ABSTRACT

The I/O power consumption in MPEG-2 decoder is significant because of the wide connection with large capacitances to the frame buffer. To reduce the power dissipation on the Memory Bus, the Gray code encoding scheme is proposed to increase the correlation of the image data transferred on the bus. The bit switching probability in the re-coded data will then decrease and in turn the bus power consumption will be reduced. Combined with the proposed bus arbitration and scheduling scheme proposed in this paper, 22% reduction of power dissipation may be achieved.

I. INTRODUCTION

ISO standard 13818-2[1] known as MPEG-2 (Moving Pictures Expert Group) has been adopted in many applications like DVD Player, set-top boxes and entertainment machines. Because the required computing power for this algorithm is huge, a high performance VLSI solution to the MPEG-2 video decompression is necessary. To promote the success of this motion picture standard, the cost in the video decoding system should also keep low. Usually, the key components in a low cost MPEG-2 decoding system includes a high performance single chip MPEG-2 decoder VLSI and the associated frame buffer DRAM.

Although most MPEG-2 applications belong to the non-portable ones, the cost of providing power and associated cooling indicates that the power reduction is still necessary for non-portable applications[2]. Several approaches to reduce the power consumption for the video decoding system has been proposed[3][4]. In [3] the circuit approach using low-power ASIC RAMs as the on-chip I/O buffers with Selective Bit Line Precharge (SBLP) scheme is adopted to reduce the bit line current. In [4] a low-power approach at architectural level is proposed to reduce power consumption for MPEG functional unit like DCT. The power reduction of on-chip buffers and functional units in MPEG-2 decoder can be achieved using these approaches. However, because of the nature of inter-frame coding in the MPEG-2 video decompressing algorithm, there must offer enough memory bandwidth for single-chip MPEG-2 decoder to perform all the frame buffer access. The fast and wide memory bus between decoder chip and frame buffer DRAM will cause a significant power consumption because of large capacitance involved at the I/O. Thus, it is also necessary to explore approaches to decrease the I/O power dissipation.

In this paper, we propose some architectural approaches to reduce power consumption for the I/O bus. This strategy has been adopted in the MPEG-2 Video Decoder Chip we designed[5]. In section II we first give an overview to our MPEG-2 video decoder VLSI. According to the static characteristics of data on the Memory Bus, the Gray code encoding scheme to reduce the circuit switching activity is described in section III. In Section IV we describe the bus arbitration and scheduling strategy to preserve the static characteristics of data while transferring them dynamically on the Memory Bus. Section V concludes this paper.

II. THE MPEG-2 VIDEO DECODER

A. System Architecture

Fig. 1 shows the block diagram of our MPEG-2 video decoder VLSI. Together with four 4M-bit DRAM and some video post-processing components, a complete video decoder system for MPEG-2 video at main profile and main level (MP@ML) can be constructed.

The Decoding Pipeline consists of three functional blocks: variable-length/run-length decoder (VLD-RLD), inverse quantization/inverse cosine transform unit (IQ-IDCT), and motion compensation/interpolation unit (MC-Interpolator). These functional units can perform the main MPEG-2 decompression algorithm. The RISC-based System Controller can control all other associated functional blocks according to the system parameters issued by host system and the programs stored in instruction RAM. Also, it parses the header information of MPEG-2 bitstream and performs the error concealment routines while error bitstream is encountered. Under the control of System Controller, the decoding pipeline can decode the variable-length-coded information in the MPEG-2 bitstream.

According to the parameters from MPEG-2 bitstream header, the Memory Controller can set up the suitable addresses to load reference pictures. Also, it stores reconstructed picture and loads display picture. It also manages a circular buffer located in external DRAM as the video rate buffer (VBV buffer). The external DRAM, which
is connected to the Memory Controller via the DRAM interface, is used to hold the VBV buffer and store decoded frames.

