A VLSI Display Controller and Processor

Adrian Gay

Personal Systems Products

IBM United Kingdom Laboratories Ltd
Hursley Park,
Winchester,
England SO21 2JN

Abstract

The heart of the IBM PS/2 Image Adapter [2] is a VLSI display controller and processor chip utilising IBM's proprietary advanced 1 micron standard cell CMOS technology. This chip is 9.4 mm square and contains the standard cell equivalent of 40000 two-input NAND gates, organised as a matrix of 27000 wireable cells. It is packaged in a 194-pin IBM Pin Grid Array and consumes approximately 1 watt.

The display controller and processor chip contains most of the digital logic which gives the IBM Image Adapter its very high performance and function, requiring only the following external components:

- High current driver/receivers which interface the PS/2 Microchannel
- Video memory RAM modules (VRAMs)
- Programmable read-only memory (PROM)
- Processor instruction memory modules (DRAMs)
- A Display Support (DS) chip, also described in this paper
- The serialiser/palette/DAC (SPD) chip described in [3].

The DCP chip provides a 'programmable adapter' which can be personalised for image, text or graphics simply by providing a new microcode load for the on-chip processor. The IBM PS/2 Image Adapter performs algorithms such as image compression and graphics rotation at rates comparable to hardware, while retaining the flexibility of programmable microcode.

A complete set of tools is provided for the processor, including a 'C' language compiler, assembler, linker, simulator and debugger.

The VLSI display controller and processor chip has been designed using the IBM Hursley VLSI design process described in [4].

Introduction

The IBM Development Laboratory at Hursley has been involved in the design and development of display products for many years. Recently it has developed adapter products for the IBM PC and IBM PS/2 personal computers.

The Display Controller and Processor (DCP) chip described in this paper has been developed for application in IBM PS/2 adapters for display, manipulation, scanning and printing of pixel data. It is used in the recently announced IBM PS/2 Image Adapter, to support a high-resolution display at 1600 x 1200 pixels, document scanners, and high quality laser printers.

The 9.4mm square chip is fabricated in IBM proprietary standard cell CMOS technology and packaged in a 194-pin pin grid array. It contains an equivalent of 40000 2-input NAND gates and consumes approximately 1 watt. A second 84-pin PLCC CMOS gate array, the DS chip, is required when a display monitor is supported.

Both the DCP chip and the DS chip were designed in a High Level Design Language and synthesised to standard cell logic. This approach enabled a small design team to achieve the level of performance and function reported here, without resorting to custom silicon design.

Functional Overview

The DCP chip and its DS chip contain a number of functional islands, all individually designed to well-defined common interfaces, illustrated in Figure 2. These functional islands are:

- An interface (IF) to the PS/2 Microchannel fully exploiting the capabilities described in [1].
- A very high performance 32-bit Reduced Instruction Set processor, referred to as the Application Processor (AP).
- A Pixel Interface (PI) that allows the processor to access memory (either on the IBM Image Adapter card, or the PS/2 system board, or in storage on the PS/2 Microchannel), as an X,Y addressed array of pixels with a variable number of bits per pixel.
- The Local Bus Controller (LC) A flexible, configurable bus controller that allows external memory (DRAM or VRAM), and I/O devices, to be directly attached to the DCP chip.
- A CRT controller (CRTC), that generates line and frame synchronisation signals for the attached monitor, and making synchronous memory requests to supply the serialiser with suitably timed video data.
- A video multiplexer (VMX) to supply two sources of data to the Frame Capture function of the Pixel Interface.
- A format converter that handles the different data and pixel formats of Intel and non-Intel processor architectures.

Functional Islands

PS/2 Microchannel Interface (IF)

The IF interfaces the busses and control signals of the DCP chip to the PS/2 Microchannel as a 16-bit device, allowing the AP and PI to access system memory. The IF also interfaces the Microchannel to the DCP Local RAM and I/O space. The DCP chip can operate as a bus master device.

Local RAM can be configured as a variable size block of storage, from 4K bytes to 8M bytes, anywhere in PC system memory space (limited by 24-bit addressing). If the PC address space occupied is less than the actual amount of local memory, then a Local RAM base register is used to position the visible 'window' into Local RAM.

The AP instruction store can be mapped in place of Local RAM, so that AP microcode can be loaded by positioning the RAM 'window' and performing a string move or similar operation.

The only external components required are the following TTL drivers/transceivers of the Microchannel signals that cannot be handled with CMOS drive capabilities:

```
74ALS245 transceiver  (5 off)
74F245   transceiver  (1 off)
74AS757  driver       (1 off)
74AS04   driver       (1 off)
```

Reference [1] describes the theory and operation of the PS/2 Microchannel in detail.

Programmable Option Select (POS) POS is the PS/2 mechanism that eliminate switches and jumpers from adapters. It allows multiple identical adapters to coexist, and also allows system identification of attached devices, so that automatic configuration becomes possible without intervention by the customer. An identification number can be read from the adapter and software-writeable registers are provided to modify the configuration.

