VLSI Design
1995, Vol. 3, No. 1, pp. 59–65
Reprints available directly from the publisher
Photocopying permitted by license only

A Comparison of Bit Serial and Bit Parallel DCT Designs

DAVID CROOK and JOHN FULCHER
Department of Computer Science, University of Wollongong, Australia.

(Received December 13, 1992, Revised January 13, 1994)

Bit parallel and bit serial VLSI designs for performing 1-dimensional Discrete Cosine Transforms are compared, in terms of size, complexity and throughput. It is concluded that the bit serial approach is more suited to this application, given the limitations of not only the available VLSI design tools, but also the available silicon real estate allocated for final chip fabrication.

1. INTRODUCTION

In recent times the Discrete Cosine Transform, DCT, has become the de facto standard for compressing video images prior to transmission over limited bandwidth telecommunications channels. Indeed, the DCT has been incorporated into both the Joint Photographic Experts Group, JPEG, standard [1], as well as the Video Codec for Audiovisual Services Recommendation H.261 [2].

The Discrete Cosine Transform—DCT—pair are defined as follows:

Inverse transform:

$$x(k) = \sum_{n=0}^{N-1} e(n)X(n)\cos\left[\frac{(2k+1)n}{2N}\right]$$

$$= \sum_{n=0}^{N-1} e(n)X(n)C_{2N}^{(2k+1)}$$

Forward transform:

$$X(n) = \frac{2e(n)}{N}\sum_{n=0}^{N-1} x(k)\cos\left[\frac{(2k+1)n}{2N}\right]$$

$$= \sum_{n=0}^{N-1} e(n)x(k)C_{2N}^{(2k+1)}$$

where $x(k)$ = time domain series

$$e(n) = 1/\sqrt{2} \quad \text{for } n = 0$$

$$= 1 \quad \text{otherwise}$$

One of the main advantages of the DCT and the resultant, (computationally efficient), Fast Cosine Transform, FCT, is the fewer number of multiplications compared to the Fast Fourier Transform, FFT. For an N-point DCT, a total of $(N/2)^*\log_2 N$ real multiplications are required, compared to the FFT which has $N \log_2 N$ complex multiplications or $4N \log_2 N$ real multiplications.

Lee's algorithm [3, 4] dates from 1984, and this is the algorithm used as the basis for the VLSI chip design in the current study. By decomposing the functions into odd and even indices of $n(k)$, an N-point forward and inverse DCT pair can be transformed into two $(N/2)$-point DCTs and two $(N/2)$-point IDCTs respectively, (assuming N is an integral power of 2). If this procedure is continued until n and k both become zero, then the butterfly module of Figure 1 results (for $N = 8$). For an 8 point DCT there are 13 multiplications and 25 additions (subtractions).

Implementation of the DCT was investigated for both bit parallel and bit serial designs. In each case the arrangement of the algorithm was investigated to determine how it could be implemented to meet the design criteria specified in Section 2.

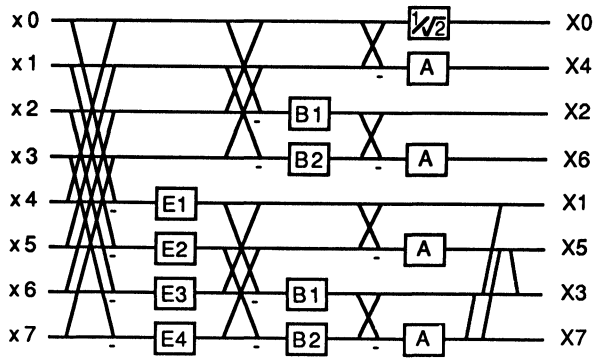


FIGURE 1 8-point Lee DCT Forward Transform

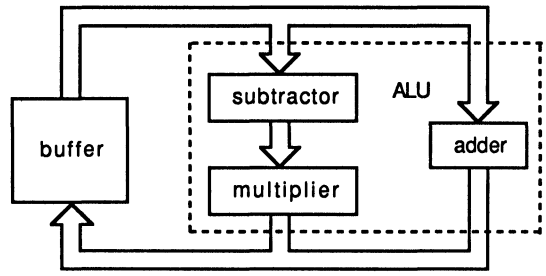


FIGURE 2 Bit Parallel Floor Plan

2. VLSI DESIGN SPECIFICATION

The constraints imposed on the VLSI implementation of the refined Lee DCT algorithm stemmed primarily from the VLSI design tools at our disposal, together with the associated silicon foundry support. Our design was allocated a floorplan of up to 6.8*6.8 mm of a multi-project chip to be fabricated in 2.5μm double metal CMOS using the AWA Microelectronics ASIC facility at Homebush, Sydney, Australia. The design is to operate in real time in the 10 to 15MHz range, and process 288 × 352 pixel images at 30 frames per second. This requires 47520 sets of 8 data values to be processed per second (288*352*30/8*8), which equates to 2104 nsecs for a set of 8 data values.