This chip has two internal buses. The 64-bit Memory Bus is used to deliver the incoming MPEG-2 bitstream and decoded pictures. All data transfers to and from the external DRAM pass through this data bus. The control commands between the System Controller and other functional blocks are issued via the 16-bit RISC Data Bus.

B. MPEG-2 Decoding Process

The steps to decode the MPEG-2 bitstream are listed as follows:

1) Coded bitstream supplied to the Host Interface is written to the VBV buffer on external DRAM through the bitstream FIFO and Memory Bus. The transfer is initialized by host system.

2) Coded bitstream is then read out from the VBV buffer, via VLD FIFO, to the VLD-RLD. If the incoming bits belong to the high layer fixed-length coded data (e.g., sequence header, GOP header, picture header and slice header as defined in MPEG-2), they will be then forwarded to System Controller to extract the decoding parameters. On the other hand, if variable-length codes (macroblock header or quantized dct coefficients) are encountered, they will be decoded in the VLD-RLD under the control of System Controller. The decoded DCT coefficients will then be transferred to IQ-IDCT.

3) System Controller first sets up the address of current macroblock for the Memory Controller and checks the type of this macroblock. If the current macroblock is non-intra coded, System Controller also has to calculate the actual motion vectors and set up the addresses of reference macroblocks for the Memory Controller for this non-intra macroblock.

4) The quantized DCT coefficients are first de-zigzagged and inverse-quantized in IQ unit. Then those values are passed to IDCT unit to recover to the original pixel values or DFD. Finally, the output will be fed to MC-Interpolator.

5) For non-intra macroblock, MC-Interpolator initializes a memory transfer through Memory Interface to load the reference macroblocks from reference Picture Buffer on external DRAM to the Reference MB buffer. The addresses of these macroblocks are derived by the motion vectors. Interpolation will be needed if motion vectors are given in the half-pel boundary. After MC-Interpolator adds the data of IDCT results and reference macroblock, it will then write the reconstructed blocks to the Picture Buffer on external DRAM. On the other hand, MC-Interpolator will forward the IDCT results to the Picture Buffer if intra-coded macroblock is encountered.

6) After decoding one frame, the reconstructed image is re-read from the Picture Buffer according to a specified format and output to the video subsystem outside the decoder chip through Video Interface.

III. BUS POWER REDUCTION

A. Bus Power Consumption

In order to reduce the number of DRAMS and the number of I/O pins, the VBV Buffer and Picture Buffer share the same memory port. Wide memory bus organization is often adopted to support sufficient bandwidth because of the low speed nature of DRAM. In our MPEG-2 decoder system that supports MPEG-2 specification at main profile and main level, a 64-bit Memory Bus is used as the I/O channel between decoder chip and frame buffer memory. Not only the larger number of switching signals in the Memory Bus will contribute a significant percentage of power consumption for MPEG-2 decoder, but they also do for the DRAM. Therefore, it is necessary to explore the strategy to reduce the power dissipation associated with the I/O bus.

The bus power consumption is dominated by the switching power consumption[2]:

\[ P_{\text{switching}} = \alpha_{0\to1} \cdot C_L \cdot V_{dd}^2 \cdot F_{CLK} \]  

(1)

where \( F_{CLK} \) denotes the clock rate, \( V_{dd} \) is the supply voltage, \( C_L \) denotes the node capacitance, and \( \alpha_{0\to1} \) is defined as the average number of times in each clock cycle that a node will make a power consumption transition (0 to 1).

As the technology is progressed to the sub-micron technology, the bus power problem is even more severe. The capacitance (\( C_L \)) percentage in bus routing with respect to the whole chip is becoming larger and larger. Under the consideration that the device process or circuit performance will not be influenced by the change of the voltage or frequency, we try to find strategies to reduce the switching probability \( \alpha_{0\to1} \) in order to decrease the bus power dissipation.

