Interframe Bus Encoding Technique and Architecture for MPEG-4 AVC/H.264 Video Compression

Asral Bahari, Tughrul Arslan and Ahmet T. Erdogan

***Abstract*— In this paper, we propose an implementation of a data encoder to reduce the switched capacitance on a system bus. Our technique focuses on transferring raw video data for multiple reference frames between off-chip and on-chip memories in an MPEG-4 AVC/H.264 encoder. This technique is based on entropy coding to minimize bus transition. Existing techniques exploit the correlation between neighboring pixels. In our proposed technique, we exploit pixel correlation between two consecutive frames. Our method achieves a 58% power saving compared to an unencoded bus when transferring pixels on a 32-bit off-chip bus with 15pF capacitance per wire.**

I. Introduction

Video application has become increasingly popular in today's wireless communications. However, video processing is computing intensive and dissipates a significant amount of power. For MPEG-4 video compression, raw video data (pixel) dominates data transfer [1]. During the compression of five minutes of video (CIF@15 fps), at least 4500 frames are transferred from the memory to the video compressor. These values increase for higher frame rates and frame resolutions.

This high data transfer translates into high power dissipation on the memory-processor busses. This is severe for systems with off-chip memory where the bus load is several orders of magnitude higher than the on-chip bus with a typical value of around 15pF on each bus wire [2]. It has been reported that the off-chip bus consumes 10%-80% of overall power [3].

In this paper, we present a data encoding technique to minimize power dissipation during multiple-frame transfer between the MPEG-4 AVC/H.264 video compressor and the external reference frame memory using an off-chip system bus as shown in Fig. 1. Power reduction is achieved by utilizing bus encoding to reduce switching activity on the bus. Bus encoding transforms the original data such that two consecutive encoded data has lower switching activity than the unencoded one.

[4], [5] and [6] address implemention of the bus encoding on address busses. The proposed methods exploit highly correlated bus addresses to reduce switching activity. Compared to address busses, data busses show more random characteristics. Bus invert [7], codebook [8] and exact algorithm [9] were proposed for this type of data.

Manuscript received June 24, 2008; revised November 20, 2008.

Asral Bahari (email: asral@unimap.edu.my) is with School of Microelectronic Engineering, University Malaysia Perlis, Malaysia.

Tughrul Arslan (tughrul.arslan@ed.ac.uk) and Ahmet T. Erdogan (ahmet.erdogan@ed.ac.uk) are with the School of Engineering and Electronics, University of Edinburgh, Edinburgh EH9 3JL U.K.

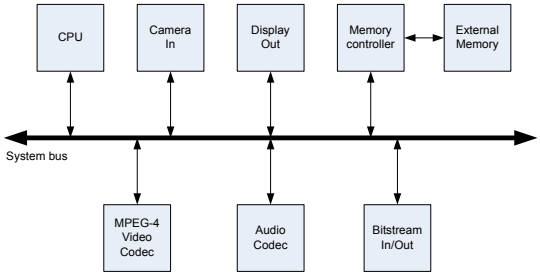

Fig. 1. Typical video communication system.

Existing techniques exploit the correlation between neighbouring pixels for video data. However, pixel correlation between frames has not been fully exploited to reduce bus transition in the literatures. In [10], we have proposed an interframe bus encoding technique where we utilised the pixel correlation between two consecutive frames without full system consideration. In this paper, we extend the technique to a complete H.264 system.

The rest of the paper is organised as follows. Section II reviews the existing intraframe techniques for bus encoding. Section III discusses our approach to reducing the transition activity during memory data transfer. Section IV discusses the proposed implementation of the interframe bus encoding technique for the H.264 system. This is followed by the results and performance benchmarking of our method in Section V. Finally, Section VI concludes the paper.

II. Intraframe Decorrelation

The technique discussed in this paper is based on the combination of difference-base-mapped and value-base-mapped (dbm-vbm), as discussed in [11]. We adopt this method because it allows us to exploit the pixel correlations widely available in video data.

Fig. 2(a) shows the distribution of two adjacent pixels' difference. Four different QCIF video sequences (Akiyo, Foreman, Table Tennis and Grasses), which represent various motion types from low to high, are evaluated. Five frames from each sequence are evaluated, which consists of 190080 pixels. The graph shows that, for highly correlated data, the difference between two consecutive pixels with smaller magnitude has higher probability than the larger magnitude. Dbm-vbm utilises this characteristic to minimize the bus transition.