The essential design criterion was to perform a 1D transform in real time. Additional (desirable) design criteria were:

(i) easy adaptability to 2D transform computations,

(ii) minimise the required silicon real estate, and

(iii) simple timing and control circuitry.

3. BIT PARALLEL DESIGN

The first design considered was a pipelined bit parallel design, based on that of Arnould & Durge [8]. Their design incorporated a single parallel multiplier, together with a pipelined subtractor and parallel adder, but was based on a different DCT algorithm to Lee. The bit parallel design adopted in the present study (Figure 2), uses a parallel multiplier with a pipelined subtractor connected to its front end. This maximizes the data flow through the ALU, since all but one multiplication is preceded by a subtraction (Figure 1).

There is an adder in parallel with the subtractor/multiplier to accommodate the five additions at the end of the butterfly. The results of the adder and multiplier are placed back onto the data bus and stored in the buffers from where the original two data values were read. This requires 13 passes through the bit parallel ALU for multiplications, and a further 5 passes to evaluate the 5 final additions.

The bit parallel multiplier is based on the design of Weste and Eshraghian's [9], and requires 8-bit coefficients. With 12 data bits and 8 bit multiplier coefficients, the settle time of the multiplier is 12 + 8 + 1 = 21 adder settle times. As there is a subtraction or addition preceding the multiplications, these settle times have to be included in the settle time of the ALU. The simplest 12 bit adder or subtractor would consist of 12 serial adders and have a settle time of 12 adders. However, if an efficient 12 bit adder or subtractor is designed as a 12 bit carry select adder then the settle time can be reduced to 6 adder settle times. Thus the total settle time of the ALU is 21 + 6 = 27 adder settle times. SPICE analysis gave a worst case adder settle time of 5 nsec, which amounts to 27*5 = 135 nsecs per ALU pass. Therefore, the total ALU processing time is a minimum of 135 nsecs for multiplications, and only 30 nsecs for additions. The data can then be read on the next clock cycle following these settle times. Performance of a bit parallel design is limited by the settle time of its slowest combinational element which in the present design is the multiplication ALU.

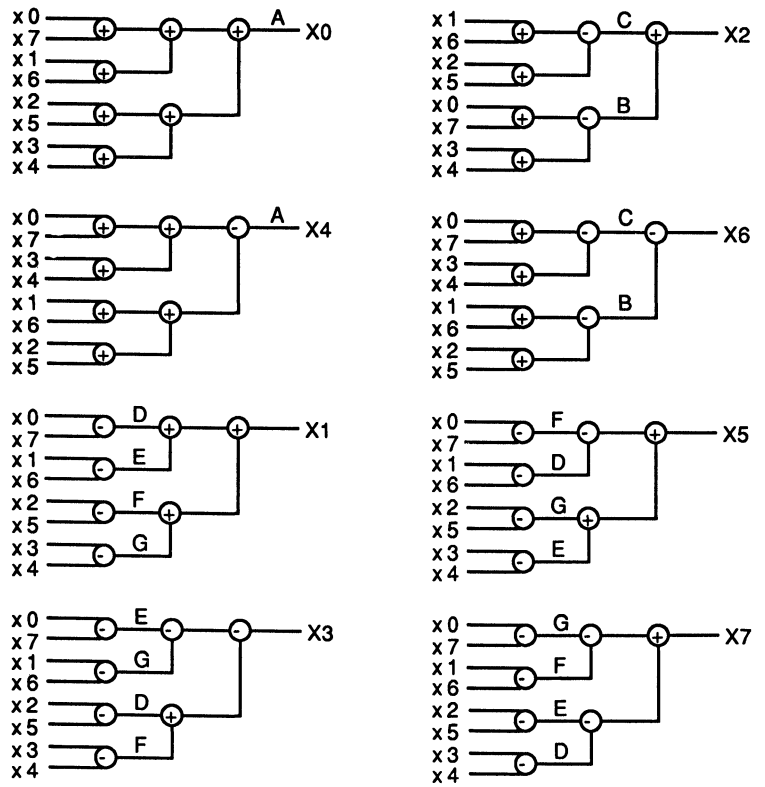
Loading data into the multiplication ALU can be done on the first cycle of the ALU evaluation period and then 135 nsecs are then required to allow the ALU to settle. Reading of the ALU will then be necessary on the next clock cycle. Using a clock rate of 15 MHz a single clock cycle is 66.7 nsecs. There-