B. Static Analysis for Data Sequence of Values

Using coding as a method to reduce the number of bit switches on the data bus has been proposed[7]. Although the "Bus-Invert Coding" in [7] for data bus is optimal for the random-distributed sequence of data, the required extra buses and "majority voter" circuits contribute a significant percentage to the system cost. Furthermore, the data sequence on the Memory Bus of our MPEG-2 decoder is not random-distributed. Therefore, we have to explore another method to reduce bus switching activity and power dissipation.

Our goal is to find an architectural approach to decrease
the bus switching activity and then reduce the power consumption on the Memory Bus of our MPEG-2 video decoding system. We first analyze the static data transfer patterns for the different traffic sources on the bus. The I/O processes occurred on the Memory Bus include:

(a) Bitstream Write: Storing compressed bitstream from Bitstream FIFO to the VBV Buffer on external DRAM.
(b) Bitstream Read: Loading compressed bitstream from VBV Buffer to VLD FIFO.
(c) MC Read: Loading reference macroblocks from the Picture Buffer on external DRAM to Motion Compensation unit.
(d) MC Write: Storing reconstructed macroblock from Motion Compensation unit to Picture Buffer.
(e) Video Read: Loading image pixels from Picture Buffer to Video FIFO for displaying.

The data of I/O processes (a) and (b) belong to the compressed MPEG bitstreams. Although this kind of data sequence does not show much temporal correlation, their occurrence on the Memory Bus is relatively rare and will not contribute much to the bus power consumption. On the other hand, the data of I/O processes (c), (d) and (e), which dominates the Memory Bus I/O, belong to the reconstructed image pixels. Fig. 2 shows the data sequence patterns for these kinds of transactions. The data sequence patterns for the MC Read/Write processes are block-wise as shown in Fig. 2(a), where the 64-bit data words (eight pixels per word) in the same macroblock column (8x16 pixel samples) are read/written column by column. For the Video Read process, the image data are read in raster-scan as shown in Fig. 2(b), where the 64-bit data words are read scanline by scanline.

For these kinds of data transfers, each bit slice [N+8:N] (where \(N = 0, 8, 16, \ldots, 56\)) on the Memory Bus contains one pixel sample. The highly spatial correlation in the image frame indicates that the temporal data sequence may be also highly correlated while transferring them on the Memory Bus. Fig. 3 illustrates the distribution of bus value difference between the value of the bus bit slice [N+8:N] on the present transfer cycle and that on the next cycle. The listed I/O processes are "MC Read" and "Video Read" and the associated MPEG-2 bitstreams include "Table Tennis", "Football" and "Flower garden". For these I/O processes, Fig. 3 shows that two numbers with small difference between them are more likely to happen successively on Memory Bus than those with large difference. Fig. 4 shows that there are also similar distributions on the bus value difference for bit slice [N+8:N+8-m] (m=4–8). The above static analysis indicates that if we re-code the bus data to the one with the smaller Hamming distance (the number of bits in which they differ) between neighbor data values, the total bit switches on the Memory Bus may be lower.

### C. Gray Code Representation

One candidate code is "Gray code" representation, which changes by only one bit as it sequences from one number to the next [8]. The conversions between (n+1)-bit binary \((b_n, b_{n-1}, \ldots, b_1, b_0)\) and Gray code \((g_n, g_{n-1}, \ldots, g_1, g_0)\) are:

**Binary to Gray Code:**

\[
\begin{align*}
g_n &= b_n \\
g_i &= b_{i+1} \oplus b_i \quad (i = n-1, 0)
\end{align*}
\]

**Binary to Gray Code:**

\[
\begin{align*}
b_n &= g_n \\
g_i &= b_{i+1} \oplus g_i \quad (i = n-1, 0)
\end{align*}
\]

The circuit overhead for converting (n+1)-bit binary code to Gray code is n two-input exclusive-OR gates with critical path of one gate delay. However, while there need still only n two-exclusive-OR gates to convert (n+1)-bit Gray code to binary code, the critical path will be n gate delays. The maximum bit width using the Gary code encoding scheme may be limited by the long critical path of this inverse conversion.