PC adapters normally have an on-card ROM containing the Power-on Self Test and initialisation software for the adapter. The DCP chip performs the address decode for this external ROM and provides a single external select line. Because all adapters must share the Microchannel ROM address range, the DCP chip provides a paging scheme so that only 8K bytes of this space is occupied by the ROM, thus allowing more adapters to co-exist. In addition, this 8K byte region, and the I/O registers of the DCP chip, can be relocated, thus allowing automatic configuration.

Interrupt Control I/O registers interface all the internal sources of interrupts in the DCP chip to the PC. The source of interrupt is identified, and all interrupts can be reset and masked individually.

AP Control The PC has access to the AP's Instruction Address Register, Status Flags, Stack Pointer, and the contents of the Stack. The AP can be single-stepped by complete instructions or run continuously. When stepping, the AP executes one complete instruction (of one or more cycles) and then enters the stopped state. Because a PI operation may take many cycles to complete, status is provided to indicate that the AP/PI is still in the process of executing an instruction, or that the AP/PI has completed the instruction.

AP and PI Bus Mastership Both the AP and PI can access 16M bytes of Microchannel memory space (24-bit addressing) using the Microchannel arbitration and mastership mechanism. The IF contains arbitration logic giving the AP and PI equal priority. If they both require access at the same time they are allocated alternate cycles.

The Microchannel allows a burst mode of operation. Both the AP and PI supply the IF with a signal indicating that multiple accesses are required, allowing the IF to maintain burst mode. The PI automatically holds its memory request during multi-pixel operations. However, the AP requires look-ahead logic to determine if the next AP instruction also requires a PC memory access.

Local Bus Controller (LC)

The LC provides access to Local Memory and I/O attached to the Local Bus (for example, SPD, attached scanners, printers, etc.), from the DCP AP and PI, and from the PS/2 Microchannel. The LC also performs instruction fetching for the AP from its separate Instruction RAM.

The LC supports either DRAMs or VRAMs allowing both display and/or non-display oriented implementations of local RAM, and RAM chip organisations of either 64K x 4 or 256K x 4 RAMs. The chip was designed before 1M bit chips were available, so support for 256K bit RAMs was included should cost and availability of 1M bit RAMs prohibit their use in the product timeframe.

Although the DCP chip works with 32-bit addresses, physical pin limitations of the current version of the AP limits the addressing of memory spaces as follows:

- AP Instruction RAM - 20 bits (1M byte)
- Local RAM - 23 bits (8M bytes)
- Microchannel memory - 24 bits (16M bytes).

The LC provides two modes of memory operation of AP Instruction RAM (IRAM) and Local RAM (LRAM). '1/1 mode' runs both IRAM and LRAM at the same clock rate. '2/3 mode' runs IRAM at at faster clock rate than LRAM, in the ratio 2:3. This option allows the system designer to trade off both the speed of instruction fetching and data access, and the cost of RAM chips, depending on the application.

The LC provides I/O to support a variety of external I/O devices. These can either operate at the speed of the Local Bus (termed 'fast I/O'), or provide a READY signal to the LC (termed 'slow I/O'). In addition to access by the AP, I/O devices on the local bus are directly accessible by the Microchannel.

Requests for accesses on the Local Bus can come from the AP or PI, or from the PS/2 Microchannel. The LC arbitrates between all requests for the Local Bus using a fixed priority order of each type of access.

1. CRTC VRAM transfer cycles
2. RAM refresh
3. Microchannel memory requests to LRAM
4. AP or Microchannel I/O requests
5. AP memory requests to LRAM
6. PI memory requests.

Local RAM Controller For memory accesses to LRAM on the Local Bus, the LC translates a linear byte address into byte enables, column address, row address, RAS address and CAS address depending upon the RAM configuration set in a control register. Physical card space, chip pins and CMOS drive capability limits the number of local RAM chips to a maximum of 64. Thus the physical local RAM addressability is 21 bits and 23 bits for 64K x 4 and 256K x 4 RAMs respectively.

LRAM is run in page mode whenever possible, for multiple pixel access by the PI. Page sizes are 1K bytes using 64K x 4 RAMs, and 2K bytes using 256K x 4 RAMs. Two-way interleaving is also used to further improve performance of multiple accesses.

Instruction RAM Controller IRAM is word organised. For instruction fetches, the LC translates a 20 bit byte address into column address, row address, RAS address and CAS address.

IRAM is run continuously in page mode, until a page fault occurs due to a sequential instruction fetch across a page boundary, or a sequencer instruction generates a jump address outside the current page, or RAM refresh occurs. Page sizes are 512 bytes using 64K X 4 RAMs, and 1K bytes using 256K X 4 RAMs.