fore, 135 nsec settle time in the ALU plus one clock cycle to read the results requires three clock cycles. For the final 5 additions one cycle is needed to load the adder and allow it to settle, then a second cycle is needed to read the results. The total processing of a single transformation is then $(13 \times 3 \times 66.7) + (5 \times 66.7 \times 2) = 3268.3$ nsecs which is outside the allowable 2104 nsecs.

The reader may ask why not divide the ALU into two sections so that on the first cycle data could be written into the first section while concurrently results are read from the second section, then on the second cycle data is fed from the first section into the second section. On the third cycle the results on the second section are read and the first section is loaded again. This would require only $2 \times 66.7 = 133.4$ nsecs to process each pass through the ALU. Considering that there would need to be an initial load cycle for the first data into the ALU, this arrangement would result in a processing time of $66.7 + (13 \times 2 \times 66.7) + (5 \times 2 \times 66.7) = 2468$ nsecs. This design would then require a second set of data buffers so that data could be read and written to the ALU concurrently. By continuing to refine and optimize this design it may be possible to achieve the required processing time of 2104 nsecs. The only way we could imagine achieving this would be to add a *second* ALU. However, this modification would have resulted in the design exceeding the allowable silicon area.

4. BIT SERIAL DCT BUTTERFLY DESIGN

In investigating bit serial designs it was necessary to formulate the algorithm so that the data could be



$A = 0.176776_{10} = 0.00101101_2$
$B = 0.230969_{10} = 0.00111011_2$
$C = 0.095670_{10} = 0.00101101_2$
$D = 0.245196_{10} = 0.00101101_2$

$E = 0.207867_{10} = 0.00110101_2$
$F = 0.138958_{10} = 0.00100100_2$
$G = 0.048772_{10} = 0.00001100_2$

FIGURE 3 Expanded Lee Algorithm

passed through the computational elements in bit serial form. For example, the original Lee algorithm requires that at each point of the butterfly the relevant data must have already been evaluated. Therefore, control over the processing must enable the data to be processed in the correct order. Investigations failed to present a method of implementing the original algorithm that would meet the throughput requirements and so efforts were then directed to refining the original algorithm into a form that would allow the throughput to meet these requirements.

The starting point for the bit serial VLSI design was the INMOS IMS A121 2D DCT Image Processor chip [5], which implements the DCT (IDCT) by making each of the eight transformed values a function of the eight input values:

$$X(i) = f(x(0), x(1), \ldots, x(7)) \qquad i = 0 \ldots 7$$

An 8×8 multiplication matrix is used, in which each row contains the coefficients to produce the values to be summed to produce a single transformed value:

$$X(0) = x(0)m(0,0) + x(1)m(0,1) + \ldots + x(7)m(0,7)$$
$$\vdots = \vdots$$
$$X(7) = x(0)m(7,0) + x(1)m(7,1) + \ldots + x(7)m(7,7)$$

Thus 64 multiplications and 64 additions are required per transform. While computationally expensive, this results in high throughput when implemented in the INMOS pipelined bit serial form.

Closer examination of the INMOS approach reveals that most of the matrix coefficients are *duplicated* in a number of positions within the matrix. By using this approach of expanding the transformation into eight equations and then removing the duplication, the Lee algorithm equations can be expanded to the point where the eight transformed values are a function of the eight input values, as indicated in Figure 3. The transform equations take one of three forms, containing either 1, 2 or 4 multiplications. This enables the computations to be reduced to 22 multiplications and 28 additions (subtractions). This represents a significant improvement over the INMOS design [5] which requires 64 multiplications and 64 additions.

