AOTF_WCM also has 16.7% (8-bit) and 26.42% (16-bit) improvement in bus delay over MST, which is the best of others' works. It is quite intuitive from the results that our AHST and AOTF_WCM methods have much more optimal energy delay product. We employ Synopsis Design Compiler to generate the gate-level net-lists for the encoders and decoders of our HST and OTF_WCM schemes. Table 4 compares the area overhead (measured by the number of NAND gates) of 8-bit and 16-bit bus encoders and decoders. We believe it is tolerable for probability-based bus encoding schemes.

Table 4. Area overhead for the encoder and decoder.

	HST		OTF_WCM	
	Encoder	Decoder	Encoder	Decoder
8-bit codec	1307	904	861	1075
16-bit codec	105012	95104	100047	117100

Table 2. Comparisons of bus encoding algorithms (8-bit bus, λ = 5).

	# of lines	Worst-caes delay	T8		U8		N8		Im8		Au8		Vi8	
			E %	D %	E %	D %	E %	D %	E %	D %	E %	D %	E %	D %
HYB [9]	9	2+4λ	10.97	4.46	10.11	0.47	30.94	23.2	9.39	1.4	21.44	8.69	16.96	15.59
HYB_xor	9	2+4λ	33.18	27.64	19.7	6.9	47.83	42.29	55.09	52.57	52.39	51.16	44.11	45.28
HST	10	2+3λ	43.74	34.46	42.61	32.99	55.79	42.8	50.26	40.99	46.98	38.08	52.86	42.76
AHST	10	2+3λ	44.21	37.51	43.14	36.21	59.53	47.28	73.86	69.28	67.73	60.6	73.84	69.19
OTF_WCM	12	1+2λ	33.85	40.72	32.17	38.4	48.13	49.63	37.89	45.56	39.9	45.12	39.91	47.25
AOTF_WCM	12	1+2λ	38.9	50.59	34.60	50.78	51.63	54.92	51.75	64.52	56.84	64.65	47.5	61.53
ST [6]	10	2+4λ	29.34	18.79	31.06	18.73	36.73	22.76	11.05	4.01	18.81	12.75	17.32	6.44
MST [6]	14	2+2λ	22.81	48	26.72	46.24	41.19	53.74	17.89	44.54	26.65	45.53	22.77	45.94
FP [2]	11	1+3λ	6.12	7.91	4.6	-0.24	14.41	27.09	11.58	41.36	28.38	26.41	20.36	50.05
CPC [3]	12	1+2λ	11.17	22.71	11.03	20.3	29.86	29.15	26.32	24.46	27.97	25.17	27.59	22.97
VCT [7]	8	4+4λ	0	9.01	0	9.48	0	-1.69	0	-5.32	0	-0.17	0	-3.92
VCTR [8]	9	3+3λ	28.2	2.66	28.56	2.82	35.52	3.08	31.4	3.27	32.55	3.06	31.79	3.36
Average % Improv of AHST			33.37	23.62	31.50	23.05	45.10	32.10	68.74	62.21	58.42	51.48	67.31	61.09
Average % Improv of AOTFW			27.02	30.42	21.21	29.38	34.38	41.94	42.31	56.35	44.38	56.47	34.40	51.42
Wst-case % Improv of AHST			21.05	-20.1	17.52	-18.6	31.18	-13.9	61.89	44.61	52.16	27.66	61.65	38.32
Wst-case % Improv of AOTFW			13.52	4.97	10.75	-1.45	17.74	2.54	29.67	36.03	36.01	35.11	23.04	22.99

Table 3. Comparisons of bus encoding algorithms (16-bit bus, λ = 5).