Fig. 3 shows the block diagram describing the dbm-vbm operation. It consists of decorrelator (dbm) and entropy coder

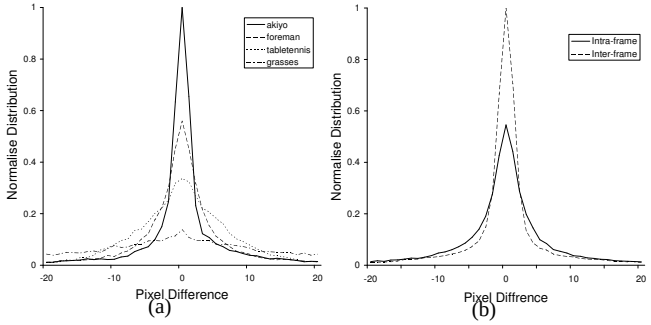

Fig. 2. Pixel decorrelation using (a) adjacent pixel (b) interframe vs. intraframe decorrelation.

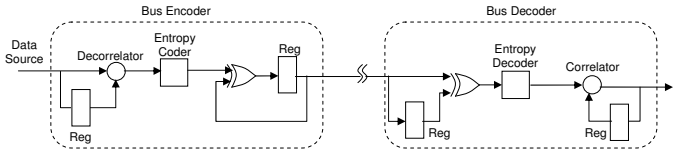

Fig. 3. Dbm-vbm bus encoder and decoder.

(vbm). The dbm-vbm technique is summarized as follows. First, two adjacent pixels (intraframe) are decorrelated using dbm. Dbm calculates the relative difference between the two pixels. Vbm maps the values to patterns that have different weights (i.e., total number of 1s). To reduce the overall transition, it maps the low magnitude value to a pattern that has the fewest 1s, whereas higher magnitude values are mapped to patterns that have more 1s. At the output, the XOR translates 1s as transition and 0s as transition-less.

The average number of transitions for the dbm-vbm method depends on its source-word, i.e., the decorrelator output. The more the graph is skewed toward zero, the more patterns are assigned with less 1s. Thus, one way to improve the transition reduction is by improving the decorrelator.

III. Interframe Decorrelation

Video sequences consist of both spatial and temporal redundancy. The existing bus encoding techniques utilize spatial redundancy within frames. However, the temporal redundancy is not fully exploited to reduce bus transition.

In [10], we have proposed decorrelating the pixels using two consecutive frames (interframe). This method is based on the observation that two consecutive frames are highly correlated. Often, the background of a scene is stationary. Furthermore, for a moving object, the differences between successive frames are very small. Fig. 2(b) compares the pixel decorrelation using the intraframe and interframe methods for five foreman sequences. The figure shows that decorrelating the pixels using interframe improves the graph skew-ness towards zero. This will translate to higher transition saving since more patterns will be assigned with less 1s.

The results in [10] show that the interframe method provides higher transition reduction compared to both bus invert and intraframe implementations. On average, our method reduces up to 65% of the transition over an unencoded bus. This is equivalent to a 1.5 and 2.6 times more transition saving over intraframe and clustered bus invert, respectively.

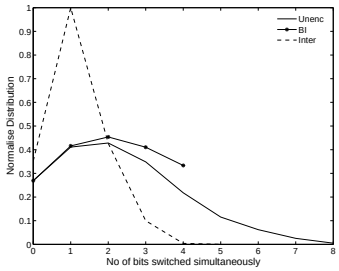

Fig. 4. Simultaneous bit switching comparison between different bus encoding methods: unencoded bus, bus invert and the proposed interframe method.

Fig. 4 shows the normalised distribution for the total number of bits that are switched simultaneously when transferring the pixels. The higher the number of bits that are switched simultaneously, the higher the peak power of the bus. As shown in the figure, interframe bus encoding results in a much lower number of bits switching simultaneously compared to the bus invert method. This shows that not only does it reduce the off-chip average power, but the interframe method also reduces the peak power of the bus.