Fig. (5) shows the distribution of Hamming distance for 4-bit and 8-bit binary code and Gray code representation. Obviously, while two numbers are encoded with Gray code representation, the average Hamming distance between them will be smaller than the distance using binary code if the value difference between these two numbers is odd. For an even value difference, although the average Hamming distance will be larger while using Gray code representation, the difference of these two average Hamming distances is not large for small value differences. In our MPEG-2 decoder, most data on Memory Bus are temporally correlated. As Fig. (3) and (4) are shown, the difference between the present bus value and the next one is usually small. Therefore, using Gray code encoding scheme to re-code the bus data for I/O processes (c)-(e) will cause the reduction in the total number of bit switches on Memory Bus.

Table 1 illustrates the number of bit switches using Gray code encoding scheme and "Bus-Invert Coding" for I/O processes "MC Read" and "Video Read". The m-bit Gray code means that we convert the bit slice \([N+8:N+8-m]\) of the data (where \(N=0, 8, \ldots, 56\) and \(m=4\) and \(8\)) into m-bit Gray code and leave the bit slice \([N+7-m:N]\) unchanged. As the table is shown, the number of bit switches using Gray code encoding scheme decreases as \(m\) increases. It is reasonable because the distribution of bus value difference is non-increasing and there is only one-bit difference between m-bit and (m-1)-bit Gray code. Also, although Gray code representation reduces the number of bit switches for data for all video sequences, the reduction degrees are different for different sequences. For sequence with larger near-still background like "Table Tennis", the reduction can be high as 22%. However, the reduction may be only 10% for sequence "Flower garden", for the spatial...
correlation is relatively lower between the adjacent pixels in
the frame. The resulting reduction of bus switches is near to
the results using "Bus-Invert Coding" scheme proposed in
[8], while the cost to implement the Gray code encoding is
far less than that to implement "Bus-Invert Coding" method.

IV. POWER CONSUMPTION AND BUS
SCHEDULING

The static pattern analysis for different I/O processes
occurred on the Memory Bus of MPEG-2 decoder shows
that the number of bit switches will be effectively reduced
by the Gray code encoding scheme. However, there is no
dedicated I/O channel for each I/O process and a single
Memory Bus is time-shared by all the traffic sources in
order to reduce the complexity of the DRAM Interface
circuitry. Instead of being transferred continuously, the data
for each I/O process will be organized in small burst to be
transferred on the Memory Bus. Because bus arbitration
/scheduling strategy determines the intermixed patterns of
data from different traffic sources, it is necessary to explore
the influence of bus scheduling on bus switching activity. In
section 4-A we first shows a bus arbitration/scheduling
model for the dynamic analysis of data sequence. Based on
this analysis, the impact of bus arbitration/scheduling on the
bus switching activity will be described in section 4-B.

A. Bus Scheduling Model

In order to acquire high throughput, all the processes
that initialize I/O transaction will access DRAM by burst to
take advantage of page mode feature. Assume that the real-
time scheduling model for the I/O processes on the Memory
Bus is a non-preemptive one. For a set of n I/O processes
$\tau_1, \tau_2, ..., \tau_n$, with priority levels $P_1, P_2, ..., P_n
(P_1 > P_2 > ... > P_n)$, the process of $\tau_i$ will not miss its deadline for any
transaction release time under fixed-priority scheduling if
the following holds [9][10]:

$$S_i = \min_{0 \leq j \leq h} \left\lfloor \frac{C_j + \Delta}{t} \right\rfloor \frac{t}{T_{ref}} + \frac{B_j}{t} \leq 1$$

(4)

where

- $T_j$ denotes the minimum cycles between two
transactions of $\tau_i$.
- $C_j$ denotes the cycles required to transfer data of $\tau_j$ at each bus transaction.
- $D_j$ is the deadline of $\tau_j$.
- $B_j$ is the worst case blocking cycles of $\tau_j$.
- $C_{ref}$ is the DRAM refresh cycles.
- $T_{ref}$ is the DRAM refresh period.
- $\Delta$ is the overhead because of bus arbitration and
DRAM page fault.

