CSL Notebook Entry

To:	Csl Notebook	Date:	February 5, 1983
From:	Anthony West	Location:	PARC/CSL
Subject:	Dorado Auxiliary Board	File:	[Ivy]<CSL-Notebook>81CSLN-YYYY

XEROX

Attributes: Communication, Database, Distributed computing, Dandelion, Dolphin, Dorado, DES, Cryptography, Encryption, Random Number Generators, Noise Diodes, Stable Storage, Atomic Transactions, WWVB, Broadcast Time Standards, Stable Storage, Universal Identifiers, Synchronizing Events, I/O Interfaces, Intel/IEEE Multibus

References: [Ivy]<CSL-Notebook>Entries>81CSLN-0050

Abstract: This is an early form of the manual describing the Dorado Aux board. It describes the design of the board so far.

1. Overview

The Dorado Auxiliary board consists of a number of disjoint device controllers which share a common interface to the Dorado Slow I/O bus. There are controllers for:

DES Cryptography. The controller hardware allows data to be encrypted according to the NBS Proposed Federal Data Encryption Standard, DES. Pipelined LSI MOS encryption chips, in conjunction with a microcode task, implement memory-to-memory encryption at rates exceeding 12 megabits/second. The encryption logic is duplicated to permit improved error detection. However, in the event of a failure of one of the units, software can reconfigure the controller to permit operation to continue at full speed, but without error detection capability.

Random Bit Generator. A thermal noise diode is used as a white-noise source to generate a stream of random bits at rates up to 100 KBaud. The output is amplified by an operational amplifier to increase the signal level to the point where an ECL comparator can turn the noise into a digital bitstream. The output bits are biassed, however, and it is intended that a suitable unbiassing algorithm will be provided in software.

Electronic Stable Storage. This controller allows records of information to be written into external electronic stable storage modules with negligable latency and at transfer rates up to 30 megabits/second. The modules, based on CMOS RAM, are provided with automatic battery-standby capability to ensure data retention in the event of power failure. The on-board bus interface allows additional storage modules to be plugged in to the external parallel bus cable either to extend the total amount of addressable storage, or to increase the modular redundancy of the storage subsystem. On writing, all modules with the same addresses are written simultaneously; on reading, they are read individually. The external bus is parity-checked, and each module performs label- and CRC-checking on the records for very high-reliability operation. Modules can be removed from or added to the Dorado without affecting the data retention or integrity.

10 megabits/second **Ethernet Controller**. Space has been left on the board for later addition of a 10 megabits/second Ethernet controller based on the VLSI group's Ethernet phase decoder, after it has been is fully debugged and tested. Extra capacity has been provided in the Dorado I/O interface for this controller.

The rest of this manual presents the theory of operation for each controller, and explains the implementation in detail. The text is intended to be read hand-in-hand with the logic diagrams, to which frequent reference will be made.

2. Dorado Slow I/O Bus Interface

The Dorado interface has 5 components:

a. TIOA Decoder
b. IOB Receivers and Drivers
c. Control Logic
d. Next Bus Decoder and Wakeup Logic
e. Dorado Clock Distribution

2.1 TIOA Decoder

Refer to page AUX10. The high-order 4 bits of TIOA are compared against the board's base address, as set by the configuration of the addressing SIP. The output of the comparator, together with the low-order 4 bits of TIOA, enters a MC175 latch, which passes TIOA from *t1* to *t2*, and holds TIOA from *t2* to *t3*. The held versions are decoded by two MC161 decoders to provide for 16 register-select signals, *340-357*. Although the DES and stable-storage logic does not require 16 address selects, the additional flexibility was incorporated to allow other controllers to be added later.

Note that the inverted versions of the low-order bits of Tioa are also available -- they are used as inputs to the DES input fifo, see below.

Additional TIOA decoding logic (MC109) at the bottom of the page decodes the addresses *341-343* as a special case and enables writing into the DES Input fifo, Ififo. The idea here is that the data written into Ififo is tagged with the bottom two bits of TIOA, so that it can be routed to a variety of destinations when it emerges from the output side of the fifo later. (The other half of the MC109 is used in the Wakeup Generator, on page AUX20.)