IV. Interframe Bus Encoding Implementation into H.264 System

As shown in Fig 1, the external memory is connected with the on chip video compressor through an off-chip bus. In typical implementation, the same bus is used to send or receive the data from the external memory. In [10], we assumed that the pixel from the two frames is transmitted in two different busses. In order to realise the interframe bus encoding into hardware implementation, modification has to be made to this setup. This is to take into account the limited availability of the off-chip bus used in the actual system.

In this section, the operation of the H.264 video compression system is first described to illustrate the interaction between the video compressor and the off-chip memory. Then, the proposed architecture for implementing the interframe bus encoding for the H.264 system is discussed in depth to minimise the off-chip bus power.

A. H.264 system

Fig. 5 (a) illustrates the main functional block of the H.264 encoder and the interaction among its main modules. The encoder consists of motion estimation (ME), motion compensation (MC), integer transform (IT), quantizer (Q), variable length coder (VLC), and deblocking filter (DFIR). In addition, the system requires search area buffers (SA) to store search area pixels temporarily from the reference frames memory, and filtered macroblock buffers to keep the intermediate data during the encoding process. The main function of the encoder is to encode the current macroblock pixels into a compact bitstream so that it can be decoded by the decoder.

In H.264, the encoder first loads the search area and the current macroblock (MB) pixels from the external frame memory through the off-chip bus. Then, the current MB is predicted using motion estimation. This process is repeated

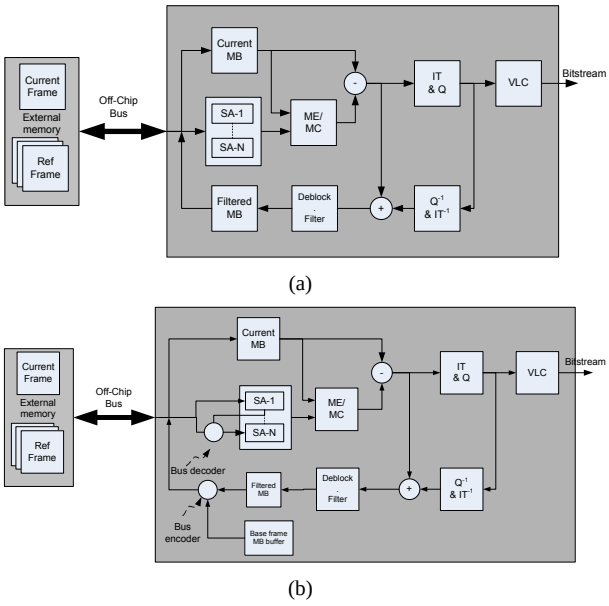

Fig. 5. (a) Interaction between the external reference frame memory and H.264 modules. (b) Modified H.264 system that includes an interframe bus coder.

until all search areas from multiple reference frames are evaluated. The predicted MB that results in the lowest cost is selected. The residue of the predicted MB is then calculated by subtracting the current MB and the predicted MB, before it is transformed by the integer transform. The transform coefficient and motion vector is coded using the variable length coder to generate a compact bitstream. The encoder also reconstructs the current MB using information from transform coefficients. The reconstructed MB is filtered by the deblockng filter before storing the MB in the external reference frame memory for future frame prediction.

The system requires 2000 clock cycles to process one MB by dividing the encoder operation into three pipeline stages: ME/MC, IT/Q/IQ/IT, and DFIR/VLC. For QCIF frame size at 30 frames per second, the system is set to 6 MHz to achieve real-time operation. Based on the UMC $0.13um$ CMOS technology, the system requires a total core area of $4.8mm^2$.

B. Integration of Interframe Bus Encoding Into H.264

As shown in Fig. 5 (a), an interaction between the H.264 modules and the external reference frame buffer occurs on two occasions: (a) when the pixel is stored into the external reference frame buffer after filtering the MB through the deblocking filter; (b) during loading of the search area pixels from the external reference frame buffer into the search area RAM. Thus, the bus encoder and decoder should be implemented in (a) and (b), respectively.