Application Processor (AP)

The DCP Application Processor (AP) contains the following hardware elements:

- 32-bit ALU providing arithmetic and logical functions
- 16 word by 32-bit 4-port register file
- 16 special purpose registers
- Single cycle barrel shift/rotate unit
- Sequencer for program branching and looping
- 128-entry subroutine call stack
- 16 pixel registers
- Additional pixel instruction set
- 32-bit multiplier/divider
- External instruction and data memory busses
- External I/O bus
- Interrupts.

Processor Registers The AP has three distinct sets of registers. Sixteen 32-bit General Purpose Registers (GPRs) are provided for data manipulation and addressing, with no 'restrictions' dependant upon use.

A second set of sixteen registers provide a number of specialised processor functions including, fast loop counting, interrupt control and Stack manipulation. Two of these registers, Result Hi (RH) and Result Lo (RL) return results from multiplication and division instructions. Another, the User Flag Register, contains 16 bit-addressible flags which provide single-instruction test-and-modify or test-and-branch.

The third set of sixteen registers are the Pixel Registers, that hold pixel addresses, pixel map parameters, pixel values, and so on.

Addressing Modes The AP has 6 distinct address spaces:

- Program (32-bit linear addressability)
- Stack (128 entry)
- LRAM (32-bit linear addressability)
- Microchannel memory (32-bit linear addressability)
- Microchannel I/O (64K bytes)
- Local I/O (128 word).

Code for the AP resides in the Program space and the processor only has read access (instruction fetch) to this space. Immediate operands also reside in program space.

Program branching using either absolute 32-bit addressing, or Instruction-Address-relative 16-bit addressing, or Instruction-Address-relative 8-bit addressing is provided.

Absolute instruction addresses are multiples of bytes, although internally, instructions themselves are multiples of words. Relative displacements are multiples of instruction words.

The hardware Stack contains 128 entries and is used to push/pop subroutine return addresses. It is not visible directly to the AP programmer. A separate pop function is provided to allow subroutine returns to be bypassed. The Stack Pointer value can be accessed. Although 128 entries will be sufficient for typical DCP applications, recursive routines should read the Stack Pointer on entry and check for possible overflow, and as necessary, continue operation with a software 'call' stack in Local or Microchannel memory.

The AP has 32-bit byte addressability to Local RAM. In practice, the most significant address bit is used to select between Microchannel memory and Local memory. The Local RAM space occupies the first half of the AP's 32-bit memory address space. The AP/PI can address the entire Microchannel memory space of 16 M-bytes (24-bit addressing). The AP/PI can also address the entire Microchannel I/O space of 64K bytes.

The AP has its own separate I/O space, partly for implementation of I/O Devices, and partly to allow access to the special-purpose and pixel registers, without expanding the 16-bit Instruction Word. This I/O space is 128 addressable locations. It provides access to a number of 'devices' and contains the special purpose and pixel registers. The remaining I/O space is available for external I/O device addressing.

Instruction Set The AP processor 16-bit instructions can be classified into the following different types.

- Register
- Register Register
- Register Immediate
- Multiply/divide
- Shift/rotate
- Register Memory
- Register I/O
- Sequencer Operations.

Register instructions and Register Register instructions perform the traditional operations on GPRs.

Register Immediate Quick instructions perform an arithmetic operation between a GPR and a 'quick' immediate value. This quick immediate value is a 6-bit 2's complement number supplied in same Instruction Word, sign-extended to 32 bits prior to the function being performed.

Register Immediate Word and Long instructions perform an operation between a GPR and a 'word' or 'long' immediate value. The word immediate value is a 16-bit 2's complement number supplied in a second Instruction Word following this instruction, sign-extended to 32 bits prior to the function being performed. The long immediate value is a 32-bit 2's complement number supplied in a second and third Instruction Word following this instruction.

For multiplication, the two source operands are GPRs. RH returns the most significant word of the 64-bit result, and RL returns the least significant 32 bits of the 64-bit result. For 32/32 division, the two source operands are GPRs. For 64/32 division, the 64-bit numerator is contained in an even/odd GPR pair. RL returns the 32-bit quotient and RH returns the 32-bit remainder.

The multiplication and division process operates in parallel with the ALU and sequencer, so that the AP can continue executing instructions. A hardware interlock is provided to 'hold' the AP if an instruction accesses RH/RL.

Bit-oriented Instructions are provided to set, clear, invert and test single bits in a GPR, and set, clear and change individual User Flags, and register bits. The Zero Status Flag is updated to reflect the state of the specified GPR bit or User Flag PRIOR to the instruction execution. This allows bit testing and modification to occur in a single instruction cycle.

Shift and Rotate instructions shift/rotate the contents of a GPR from 1 to 31 places. A barrel shifter is used so that these instructions take a single cycle to shift regardless of the number of shifts. On every shift/rotate instruction, the Shift Extend Register (SXR) is loaded with the bits shifted/rotated out of the GPR, and has the effect of a variable-length 'holding' register. The SXR contents can be shifted/rotated into another register in a single cycle. 'Shift with Extend' instructions perform multiple register shifts, at one instruction cycle per register.