The following equations are derived from Figure 3.

$$s1 = x(0) - x(7) \qquad s2 = x(1) - x(6) \qquad s3 = x(2) - x(5) \qquad s4 = x(3) - x(4)$$

$$a1 = x(0) + x(7) \qquad a2 = x(1) + x(6) \qquad a3 = x(2) + x(5) \qquad a4 = x(3) + x(4)$$

$$s5 = a1 - a4 \qquad s6 = a2 - a3 \qquad s7 = a5 - a6$$

$$a5 = a1 + a4 \qquad a6 = a2 + a3 \qquad a7 = a5 + a6$$

$$X(0) = A * a7$$

$$X(4) = A * s7$$

$$X(2) = C * s6 + B * s5$$

$$X(6) = C * s5 - B * s6$$

$$X(1) = D * s1 + E * s2 + F * s3 + G * s4$$

$$X(5) = F * s1 - D * s2 + G * s3 + E * s4$$

$$X(3) = E * s1 - G * s2 - D * s3 - F * s4$$

$$X(7) = G * s1 - F * s2 + E * s3 - D * s4$$

where: $A = 0.176776$, $B = 0.230969$, $C = 0.095670$, $D = 0.245196$, $E = 0.207867$, $F = 0.138958$, $G = 0.048772$

---

5. BIT SERIAL DESIGN

There are a number of different ways in which the above equations could be implemented. The final design was adopted primarily because it had the highest throughput rate, the simplest timing and control circuitry and could be easily adapted to 2D transform calculations. This design requires 30 cycles of the 15 MHz clock, which results in $30 * 66.7 = 2001$ nsecs to perform a single transform. Timing and control circuitry for sequencing the data through the chip is achieved by routing the two clock phases throughout the design rather than enabling different sections at different times as in the bit parallel design. Although this bit serial design is slightly larger in size than alternative bit serial designs investigated, it still fits within the $6.8 * 6.8$ mm allocation.

Now since the multiplication coefficients are already known, and multiple data streams are processed in parallel, these coefficients can be incorporated on-chip, rather than resorting to a generic multiplier which could cater for more than one coefficient. This leads to a simpler design.

From Figure 4, we see that each generic bit serial multiplier stage requires an adder together with associated AND and delay elements. The coefficient enable AND-gate can be removed, as each multiplier unit which has the coefficient enabled is simply connected to the above data line.

The throughput of this design can be improved further by realising that several consecutive adders compute results during a single clock period. Accordingly, the delay elements between the adder cells and the data stream can be removed, as indicated in Figure 5.

Figure 6 shows a specific multiplier cell for the coefficient 10101_2. In stages where the coefficient is zero, the adder and AND cells can be removed and replaced with delay cells for the data transfer. This

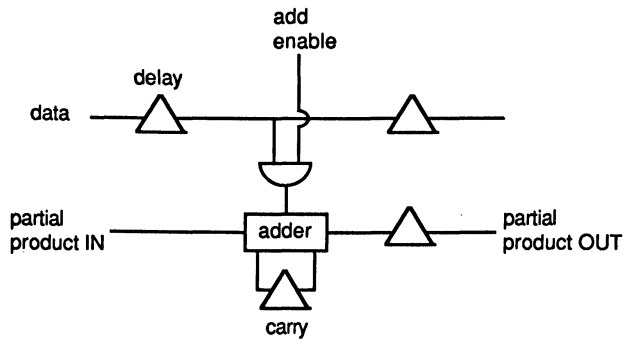


FIGURE 4 Bit Serial Multiplier Cell

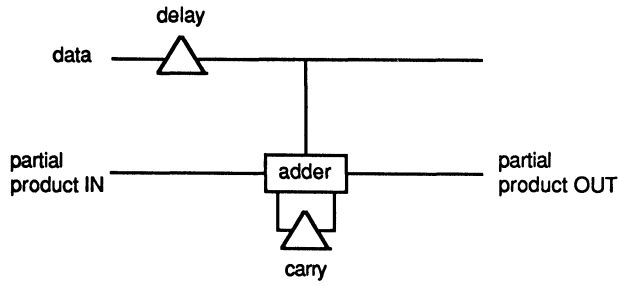


FIGURE 5 Modified Multiplier Cell

greatly reduces the amount of circuitry required. Thus, 22 generic multipliers would require more circuitry than 22 *specific* multipliers, with some of these duplicated in order to enable parallel processing.