	# of lines	Worst-caes delay	T16		U16		N16		Vi16		Im16	
			E %	D %	E %	D %	E %	D %	E %	D %	E %	D %
HST	18	1+4λ	45.47	28.15	43.51	27.06	49.70	29.87	41.47	31.82	45.51	31.41
AHST	18	1+4λ	58.54	41.89	55.35	37.17	60.27	43.58	57.32	57.40	65.86	48.56
OTF_WCM	23	1+2λ	38.46	48.46	36.30	47.84	41.29	50.82	33.98	52.20	31.45	53.26
AOTF_WCM	23	1+2λ	53.64	59.31	49.16	56.55	53.05	62.08	46.32	71.33	48.40	61.84
ST [6]	21	2+4λ	49.40	30.50	47.25	29.43	48.54	28.18	37.59	24.98	27.23	14.80
MST [6]	31	2+2λ	45.91	50.57	43.66	49.54	44.84	50.06	33.08	49.44	21.72	44.76
FP [2]	23	1+3λ	8.06	26.40	4.97	7.98	1.27	11.38	9.89	10.91	22.73	13.51
CPC [3]	23	1+2λ	25.78	34.54	23.47	33.82	27.84	31.45	18.72	31.89	25.47	21.09
VCT [7]	16	4+4λ	0.00	-6.21	0.00	-5.12	0.00	-2.52	0.00	-6.31	0.00	-14.2
VCTR [8]	17	3+3λ	33.51	4.76	31.43	4.69	31.09	4.09	33.89	3.02	31.33	-9.86
DPack [10]	18	1+4 λ	0.19	0.18	0.17	0.18	0.17	0.19	0.03	0.00	-0.12	-12.2
DPerm [10]	22	1+4 λ	2.84	1.57	-0.22	-0.30	-2.90	-2.64	2.17	4.67	0.08	5.77
Average % Improv of AHST			47.71	29.33	44.99	26.04	51.04	33.60	48.62	50.00	59.33	44.11
Average % Improv of AOTFW			41.53	50.51	37.36	48.85	42.14	55.37	35.37	66.36	38.53	58.53
Wst-case % Improv of AHST			18.06	-17.5	15.36	-24.5	22.81	-12.9	31.61	15.73	53.09	6.87
Wst-case % Improv of AOTFW			8.38	17.68	3.61	13.89	8.78	24.06	13.98	43.30	29.09	33.19

5. CONCLUSIONS

In this paper, we have proposed two bus encoding algorithms that reduce bus delay and minimize energy consumption. The first one minimizes the occurrence of type 5 and 6 crosstalk by inserting temporal redundancy, and minimizes the occurrence of energy-expensive crosstalk by assigning the higher mapping priority to the code with lower *IWS*. The second one selects the codebook that forbids "2λ" transitions to reduce the worst case delay, meanwhile, minimizes the expected bus energy.

6. REFERENCES

[1] S. R. Sridhara, A. Ahmed, and N. R. Shanbhag, Area and Energy-Efficient Crosstalk Avoidance Codes for On-Chip Buses, Proc. Of IEEE International Conference on Computer Design, 2004.

[2] C. Duan and A. Tirumala, Analysis and Avoidance of Cross-talk in On-Chip Buses, Hot Interconnects 9, pp. 133-138, Aug. 2001.

[3] B. Victor and K. Keutzer, Bus Encoding to Prevent Crosstalk Delay, IEEE/ACM International Conference on Computer Aided Design, 2001.

[4] Sotiriadis, P., A. P. Chandrakasan, Bus Energy Reduction by Transition Pattern Coding Using a Detailed Deep Submicrometer Bus Model, IEEE Transactions on Circuits and Systems, pp. 1280-1295, October 2003.

[5] K. S. Sainarayanan, C. Raghunandan, M. B. Srinivas, Bus-encoding schemes for minimizing delay in VLSI interconnects, Proc. of Integrated circuits and systems design, pp. 184-189, 2007.

[6] Sotiriadis P, Chandrakasan A. P., Reducing bus delay in submicron technology using coding, In Proc. of IEEE Conf. ASPDAC'01, pp 109-114, 2000.

[7] L. Li et al., A Crosstalk Aware Interconnect with Variable Cycle Transmission, In DATE, 2004, pp. 102-107.

[8] Madhu et al., Delay and Energy-Efficient Data Transmission for On-chip Buses, In ISVLSI'06, pp355-360, 2006.

[9] A. Brahmbhatt, J. Zhang, Q. Wu, and Q. Qiu, Low-power bus encoding using an adaptive hybrid algorithm, Proceedings of the 43rd annual conference on Design automation, pp. 987-990, 2006.

[10] N. Satyanarayana, M. Mutyam, A. V. Babu, Exploiting on-chip data behavior for delay minimization, Proc. of international workshop on System level interconnect prediction, pp. 103-110, 2007.