2.2 IOB Receivers and Drivers

When data is transferred from the processor to the board, the I/O bus, IOB, is received by MC195 buffers, and is bussed on the board onto the board as RIOB. The parity of RIOB is checked, and, in the event of a parity error, bit 15 of the board status register, SR15, is set at *t3*. Further, as an additional concession to paranoia, parity errors prevent the generation of the various WriteEnable' signals which condition registers to be written at *t3*.

Refer to page AUX17. When data is transferred from the board to the processor, data from the board is driven onto IOB by MC164 8-way multiplexors which are activated for addresses in the range *340-347*. Full control logic is provided at the bottom of the page to allow a second bank of MC164's to be added for expansion to addresses *350-357* if required later. Again, 8 registers is generous for the logic on the board at the moment, but 8-way multiplexors were chosen over 4-way multiplexors to permit the addition of additional registers later.

Note that each register which connects to the IOB Multiplexor is expected to generate its own parity.

2.3 Control Signals

Refer to page AUX10. The logic at the bottom of the page consists of the receivers for IOout, IOin, and IOReset, as well as the additional gating which is required to generate the WriteEnable' signals for the on-board registers. There are two points to note:

a. Registers on the board cannot be written if RIOB has bad parity.

b. There is additional decoding on the WriteEnable' for the DES Input fifo, Ififo, which will allow writes when TIOA is in the range *341-343*.

2.4 Next Bus Decoder and Wakeup Logic

Refer to page AUX20. The 4 bits of the NEXT bus, which broadcast the task number of the task which will run in the next cycle, are compared against the configuration of the addressing SIP to see if they match the address of the DES task.

When the current task, about to block, executes an instruction which causes a HOLD, however, NEXT is incorrect since the current cycle will be repeated until HOLD is removed and this is only known too late in the cycle to restore the NEXT bus. Therefore, the signal UsNext?', when examined in the next cycle as UsNow?', has to be gated with LastRepeating to determine if it really is our task running now, or whether we are experiencing an extension of the previous cycle, with NEXT being incorrect.

There are actually two conditions to check to see if our task is actually running in this cycle (see the MC117):

a. NEXT says that it might be our task running now and the last task is not repeating (UsNow?' and NOT LastRepeating), or

b. Our task was running in the last cycle, and executed an instruction which Blocked and Held at the same time, causing a repeat (UsLast' and LastRepeating').

These signals are used below in the Wakeup Generator. Wakeup Requests are only permitted to generate a wakeup if our task is not running or pre-empted (OurTaskIsBlocked'), is not running now (UsNow), is not suspected of being about to run next (UsNext?), and Wakeups are enabled in the board control register, (WakeupsEnabled').

Note that this particular scheme might not permit a wakeup to be generated when NEXT and therefore UsNext? is incorrect. This is hardly critical, however, since all that will happen is that the wakeup will be generated a little later. There are no time-critical controllers on the board (except, maybe, the 10 megabits/second Ethernet interface, which would be buffered).

The idea behind the Wakeup Generator is that the wakeup request will be removed as soon as our task services the board. Presumably the microcode or software will cause the signal which generated the wakeup in the first place to be removed.

It was originally intended to allow a microcode task to be multiplexed between several controllers by latching the Wakeup-request signals entering the MC106 in a WakeupDispatch register, which could be read and dispatched-upon by microcode in one or two microinstrucions. This has not been implemented.

2.5 Dorado Clock Distribution

Refer to page AUX10. Clock distribution is accomplished in the usual fashion for Dorado boards. Note the placement of the SE210 chips which distribute the clocking. First, CLK' is received by an SE231 and an SE210 located as close to the appropriate pins of the edge-connectors as possible. Then, the PrePreClock2' is carried to the centre of the board where it is gated to form PreClock0' and PreClock1' These signals are then distributed to the centres of the 4 quadrants of the board to ensure that the skew between the arrival times of the clocks in the various quadrants is minimised.