Fig. 5 (b) shows the modified H.264 system to include the proposed interframe bus encoding method. Since reference frames are accessed in serial between video processor and the external frame memory, some modification is made on the interframe bus encoding circuit proposed in [10]. To allow interframe decorrelation during bus encoding, reference frames

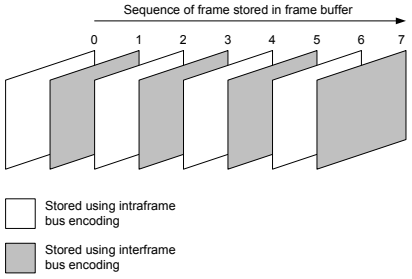

Fig. 6. Reference frame arrangement of external buffer and the type of bus encoding applied to it.

stored in the external memory are arranged in a sequence as shown in Fig. 6. Since the interframe technique requires other frames, the bus coding is alternated between intraframe and interframe.

Base frame (BF) refers to the frame that is stored as intraframe by the bus encoder, since it does not require any other frames. Dependence frame (DF) is a frame that is stored as interframe by the bus encoder, since it requires other frames to decode the pixel values. DF is always stored after its BF for ease of decoding later on.

In order to allow interframe decorrelation, pixels from the base frame have to exist for the bus encoder. The bus decoder has a similar requirement. For the bus decoder, since the search area is accessed from a multiple number of frames, the proposed solution is to load the base frame first, followed by the dependence frame. When the search area from the dependence frame is loaded, the search area pixel from the base frame is accessed as well to perform the interframe decorrelation.

In contrast to the bus decoder, since the fully filtered MB is only written into the external frame buffer, no base frame is available at the bus encoder. The solution proposed for this is to store the required base frame MB in a buffer. The base frame MB can be stored when loading the base frame search area into the search area memory. Since the deblocking filter operation is located in the third stage of the pipeline buffer, an additional MB buffer is required to store the last three MBs of the base frame. This buffer will be used during interframe decorrelation when writing the pixels into the reference frame buffers. Using this approach, it is possible to perform interframe decorrelation at the bus encoder.

The detailed hardware implementations for the bus encoder and decoder are shown in Fig. 7 and 8, respectively. For this implementation, four pixels at a time are transferred to/from the external memory on a 32 bits bus (8 bits per pixel). Thus, four encoders and decoders are used to encode and decode the bus. Due to space limitation, only one encoder and one decoder are shown in Fig. 7 and 8. The architectures are able to perform either interframe or intraframe bus encoding. When intraframe is selected, the bus encoder calculates the dbm decorrelation using the filtered pixel and the previous pixel stored in the register (Reg). Similar input is used to calculate the dbm^{-1} at the bus decoder.

During the interframe bus encoding, the encoder calculates the dbm using filtered pixels and pixels stored in the buffer MB as shown in Fig. 7. Since the buffer MB stores the cor-

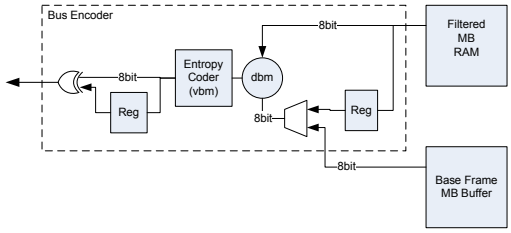

Fig. 7. Interframe-intraframe encoder for H.264 hardware

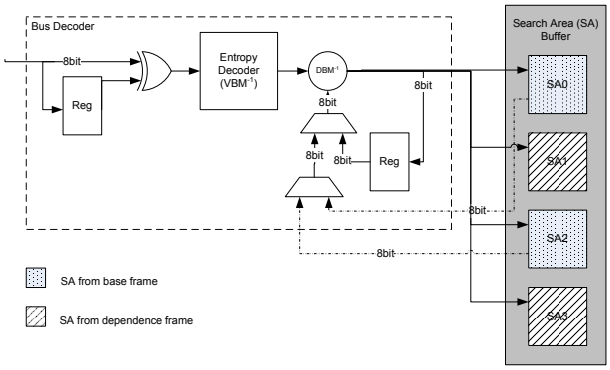

Fig. 8. Interframe-intraframe decoder for H.264hardware

responding base frame MB, this is equivalent to decorrelating a pixel from two frames. To decode the pixel, the search area from the base frame is first loaded into the search area buffer. When the search area from the dependence frame is loaded, the loaded search area from the base frame is read in parallel to correlate the search area pixel before storing it in the search area RAM. This is shown by the dotted line with an arrow in Fig. 8.