6. RESULTS AND DISCUSSION

The first design considered for implementing the DCT chip was a bit parallel one. It was found that *two* ALUs are necessary in order to perform DCTs in real time [8]. With a single ALU the best possible performance to perform a single transform was 2468 nsec, which exceeds the 2104 nsecs required to perform a transformation in real time. Therefore, a second ALU would be required to enable the calculations to be performed in parallel and to enable the processing to be within the real time constraint. Moreover, the limited silicon real estate at our disposal prohibited any further consideration of a parallel solution.

Several bit serial designs were then considered, each of which used a refined version of Lee's algorithm, in order to improve computational efficiency. The final design was selected on the basis of having the fastest throughput, simplest timing and control circuitry and easy extension to 2D transforms, despite its slightly larger size.

The Lee algorithm, in both original and refined forms, was verified via C code implementations on a SPARCstation (by confirming that a DCT cascaded with an inverse DCT produced the original data set). The bit serial DCT chip cycle time to process eight data points comprises:

(a) 8 cycles to read(write) data to(from) the chip, plus

(b) 21 cycles to cycle the data through the ALU, plus

(c) 1 more cycle to write the last data bit.

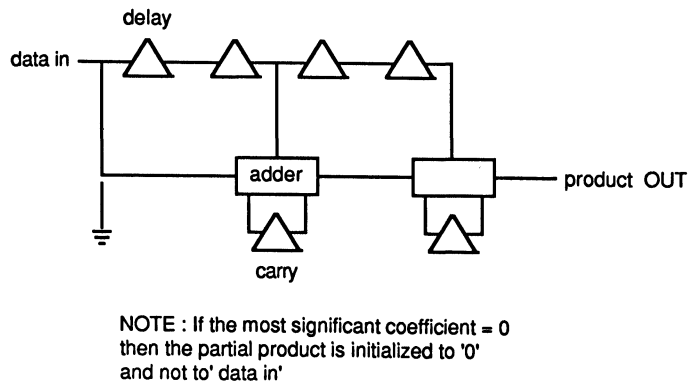


FIGURE 6 Specific Multiplier Cell for a coefficient of 10101_2

The DCT chip uses an integer multiplication unit, and thus will lead to roundoff errors. The maximum expected error occurs when multiplying the largest result, coefficient $x(0)$, by half the multiplier accuracy:

$$(255*8)*(1/256)*(0.5) = 3.9843$$

Actual tests using TREK (see below) revealed that an error of 2.0 would be more typical in practice.

SPICE was only able to be used with individual modules of the DCT chip design, not with the entire design. In this manner, we were able to conform that the following modules performed as expected:

(i) multiplier cell,
(ii) delay buffer,
(iii) clock driver, and
(iv) input & output buffers.

The only extensive testing which was possible using the VLSI design tools at our disposal involved TREK—the switch level simulator. Data sets were run using the TREK simulator to verify that the design were correct. The data simulated a normal 1D transform, transforming the results of the 1D transform to simulate a 2D transform, interrupt control of input and output control lines, cascading two transforms in order to demonstrate the chip's cycle performance. All these tests were verified with a C program to ensure the correct results were obtained.

7. CONCLUSION

The final chip design had the following characteristics:

(i) 15 MHz master clock which is supplied from the master processor,
(ii) parallel data IO (requiring 8 clock pulses each),
(iii) two IO buffers, for storage of both the (eight 12-bit) input and transformed data,
(iv) single-pass (integer) ALU,
(v) 8 clock pulses required to read(write) data to(from) the chip,
(vi) 22 clock pulses required to transform the data, and
(vii) occupies 5*4 mm of silicon (cf. 6.8*6.8 mm allocation).

Thus the time required to compute a 1D DCT transform of an 8 point DCT is 2001 nSec, which increases to 32 mSec for a 2D transformation, (an 8*8 point DCT). The design objectives were satisfied, primarily the ability to perform 1D DCTs in real time. For the purposes of comparison, Table 1 shows the performance of our bit serial DCT design with earlier bit parallel ones [10, 11]. Allowing for restrictions in transistor count and clock rate, the bit serial design exhibits throughput comparable with the bit parallel chips.

Even though our final bit serial design was not committed to silicon, we were able to gain significant insights into the advantages of various bit parallel and bit serial solutions to this problem. Based on our experience, we conclude that bit serial is the preferred approach for this application. Indeed, this confirms previous work which purports that bit serial designs are admirably suited to applications involving computationally complex tasks, exhibit a high degree of concurrency, have a large dynamic range, yet have a narrow signal processing bandwidth [12]. We concur from our present study that the advantages of bit serial designs are their efficiency (in terms of speed/area product) and simple interconnections (which has significant repercussions in realisation in VLSI form).