Register Memory instructions transfer data between a GPR and memory, either in Microchannel memory or DCP Local Memory. When reading memory, bytes and words are either zero-extended or sign-extended to 32 bits.

Memory addressing modes provided are:

indirect
indirect with post-increment
indirect with pre-decrement
indirect with 16-bit displacement
indirect with 32-bit displacement
absolute 32-bit memory address

Memory writes are overlapped with other AP operations, so that instruction execution can proceed without waiting for a 'ready' signal. A hardware interlock is provided should subsequent writes occur before the current access is complete.

Register I/O instructions transfer data between a GPR and an I/O location. The function performed will depend upon the I/O address that is accessed.

Sequencer instructions are used to change the flow of control of the AP program. These instructions take one or two instruction cycles to check the condition, calculate any resultant program address and take the specified action. If the result of the operation changes the Processor Instruction Address, a second cycle is needed to perform the address calculation and update the IAR. Otherwise the sequencer operation takes a single cycle.

All sequencer instructions are 'conditional' ie. the operation specified only takes place if the condition is satisfied. 32 conditions are available, including the state of the individual User Flags, and 5 External Conditions. The External Conditions can be connected to I/O device 'data ready' signals, for example.

Interrupts The AP has the following interrupt capabilities:

- Interrupt the PC and test for reset
- Be interrupted by the PC and reset the PC interrupt
- Be interrupted by an external device and reset it
- Be interrupted on specified Microchannel activity.

PC Interrupt The AP can set an interrupt to the PC. This interrupt is maskable by the PC and can only be reset by the PC. The AP can test the state of this interrupt directly using one of the External Conditions available to sequencer instructions. Attempting to generate subsequent interrupts will 'hold' the AP until the previous interrupt has been reset by the PC.

AP Interrupts An AP interrupt takes effect at the end of the currently executing instruction. When an interrupt is taken, the current value of the IAR is pushed to the hardware Stack, processor status is saved, further interrupts are disabled, and the IAR is updated to the single Interrupt Vector location. Any other hardware registers used during interrupt service must be saved and restored by software. Subsequent interrupt conditions are latched, but do not generate interrupts until interrupts are re-enabled. At the end of the interrupt service routine, a 'return from interrupt' instruction is used to restore status, pop the return address off the Stack, re-enable interrupts, and return to the point at which the interrupt occurred. Nesting of interrupts can be implemented in software by maintaining a software 'interrupt stack'.

The presence of a PC to AP interrupt can be tested directly by the AP using one of the External Conditions available to sequencer instructions. This interrupt is particularly useful for implementing 'interruptible' pixel algorithms, without additional code overhead.

External Interrupt sources are connected via a pin on the DCP chip. This signal will operate as either level sensitive or pulsed. Multiple sources must be handled with off-chip priority/vectoring logic. An External Interrupt Reset signal is provided.

"Snooping" is the term used to describe the mechanism in DCP that monitors I/O activity on the Microchannel and generates AP interrupts when read and/or write accesses occur within a range of Microchannel memory or I/O addresses specified by software.

Direct Data Bus

The Direct Data Bus (DDB) is an independent bi-directional 11-bit bus. It is connected to the AP as a high-speed output bus and to the PI as the Frame Capture input bus. The bus signals are as follows:

- CLOCK (1)
- DATA VALID (1)
- DATA (8)
- SYNC (1).

DDB output is written by the AP. Each write is split into bytes that are sent out on the DDB at twice the AP clock rate. DDB input on the DATA pins are captured at the rate determined by the asynchronous CLOCK signal, when the DATA VALID signal is active. The Frame Capture PI operation will terminate when the DATA VALID signal goes inactive.

Pixel Interface

Pixel Maps and Memory Organisation From a processor's point of view, memory is organised as a linear array of bytes, words, double-words, and so on, linearly addressed by a byte address. From a programming point of view, a 2-dimensional array of pixels, addressed by X,Y coordinates, is required. Software is required to convert between X,Y pixel addressing and linear memory addressing when the application updates pixels stored in memory. This conversion is complicated by the fact that pixels may be of varying sizes (bits/pixel) and thus a byte may contain more than one pixel (eg. 2 pixels at 4 bits/pixel), or a pixel may occupy more than one byte (eg. 16 bits/pixel). More than one size of pixel may need to be supported. In addition, moving from one pixel position to another in the X,Y array requires calculating addresses dependent on the X dimension of the array and the number of bits/pixel.

The PI provides the DCP processor with a pixel memory addressing mechanism, in addition to the normal linear mechanism, so that it can manipulate Pixel Maps that reside in linear memory, without the software overheads described above. Pixel maps are defined by:

- A Pixel Address
- A Pixel Map Width
- The number of bits/pixel.

The Pixel Address is effectively the X,Y address of a pixel within a pixel map. Although the physical location in linear memory varies with the pixel size, the Pixel Address remains the same, making the Pixel Map device-independent. During pixel drawing operations, the PI will automatically update the Pixel Address to one the 8 neighbouring pixels of the current pixel.