V. Results and Discussion

The design was synthesised using the UMC $0.13um$ CMOS library. Verilog-XL and Power Compiler were used to perform functional simulation and power analysis using the extracted layout data, respectively. Actual video data was used to verify the hardware and to obtain the estimated power consumption.

Tables I and II illustrate the resources required to implement the proposed interframe bus encoder and decoder in the H.264 system. For this implementation, four pixels at a time are transferred to/from the external memory on a 32 bit bus. Thus, four encoders and decoders are used to encode and decode the bus. The tables show that the maximum delay and total area overhead required to implement the hardware is 4.6 ns and $0.16mm^2$, respectively. This is equivalent to 3% of the total area in the conventional H.264 as discussed in Section IV-A.

Tables I and II also show the power evaluation result of the proposed architectures. From the table, it can be seen that the proposed encoder and decoder circuits consume 0.568 and 0.470 mW of power during interframe bus coding mode, respectively, with memory dominating the circuit power overhead. Since the intraframe decorrelation does not require any memory access, the total circuit power during intraframe bus encoding and decoding mode is much lower at 0.176 and 0.060 mW, respectively.

TABLE I

The bus encoding area and power overhead required to implement the interframe-intraframe bus coding on a 32 bit bus at 6MHz

Encoder Modules	Area (mm^2)	Power (mW)		Delay (ns)
		Interframe mode	Intraframe mode	
Logic	0.03	0.178	0.176	4.4
Memory	0.10	0.39	-	
Total	0.13	0.568	0.176	

TABLE II

The bus decoding area and power overhead required to implement the interframe-intraframe bus coding at 6MHz

Decoder Modules	Area (mm^2)	Power (mW)		Delay (ns)
		Interframe mode	Intraframe mode	
Logic	0.03	0.119	0.060	4.6
Memory	-	0.352	-	
Total	0.03	0.470	0.060	

Fig. 9 and 10 show the total power consumption during interframe bus encoding and decoding, respectively, as compared to unencoded bus, bus coded using cluster bus invert and intraframe. The total power consumption, P_T, is calculated as $P_T = P_{Circuit} + P_{CL}$, where $P_{Circuit}$ represents the power consumption due to bus encoder or decoder circuits. P_{CL} is the total bus power consumption estimated by $P_{CL} = \frac{1}{2}C_L V_{DD}^2 f\alpha$, where C_L is capacitance load, V_{DD} is operating voltage, f is operating frequency and α is switching activity. The slope of the graphs in Fig. 9 and 10 is proportional to α, which is dependent on the type of bus encoding used.

For a typical off-chip wire capacitance of 15pF, the total bus encoder power consumption during interframe mode is 3.39mW, while it is 4.8mW during intraframe mode. These are equivalent to 58% and 40% power savings compared to unencoded bus, respectively. The greater power saving achieved by interframe is due to much lower transitions occurring on the busses. A smaller slope, as shown in the graphs in Fig. 9 and 10 reflects this.

Tables III and IV show the total energy consumed by the

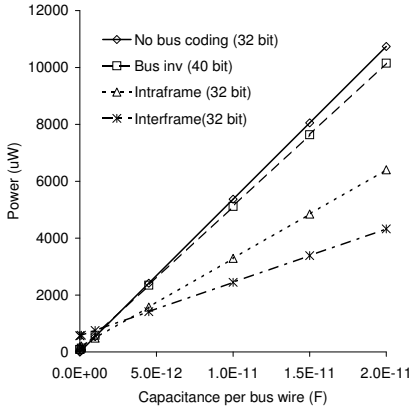

Fig. 9. Power consumption of 32 bit bus at 6MHz when transferring pixels into the external reference frame memory

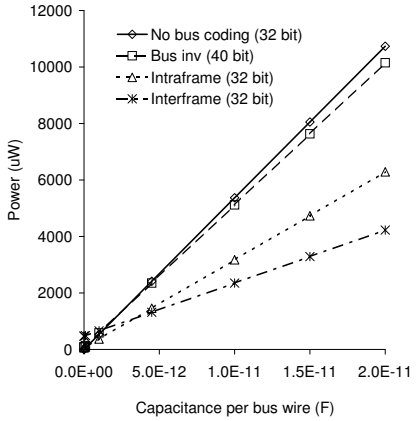

Fig. 10. Power consumption of 32 bit bus at 6MHz when loading pixels from the external reference frame memory