TABLE 1
Comparison of DCT VLSI Designs

	CHIP		
	Vetterli and Ligtenberg	Totzek and Matthiesen (Siemens)	Crook and Fulcher
Operation Mode	1D	2D	1D
Algorithm	butterfly/ rotator	matrix-vector multiplication	modified butterfly
Arithmetic	bit parallel	bit parallel	bit serial
Wordwidth-	12-bit	12-bit	12-bit
data (constants)	(10-bit)	(12-bit)	(8-bit)
Throughput	8.33Mwords/sec	> 45Mwords/sec	4.0Mwords/sec
Clock Rate	?	> 45MHz	15MHz
Technology	2.5 μm CMOS full custom	1.5 μm CMOS full custom	2.5 μm CMOS full custom
# Transistors	34,000	284,000	7,100
Area	35 mm^2	92 mm^2	25 mm^2

Acknowledgments

This work was conducted as part of the Advanced Telecommunications Research Program at the University of Wollongong. The authors would also like to thank Chong Hee Tan and Prof. Graham Hellestrand of the University of New South Wales for supplying the VLSI design tools used in this project, Van Dao Mai of the University of Wollongong for porting these UNSW VLSI tools, and to Australian Silicon Structures for the donation of the ES2 Solo 1400 VLSI design tools.

References

[1] Joint Photographic Experts Group Standards, J.P.E.G., 1990.

[2] "Video Codec for Audiovisual Services of x64 kbit/sec" *CCITT Recommendation H.261*, 1990.

[3] B.G. Lee "A New Algorithm to Compute the Discrete Cosine Transform", *IEEE Trans. Acoustics, Speech & Signal Processing* **ASSP-32** 6 December 1984 pp. 1243–1245.

[4] H. Wu, "The Implementation of Multidimensional Discrete Transforms For Digital Signal Processing", *PhD Thesis, Dept Electrical Engineering, University of Wollongong*, 1990.

[5] INMOS *IMS A121 2D DCT Image Processor Advanced Information* 1986.

[6] Joint Microelectronic Research Centre, University of New South Wales, *VLSI Design Tools*, 1990.

[7] European Silicon Structures *Solo 1400 Reference Manual* 1989.

[8] E. Arould & J. Dugre, "Real Time Discrete Cosine Transform, an Original Architecture", *Proc. 1984 ICASSP—IEEE Intl. Conf. Acoustic Speech & Signal Processing*.

[9] N.H.E. Weste & K. Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective*, Addison-Wesley, 1985.

[10] M. Vetterli & A. Litenberg, "A Discrete Fourier-Cosine Transform Chip", *IEEE J. Selected Areas Communications* **SAC-4** (1) January 1986 pp. 49–61.

[11] U. Totzek & F. Matthiesen, "Two-Dimensional Discrete Cosine Transformation with Linear Systolic Arrays" in *Systolic Array Processors* Prentice-Hall, 1989, pp. 388–397.

[12] J. Fulcher, "Application of Bit Serial Architectures", *J. Electrical & Electronics Engineering, Australia* **10** (2) June 1990, pp. 88–94.

Biographies

JOHN FULCHER is currently a Senior Lecturer in Computer Science at the University of Wollongong, Australia, as well as the Director of the Neural Network Research Group within the University's Centre for Information Technology Research. In this latter capacity, he heads a project team researching face recognition for airport security (funded by SITA, the worldwide airline cooperative).

John teaches a graduate course in ANNs as part of the Intelligent Systems research program in Computer Science at the University of Wollongong. He also teaches undergraduate courses in computer architecture, computer systems and microcomputer interfacing & real-time computing. The course notes from this latter course formed the foundation for his bestselling Addison Wesley book *An Introduction to Microcomputer Systems: Architecture and Interfacing*.

John is a member of ACM, IEEE Computer Society and the International Neural Network Society.

DAVID CROOK is currently a research student at Wollongong University undertaking a PhD. in robotics. He earned both his B.E., (Mech), 1985, and MSc. Hons (Computing Science), 1992 at Wollongong University. He undertook both these degrees on a part time basis while working at the Broken Hill Proprietry, Slab and Plate Products Division, firstly as a Mechanical Engineer and then a Software Engineer. His current research interests involve improved adaptation of ultrasonics in the robotics field in particular object shape recognition.