The Pixel Map Width is specified as a number of pixels and the PI supports pixel sizes of 1, 2, 4, 8 and 16 bits/pixel.

The PI has access to all memory, both in Microchannel memory and local to the DCP chip. All memory is pixel memory when using the PI. Pixel maps may be manipulated in local display memory (VRAM), other local memory (DRAM), and, as DCP is a bus master, memory on the planar, memory extension cards, and even other adapters.

The PI has the concept of a source and a destination for pixel operations. These can be either a fixed pixel value, varying pixel values from or to the AP, and pixel values within a pixel map.

To support "Pixel Map to Pixel Map" operations, the PI has two sets of registers containing the Pixel Address, Pixel Map Width and Number of Bits/pixel; one set for the source and the other for the destination.

Pixel Values and Update Mixes On every pixel operation, the source pixel value is combined with the destination pixel value according to a Mix function. In the simplest case, the mix would specify that the source overwrites the destination. 16 logical, and 13 arithmetic mix functions are provided.

Colour Compare functions are provided that determine whether an old pixel is updated by the new pixel to be drawn, either based upon number of different comparisons of the old pixel value and/or the new pixel value with a fixed comparison value, or whether the new pixel is to be transparent.

Draw Pixel and Step Many image and graphics algorithms perform some operation on one pixel and then need to step to one of the 8 adjacent pixels. Draw Pixel and Step allows this to be done in a single instruction and the linear RAM address is automatically maintained. The direction step may be fixed, or selected by the Status Flags resulting from AP instruction execution.

Read Pixel and Step This function reads a pixel from the current Pixel Address and has the same automatic pixel address stepping mechanism as Draw Pixel and Step.

Coded Direction When operating incremental line drawing algorithms, the decision of whether to step axially or diagonally is taken by looking at the sign bit of some error measure. It is practical to precompute these 'direction steps' for 'short' lines and store them as binary strings in a table. When a short line is found, the algorithm set up can be bypassed and table lookup used instead. This significantly improves the overall drawing performance when set up time predominates, which is often the case.

The lookup table contains binary strings that are written to the PI. A string is interpreted as 'Draw Pixel and Step', with each binary digit selecting an axial or diagonal step between pixels.

Draw Pixel Pattern

For text characters, a bit map type of definition is used in which each row of a character cell is defined by a bit string. Each bit defines whether that pixel in the character should have foreground or background colour and mix. The Draw Pixel Pattern function performs the conversion from bit-map to pixel values and update mix functions. From 1 to 16 bits at a time are supplied from the AP to the PI, that expands each 1-bit source pixel to 2, 4, or 8 bits, draws it into the pixel map and automatically steps the pixel address.

Bit Boundary Block Transfer (BITBLT)

Bit Boundary Block Transfer (BITBLT) is a function for moving a rectangular array of pixels (of arbitrary number of bits/pixel) from a source to a destination. The combinations of source and destination are:

- Local RAM to Local RAM
- Local RAM to Microchannel memory
- Microchannel memory to Local RAM
- Microchannel memory to Microchannel memory
- Video source to Local RAM (see "Frame Capture")
- Video source to Microchannel memory (see "Frame Capture")

BITBLT is used for transferring images to/from Microchannel memory and a pixel map (in Local RAM), drawing image-type characters (bi-level or multi-level) from definitions stored in Local or Microchannel memory into a pixel map, saving and restoring part of the screen image when using pop-up menus, multiple bit map management, and so on.

The BITBLT hardware contains a 512-bit FIFO buffer that is loaded from the source and dumped to the destination under the control of the AP. The FIFO can buffer 64 8-bit pixels, 128 4-bit pixels, and so on, and is alternately loaded and dumped to keep local memory running in interleaved page mode. The source data can be written at the destination address in any of the 8 possible pixel directions. Pixel addresses are automatically updated.

Incremental Algorithm Assistance Many incremental algorithms for generating pixel primitives use a technique that measures the error between the position of the current pixel drawn and the "true" geometric representation of the primitive. The sign of the resulting error measurement is used to determined the position of the next pixel such that it is "closer" to the true representation. An AP pixel instruction allows a pixel to be drawn according to the error measure, the new error measure to be calculated, and the pixel address to be updated, all in a single processor cycle. This dramatically improves the performance of incremental algorithm kernels.

Frame Capture The Direct Data Bus can be connected as the pixel source, to the BITBLT hardware. Asynchronous pixel data from external devices, including:

- The video output of other adapters
- Broadcast standard video (digitised)
- Image scanners
- Magnetic or optical media.

can be captured directly in DCP memory in real time.

For video capture, horizontal and vertical syncs/blanking are available to the AP, defining the frame to be captured.

The AP sets the number of pixels to be transferred and initiates the transfer from the DDB to Pixel Memory. Pixel values are buffered in the BITBLT FIFO and written to the RAMs 16 bits at a time. The FIFO is unloaded in parallel and ensures that overrun cannot occur when RAM bandwidth is taken by refresh and VRAM transfer cycles.