For a single bus architecture with bus width $W_{bus}$,
suitable I/O buffering for the I/O processes may be needed
in order to prevent the associated functional unit from
starving. Assume that the buffer associated with process $\tau_i$
follows FCFS discipline and has a FIFO length $L_i$ and a
FIFO threshold $TH_i$. In addition to send/receive data to/from the Memory Bus, the FIFO is also filled/emptied by
associated functional unit with rate $R_i$ (Byte/cycle). The data
filling/emptying will change the FIFO data level and a bus
request from FIFO will be generated while the FIFO level
becomes higher/lower than the FIFO threshold.

The deadline $D_i$ is of $\tau_i$ can be defined as the time that
the FIFO become underflow or overflow. Then we have:

$$D_i = \frac{TH_i}{R_i}$$

(5)

The worst case blocking time of $\tau_i$ is simply the
maximum execution time over all lower priority processes:

$$B_i = \max_{j: P_j < P_i} C_j$$

(6)

Let $t = D_j$ and from (4), (5) and (6) we can estimate
FIFO threshold $TH_j$ as:

$$TH_j \geq \frac{\max_{j: P_j < P_i} \left( C_j + \Delta \right)}{\sum_{j=1}^{n} \frac{C_j + \Delta}{T_j} + \frac{C_{ref}}{T_{ref}}} - \frac{R_i}{1}$$

(7)

Note that the above equation is slightly pessimistic
because we ignore the minimum constraints in (4) and
assume $S_i$ to be unity.

The FIFO length $L_i$ can also be calculated as:

$$L_i = C_i(W_{bus} - R_i) + TH_i$$

(8)

providing that:

$$T_i = \frac{C_i(W_{bus} - R_i) - TH_i}{R_i} > 0$$

(9)

The bus and FIFO model for our MPEG-2 decoder is
shown in Fig. 6. The Memory Controller consists of a non-
preemptive fixed-priority scheduler to arbitrate the bus I/O.
If we let $P_{video_fifo} > P_{audit_fifo} > P_{bitstream_fifo} > P_{mem}$, we have the lower bound of the FIFO length associated with Video
FIFO, VLD FIFO and Bitstream FIFO as:

$$L_{video_fifo} \geq C_{video_fifo}(W_{bus} - R_{video_fifo}) + \max\left( C_{mem}, C_{video_fifo}, C_{bitstream_fifo} \right) + \Delta$$

$$1 - \left( \frac{C_{ref}}{T_{ref}} \right)$$

(10)

$$R_{video_fifo}$$
The equations (10), (11) and (12) shows that the FIFO length is determined by burst size of each transaction and the FIFO filling/emptying rate of associated functional unit. The first term in equations (10)-(12) accounts for the FIFO capacity to send/receive a burst of data to/from Memory Bus, while the capacity to accommodate the blocking by other tasks contributes the second one. Obviously, if the MPEG-2 decoder chip adopts fixed-priority scheme to arbitrate bus VO, data transfer may be blocked by another data transfer with the time proportional to the transfer burst size.

**B. Bus Scheduling and Switching Activity**

In order to preserve the data correlation of VO transfers for image-type data, it is necessary to reduce the probability of intermixing the transactions of different I/O processes on Memory Bus. As Fig. 7 is shown, transferring those I/O processes with larger burst size is a good way to reduce the intermixed probability and preserve the data correlation. However, equations (10)-(12) indicate that the internal buffer size should be increased in order to ensure that the FIFO will not be overflow or underflow and to prevent the functional units from starvation. Large internal buffer memories not only increase the chip area, but also consume more power. Therefore, in order to preserve the data correlation, the non-preemptive fixed-priority arbiter in Memory Controller must be modified to accommodate larger burst of memory access without affecting the size of the internal buffer.