TABLE III

TOTAL ENERGY CONSUMPTION ON 32BIT BUS AT 6MHZ WITH 15PF PER BUS WIRE WHEN SENDING ONE FILTERED MB TO EXTERNAL FRAME BUFFER

Bus encoder	Power (μW)	Clock	Energy (nJ)	Saving (%)
Unencoded	8053.16	64	85.90	-
Bus Inv	7633.83	64	81.43	5
Intraframe	4846.83	64	51.70	40
Interframe	3386.60	64	36.13	58

32bit bus when accessing the external reference frame buffer. Since loading of one search area transfers more data (512 pixels) than storing one MB data into the external reference frame buffer (256 pixels), more energy is consumed during loading of the search area pixel. In addition, the data loaded from the external reference frame buffer is propotional to the number of reference frames used during motion estimation. From the tables, for both cases the interframe bus encoding saves 58% of total energy as compared to an unencoded bus.

VI. SUMMARY

We have presented an interframe bus encoding technique for applications where multiple frames are transferred between off-chip memories, such as in MPEG-4 AVC/H.264 applications. The proposed interframe bus encoding technique results in a 65% transition reduction over the unencoded bus. This is equivalent to a 58% power saving compared to an unencoded bus when transmitting four pixels at a time over a 32-bit bus with 15pF capacitance per wire.

TABLE IV

TOTAL ENERGY CONSUMPTION ON 32-BIT BUS AT 6MHZ WITH 15PF PER BUS WIRE WHEN RECEIVING ONE SEARCH AREA FROM EXTERNAL FRAME BUFFER FOR ONE REFERENCE FRAME.

Bus decoder	Power (μW)	Clock	Energy (nJ)	Saving (%)
Unencoded	8053.16	128	171.80	0
Bus Inv	7633.83	128	162.85	5
Intraframe	4730.83	128	100.92	40
Interframe	3288.60	128	70.16	58

REFERENCES

[1] C.-H. Lin, C.-M. Chen, and C.-W. Jen, "Low power design for MPEG-2 video decoder," *IEEE Transactions on Consumer Electronics*, vol. 42, no. 3, pp. 513 – 521, 1996.

[2] W.-C. Cheng and M. Pedram, "Chromatic encoding: a low power encoding technique for digital visual interface," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 320 – 8, Feb. 2004.

[3] T. Givargis and F. Vahid, "Interface exploration for reduced power in core-based systems," *Proceedings. 11th International Symposium on System Synthesis*, pp. 117 – 22, 1998.

[4] H. Mehta, R. Owens, and M. Irwin, "Some issues in gray code addressing," *Proceedings. The Sixth Great Lakes Symposium on VLSI*, pp. 178 – 81, 1996.

[5] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic zero-transition activity encoding for address busses in low-powermicroprocessor-based systems," *Proceedings Great Lakes Symposium on VLSI*, pp. 77 – 82, 1997.

[6] W. Fornaciari, M. Polentarutti, D. Sciuto, and C. Silvano, "Power optimization of system-level address buses based on software profiling," *Proceedings of the Eighth International Workshop on Hardware/Software Codesign.CODES 2000*, pp. 29 – 33, 2000.

[7] M. Stan and W. Burleson, "Bus-invert coding for low-power i/o," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 3, no. 1, pp. 49 – 58, Mar. 1995.

[8] S. Komatsu, M. Ikeda, and K. Asada, "Low power chip interface based on bus data encoding with adaptive code-bookmethod," *Proceedings Ninth Great Lakes Symposium on VLSI*, pp. 368 – 71, 1999.

[9] L. Benini, A. Macii, M. Poncino, and R. Scarsi, "Architectures and synthesis algorithms for power-efficient bus interfaces," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 9, pp. 969 – 80, Sept. 2000.

[10] A. Bahari, T. Arslan, and A. Erdogan, "Interframe bus encoding technique for low power video compression," *20th International Conference on VLSI Design*, pp. 691 – 698, 2007.

[11] S. Ramprasad, N. Shanbhag, and I. Hajj, "A coding framework for low-power address and data busses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 2, pp. 212 – 21, June 1999.