CRT Controller

The CRTC in the DSC generates the following signals:

- Horizontal (line) sync pulses
- Vertical (frame) sync pulses
- Line and Frame Blanking signals to the AP
- Composite Blanking signal for the serialiser
- RAM addresses for video refresh memory cycles
- Video shift register load requests to the LC.

The following CRTC parameters are programmable:

- Horizontal
  - Sync pulse start
  - Sync pulse end
  - Blanking start
  - Line end (total line time).
- Vertical
  - Sync pulse start
  - Sync pulse end
  - Blanking start
  - Frame end (end of frame time).
- Start address of the display map
- Pixel Map Width
- Shift register length - 128, 256, 512, or 1024 bits.
- Polarity of sync signals
- Line replication.

Video Multiplexor

The video multiplexor in the DSC selects between one of two source 11-bit busses and is used to supply data to the Direct Data Bus of the DCP chip. The polarity of the control signals can be individually inverted under program control. This is especially useful when the polarity of sync signals are used to define the "mode" of the incoming data (multisync monitors). The multiplexor output can be set high impedance, when, for example, the AP is outputting data on the Direct Data Bus.

Data Formats

Byte and Word Ordering The difference between the Intel ordering and other vendor ordering of bytes in physical memory is well known. Non-Intel byte order will be referred to as "IBM" byte order in this section. Basically, bytes, bytes-within-words and bytes-within-dwords, are stored with the least significant byte at the lowest byte address in Intel format and the least significant byte at the highest byte address in IBM format.

A consequence of the IBM data format is that data is always in byte order, whether originally generated as bytes, words or dwords, and subsequently accessed as bytes, words or dwords. With Intel format, however, it is necessary to know how the data was generated in order to access it properly, for example, accessing a byte string using a dword operation (for memory efficiency) results in the bytes in reverse order.

Pixels Traditionally, pixels in a pixel map have been stored such that the 'leftmost' pixel is stored in the most significant bit position(s) of bytes, words or dwords. This will be referred to as the "IBM" format for pixel maps, as recently, an alternative pixel map format has been proposed that takes advantage of the Intel 80386 shifting abilities to provide fast pixel manipulation (BITBLIT) in the absence of special-purpose hardware. This new format will be referred to as the "Intel" format for pixel maps. In this scheme, the leftmost pixel is stored in the least significant bit position(s) of bytes, words or dwords. The differences are illustrated in Figure 1.

It should be noted that IBM data and Pixel formats are identical, and hence use the same mechanism, whereas Intel Data and Pixel formats are not. Pixel formats are identical at 8 bits per pixel, but differ at 1, 2, and 4 bits per pixel.

IBM Pixel Map Format:

	+0								+1								byte no.
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	bit no.
P0								P1								8 bpp
P0				P1				P2				P3				4 bpp
P0		P1		P2		P3		P4		P5		P6		P7		2 bpp
P2	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	1 bpp

Intel Pixel Map Format:

	+0								+1								byte no.
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	bit no.
P1								P0								8 bpp
P3				P2				P1				P0				4 bpp
P7		P6		P5		P4		P3		P2		P1		P0		2 bpp
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P2	1 bpp

Figure 1. IBM and Intel Pixel formats

Format Conversion To provide compatibility with the differing data and pixel map formats between the Microchannel, the AP and the PI, with existing PC pixel maps, and with IBM host-generated data, the DCP chip provides programmer selectable mechanisms for automatic format conversion. The basic assumption in the design is that within the DCP subsystem, all data is IBM format. A bi-directional format converter is provided in the IF that performs 3 independent functions in both directions:

- Byte mapping
- Word mapping
- Pixel mapping.

Byte Mapping converts between Intel and IBM byte order. It controls the mapping of the high and low bytes of the PC bus to the high and low bytes of LRAM, the AP and the PI - either directly connected or interchanged.

Word Mapping converts between Intel and IBM word order (of dwords). It controls the mapping of odd and even addressed words on the PC bus to odd and even addressed words of LRAM, AP accesses and PI accesses - either directly connected or interchanged.

Pixel Mapping converts between Intel and IBM pixel formats. It controls the mapping of the pixel order within bytes on the PC bus to the pixel order of bytes of LRAM, the AP and the PI. Pixels can be either IBM format or Intel format.

Each of these functions can be controlled separately by the PC, AP and PI on a cycle-by-cycle basis across the interface. The converter functions allow data in memory to appear in the right format to both sides of the interface at the same time.

Format conversions of data accessed in DCP Local Memory by the PC, and data accessed in Microchannel memory by the PI are selected independently with control registers.

Format conversion of data accesses by the AP are controlled by high-order bits of the 32-bit memory address being accessed. This mechanism allows each of the AP GPRs to access memory with independent conversions, rather than have a modal mechanism that has to be switched for each type of access, or 16 separate conversion controls that would cost a significant number of logic cells.