Our goal is to construct a arbiter/scheduler to accommodate larger transaction length for different I/O processes without the necessity to increase the total FIFO size. We first analyze the characteristics of memory I/O transactions as listed in table 2. It can be found that the required bandwidth for the processes "Bitstream Read" and "Bitstream Write" are relatively lower. This means that the associated FIFO filling/emptying rates and are also low and we can estimate the length of both FIFOs as:

\[
L_{\text{vid}.\_\text{fifo}} \geq C_{\text{vid}.\_\text{fifo}}(W_{\text{bus}} - R_{\text{vid}.\_\text{fifo}})
\]

The data in table 2 also indicate that the transactions MC read, MC write and Video Read dominate the memory bus I/O time. However, because these I/O processes are deterministic, it is possible to schedule these tasks in the decoding time domain. According to the off-line schedule analyzed in advance, the Memory Controller then performs arbitration as follows. Normally, the Memory Controller monitors the I/O requests to or from VBV buffer and performs the compressed bitstream transfer for VLD FIFO and bitstream FIFO. While it is time for I/O process like Video Read, MC Read and MC Write, the bus will be allocated to that process until the transaction ends. The Memory Controller will then return to the state to handle the memory access to/from VLD FIFO/bitstream FIFO. Because the processes Video Read, MC Read and MC Write are off-line scheduled, the latter processes will not contribute the blocking time to the former. On the other hand, the blocking time of Video Read contributed by process "Bitstream Read" and "Bitstream Write" will also be less because the scheduler polls the state of those FIFOs and fill/empty them as soon as possible. Therefore, by using the propose bus arbitration/scheduling scheme, we can estimate the length of Video FIFO as:

\[
L_{\text{video}.\_\text{fifo}} = C_{\text{video}.\_\text{fifo}}(W_{\text{bus}} - R_{\text{video}.\_\text{fifo}})
\]
image data is then an effective way to reduce the bit switches on the Memory Bus because this code representation minimizes the Hamming distance for adjacent numbers. On the other hand, using large burst to transfer these image data may preserve more data correlation than using the shorter one. However, the total size of required I/O buffer is also larger while transferring data with large burst. A bus arbitration and scheduling scheme is then proposed to allow larger burst of I/O transfer with less buffer size penalty than the fixed-priority approach. Both the Gray code representation and the proposed bus scheduling scheme are adopted in our MPEG-2 decoder VLSI to reduce the chip and system power dissipation.

VI. ACKNOWLEDGE
This work is supported by the National Science Council under grant NSC 79-0414-E009-008 and United Microelectronics Corporation.

VII. REFERENCES
Fig. 3. The distribution of bus value difference for the bit slice \([N+8:N]\) on the Memory bus, (a) I/O process "MC Read", (b) I/O process "Video Read".

Fig. 4. The distribution of bus value difference for the bit slice \([N+8:N+8-m]\) of the data (where \(N=0, 8, \ldots, 56\) and \(m=4-8\)). The I/O process is "MC Read".

Fig. 5. The distribution of Hamming distance for binary code and Gray code, (a) 4-bit representation, (b) 8-bit representation.

Fig. 6. The bus and FIFO model for the MPEG-2 decoder.
Fig. 7. The relation between transfer bulk size and number of bit switches on the bus.

Fig. 8. Relation between transfer bulk size and the size of internal buffer (Bitstream FIFO + VLD FIFO + Video FIFO) in two scheduling/ arbitration schemes; dot line: fixed-priority scheme, solid line: the proposed scheme.