Serialiser/Palette/DAC (SPD)

The SPD chip [3] provides all the function of the INMOS G171 component used on PS/2 planars and in the IBM 8514A display adapter.

However SPD provides a further level of integration resulting in a reduction in total product cost and board area, whilst improving flexibility and function over the IBM VGA and IBM 8514A:

- INMOS G171 compatible palette & DAC
- Colour comparators for diagnostic verification of DAC outputs
- 32 bit to 8 bit serialiser
- Cost-reduced DAC current and comparator voltage reference sources
- Clocks and dividers for PS/2 monitors of 25.175, 28.321 & 44.90 MHz.

Additional function and flexibility is provided:

- 256 by 24 palette with 8 bit DACs
- Support for a video data word width of 8, 16 and 32 bits formatted as 1, 2, 4, 8 and 16 bits per pixel.
- Generation of video logic and shift clocks for the CRTC and video RAMs
- Differential current outputs.

Software Development System

A complete software development system is provided for the DCP AP, to be used by systems development programmers. The tools are 'cross' tools in that they host on the PC and target the AP. This toolkit will be available under license from IBM to its business partners to take advantage of the on-chip processor and to off-load function from the main PS/2 processor.

C Language Compiler

The compiler is the standard Whitesmiths C compiler version 3.1 with a new optimising code generator specifically written for the DCP AP. This compiler conforms very closely to the ANSI X3J11 draft standard for C and is validated using the Plum-Hall Validation Suite. ANSI standard and extended libraries, a double-precision floating-point library, and an image subroutine library are provided, and a number of utilities, including a library management system.

Macro Assembler

The macro assembler produces an object file from a source file containing "assembler level" instruction mnemonics. In addition, a set of assembler directives are provided to define symbols, external references, etc., and to control the output listing.

Linker

This program accepts object files, generated by the assembler and compiler, and links them into a single object file, with all external references resolved and relocation to absolute addresses performed as required.

Software Simulators

Two software simulators are available. The first, written by the DCP hardware development group, is the 'yardstick' by which the software development system and the hardware is measured. This simulator accurately simulates all the DCP chip function, and was used to validate the design against the specification and the software tools. Test software is written to the specification using the tools and run on this simulator. The same software is then run on the simulation of the hardware, to ensure that from a 'functional' point of view, all are in agreement. The second simulator is provided to allow source level simulation of object code created by the compiler.

Source Level Cross Debuggers

Two C source level cross debuggers are provided. A hardware debugger runs object code on real hardware, for real debug. A software debugger runs C source code against the software simulator.

Further development will allow all the functions of software simulation, hardware debug, and source level debug, to be performed using a single software tool.

Performance

Performance can be a subjective, and often a sensitive subject. This section gives absolute performance numbers for various aspects of the DCP chip, without comparisons with similar, competitive chips. The main point is the excellent level of performance that has been achieve d with a handful of designers and technologists, using a standard cell, rather than custom, design system. Of course, credit is also due to the IBM Burlington Laboratory, for the CMOS technology and the standard cell definitions themselves.

The DCP chip is a worst case design - the design system enforces this. The basic machine cycle time is 120nS. All chips passing wafer test operate at this minimum performance. Random chips have been (and still are) run at 80ns cycle time, which is a good indication that sorting would yield at least our conservatively estimated percentages.

The effect of the parallelism of the DCP chip is hard to quantify, but the AP, PI, multiplier, and memory subsystem can all operate overlapped. For pixel algorithms, the additional effort of 'arranging' the code to take advantage of parallelism is justified.

Processor Instructions

The processor is hard-wired logic. Thus instruction timing is simply one cycle per 16-bit instruction word. Unary, quick and register-register instructions execute in one cycle. Immediate operand instructions take two cycles for 16-bit immediates and three cycles for 32-bit immediates.

Sequencer instructions have the same timing, plus an additional cycle when the branch is taken, to calculate the target address, except in the case of 32-bit absolute addresses, where only 3 cycles are required for either leg of the branch.

Cell count restrictions limited the multiplier/divider implementation to a 'shift and add' type algorithm (without 'early out') executing at a half cycle rate, resulting in a fixed cycle count of 16. However, processor instruction execution can be fully overlapped with multiplication/division, significantly increasing the effective performance.

Register memory instruction cycle count is the sum of the number of instruction words, plus effective address calculation, plus the number of memory cycles, as shown in the following table:

	Reads		**Writes**		
	All		First	Successive	
	byte word	dword	any	byte word	dword
indirect	2	3	1	2	3
indirect with post-increment	2	3	1	2	3
indirect with pre-decrement	3	4	2	3	4
indirect with displacement 16	3/4	4/5	2/3	3/4	4/5
indirect with displacement 32	4	5	3	4	5
Absolute memory (32)	4	4	3	4	5

Two cycle counts are given for the 16-bit displacement case depending on whether no carry, or carry, respectively is generated when adding the displacement to the contents of the GPR.

For isolated writes, the memory access time is completely overlapped with further instruction execution, saving one cycle.

Note that the cycle counts are processor cycles. The AP memory performance is determined by the AP cycle time and running the AP in '2/3 Mode' results in the better memory performance, due to clock granularity. When accessing Microchannel memory, additional time must be added for the arbitration protocol and bus bandwidth limit.

Pixel Instructions

Pixel performance is determined by the local memory subsystem. Along a scan line, the RAMs are operated in page mode with a 2-way interleave, resulting in 60ns/word accesses. For a single pixel access, one cycle is required.

Draw Pixel and Step (overlapped) takes 120ns. Read Pixel and Step takes 240ns.

Line drawing is a combination of processor instructions to calculate algorithm parameters given two end-points and the pixel drawing loop:

setup 4us

short lines 120ns/pixel
long lines 240ns/pixel

The performance of drawing a run of pixels of a single colour, or a pattern, is simply the memory access time divided by the number of pixels per word:

bpp	ns/ pixel	Mpixel /S
8	30.0	33.3
4	15.0	66.6
2	7.5	133.3
1	3.75	266.6

BITBLT performance is achieved by first loading a run of pixels into the FIFO from the source in page mode with interleave, then reversing the process to write them at the destination, resulting in a peak rate. Page faults when switching to and from reading and writing, and a looping overhead give a realistic average rate:

	peak rate		avg. rate	
bpp	ns/ pixel	Mpixel /S	ns/ pixel	Mpixel /S
8	60.0	16.7	73.1	13.7
4	30.0	33.3	36.5	27.4
2	15.0	66.6	18.3	54.7
1	7.5	133.3	9.1	109.5

The maximum sustainable rate of the Frame Capture hardware (into Local Memory) is 34ns per byte, allowing for the memory bandwidth lost during RAM refresh and 3 VRAM transfer cycles per scan line. Note that the external clock is asynchronous and can be varied from DC.

References

[1] "IBM Personal System/2 Seminar Proceedings," Vol. 5, No. 3, May 1987, IBM Corporation

[2] "The IBM Image Adapter," R. J. Bowater, These proceedings

[3] "A 1um CMOS 128MHz Video Serialiser, Palette and DAC Chip," M. Chesters, These proceedings

[4] "The VLSI 'Silicon Compiler' Design Process," R. M. P. West, These proceedings file

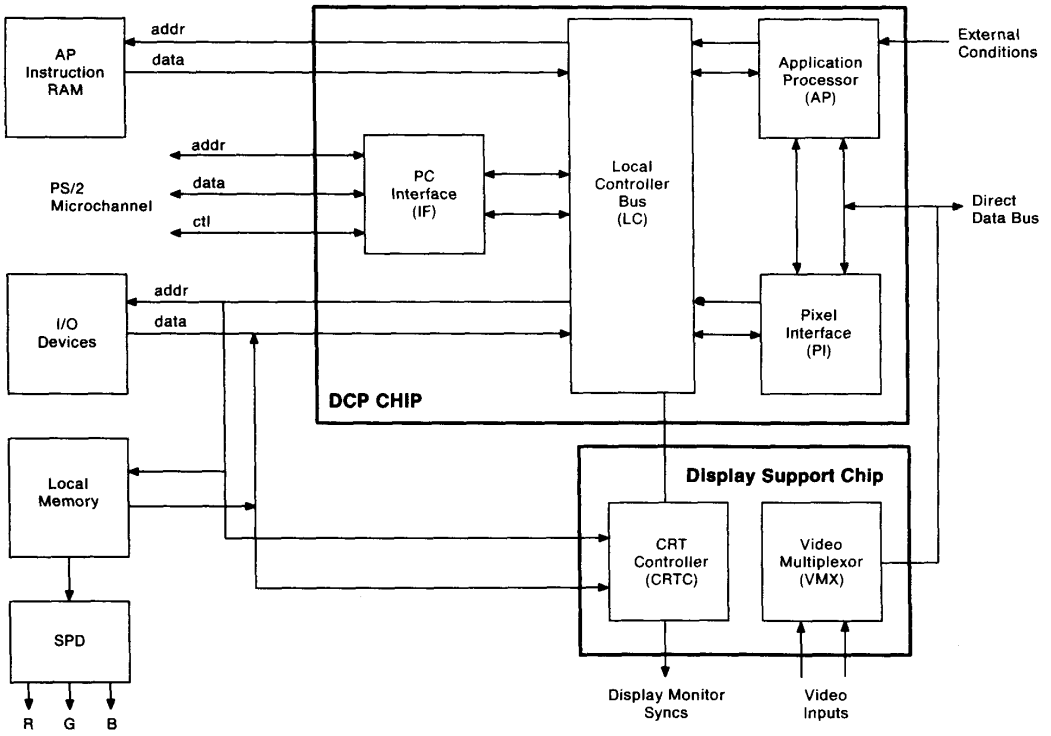

Figure 2.