<table>
<thead>
<tr>
<th>Video sequence</th>
<th>Process</th>
<th>Bit switches reduction (4-bit Gray code)</th>
<th>Bit switches reduction (8-bit Gray code)</th>
<th>Bit switches reduction (Bus-Invert Coding, one invert b for 8 b bus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table tennis</td>
<td>MC Read</td>
<td>19%</td>
<td>22%</td>
<td>24%</td>
</tr>
<tr>
<td></td>
<td>Video Read</td>
<td>17%</td>
<td>19%</td>
<td>23%</td>
</tr>
<tr>
<td>Football</td>
<td>MC Read</td>
<td>7%</td>
<td>14%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Video Write</td>
<td>10%</td>
<td>13%</td>
<td>15%</td>
</tr>
<tr>
<td>Flower garden</td>
<td>MC Read</td>
<td>6%</td>
<td>12%</td>
<td>13%</td>
</tr>
<tr>
<td></td>
<td>Video Write</td>
<td>6%</td>
<td>10%</td>
<td>18%</td>
</tr>
<tr>
<td>Implementation Cost</td>
<td></td>
<td>24 XORs</td>
<td>56 XORs</td>
<td>128 XORs</td>
</tr>
</tbody>
</table>

Table 1. Comparison of switching activity reduction and implementation cost between Gray code encoding and "Bus-Invert Coding" method.

<table>
<thead>
<tr>
<th>Memory I/O Process</th>
<th>Characteristics</th>
<th>Bandwidth (Byte/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitstream Write</td>
<td>Stochastic</td>
<td>600k</td>
</tr>
<tr>
<td>Bitstream Read</td>
<td>Stochastic</td>
<td>4M</td>
</tr>
<tr>
<td>MC read</td>
<td>Deterministic</td>
<td>38.4M</td>
</tr>
<tr>
<td>MC Write</td>
<td>Deterministic</td>
<td>15.2M</td>
</tr>
<tr>
<td>Video Read</td>
<td>Deterministic</td>
<td>15.2M</td>
</tr>
</tbody>
</table>

Table 2. I/O Characteristics on the Memory Bus.

<table>
<thead>
<tr>
<th>Scheduling</th>
<th>4-bit Gray Code</th>
<th>8-bit Gray Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed-priority scheme (C_{\text{video}}=4, C_{\text{soc}}=16)</td>
<td>The proposed Scheme (C_{\text{video}}=6, C_{\text{soc}}=32)</td>
<td>Fixed-priority scheme (C_{\text{video}}=4, C_{\text{soc}}=16)</td>
</tr>
<tr>
<td>Table Tennis</td>
<td>14%</td>
<td>26%</td>
</tr>
<tr>
<td>Football</td>
<td>5.6%</td>
<td>18%</td>
</tr>
<tr>
<td>Flower garden</td>
<td>4.3%</td>
<td>17%</td>
</tr>
</tbody>
</table>

Table 3. Comparison of switching activity reduction by 4-bit and 8-bit Gray code encoding method with different scheduling schemes.
Chia-Hsing Lin was born in Taipei, Taiwan, ROC, in 1969. He received the B.S. degree from National Chiao Tung University in 1991. He is a Ph.D. Candidate in Electronics Engineering at EECS College of National Chiao Tung University, Hsinchu, Taiwan, ROC. His research interests include VLSI architecture for video signal processing, computer graphics and multimedia networking. He is a student member of IEEE and ACM.

Chen-Min Chen was born in Kaohsiung, Taiwan, ROC, in 1972. He received the B.S. and M.S. degrees from National Chiao Tung University, Hsinchu, Taiwan, in 1994 and 1996, respectively. His major research interests include low power analysis in software level of abstraction and embedded processor design. He is a member of Phi Tau Phi.

Chein-Wei Jen was born in Shanghai, China, in 1948. He received the B.S. degree from National Chiao Tung University in 1970, the M.S. degree from Stanford University in 1977, and the Ph.D. from National Chiao Tung University in 1983. He is a professor in the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan. From 1985 to 1986 he was a Visiting Researcher at the University of Southern California, USA. His current research interests include VLSI signal processing, VLSI architecture design, design automation, and fault tolerant computing. He is a member of IEEE and Phi Tau Phi.