3. DES Cryptography

The DES hardware consists of 7 major parts:

a. A shared 16-word DES input fifo, Ififo
b. Dual redundant DES cipher chips
c. A DES input finite-state machine
d. An 8-bit Counter Timer
e. A 16-word DES output fifo, Ofifo
f. A DES output finite-state machine
g. A 64-bit Checksum Unit

3.1 DES Input Fifo

Refer to block diagram page AUX01. RIOB data for the DES hardware is written into the 16-word DES input fifo, Ififo. The control logic for this fifo is on page AUX11. The fifo is constructed of Fairchild F10145A 16-word by 4-bit RAMS (page AUX12), which are dual-ported by the MC158 address multiplexor. The dual-porting is arranged such that RAM cycles take one Dorado Clock, or ~32 nS, with read cycles taking place from *t1* to t*2*, and write cycles from *t2* to *t3*. Each time a read or write occurs, the corresponding Read or Write address counter is incremented.

The logic at the bottom of page AUX11 prevents overflow or underflow of the fifo from occurring as follows:

If the Write address counter and the Read address counter become equal (see the MC166), then either this has happened because the last word was just read out of the fifo and the fifo is empty, or else the 16th word has just been written into the fifo and the fifo is full. The lower MC135 flip-flop, clocked with Clock2', remembers whether the last cycle was a Read cycle or a Write cycle, thus allowing the Full case to be distinguished from the Empty case. IfifoFull and IfifoEmpty are then used at the top left of the page to inhibit further writes or reads respectively. A similar technique is used for the DES output fifo, discussed below.

EnableIfifoWrite', generated by the control logic on page AUX10 when TIOA is in the range *341-343*, allows writes to take place in the fifo at *t3*. Reads from the fifo occur when there is some data in the fifo (ie. NOT IfifoEmpty), and the upper MC135 status flip-flop is not set. As soon as a word is read out from the fifo into the MC176 latches in the middle of page AUX12, the MC135 status flip-flop is set, inhibiting further reads until the IfifoAck' signal arrives from the DES input finite-state machine, signifying that the data has been consumed.

Refer now to page AUX12. Note that, apart from the RIOB data being written into the F145A RAMS, the bottom two bits of TIOA are also being remembered. Thus, when the data emerges from the fifo, the DES input fifo can discriminate between three different types of data (not four, since TIOA=*340* is illegal, as explained above).

If the data is tagged with TIOA[6..7]=01 or 10, then it is for the DES encryption units.

If the data is tagged with TIOA[6..7]=11, then it is for the Counter/Timer.

(TIOA[6..7]=00 should never occur.)

3.2 Dual DES Encryption Units

The heart of the Encryption logic consists of two AMD Am Z8068 MOS LSI DES Encryption chips, shown on page AUX14. Careful reading of the AMD specifications for these chips reveals a number of important points -- read these carefully:

a. The Am Z8068 is capable of being used in two modes, *Direct-Control Mode,* DCM, for Am2900 bit-slice architectural applications, and *Multiplexed-Control Mode,* MCM, for more bus-oriented applications. The Dorado design uses MCM exclusively.

   In MCM, the Master Port is multiplexed between data input and register address input. The processor commands the device to perform various operations by writing commands into internal mode- and command-registers, previously selected by supplying a 2-bit register address on Master Port bits 1 and 2 and pulsing Master Port Address Strobe, MAS'.

   In DCM, the Auxiliary Port is unavailable, and its signals are used instead for controlling internal functions directly by external hardware. Unfortunately, although it might, at first glance, appear that this would be more appropriate for a Dorado design, the full functionality of the chip is not available in DCM. In particular, Cipher-Block-Chaining operations in DCM are complicated, since one can't load the CBC Initialisation Vector in DCM, only in MCM.

b. The Am Z8068 has 3 data ports, Master, Slave, and Auxiliary. The device can operate with 2 different encryption keys, a Master Key and a Session Key. The Auxiliary port, sometimes referred to in the AMD documentation as the Key port, is used for entering the Master Encryption Key, which is intended for use in encrypting session keys for transmission to remote DES equipment. We don't use this feature, so we don't use the Auxiliary port at all.

c. In the AMD literature, the Z8068 is usually shown placed between a computer data bus and a communications link. Thus, the clear text is always on one side of the chip (eg. Master port), and the cipher text on the communication link, on the other (eg. Slave port). On encryption, data is written into the Master port by the CPU (say), and is read from the Slave port by the communication link controller. On decryption, the direction of data flow is reversed, with cipher text being written into the Slave port by the communication link, and being read from the Master port by the CPU.

   Examination of the commands which can be issued in MCM reveals that the ports are not symmetrical in that:

   1. You can't load keys or vectors through the Slave port.

   2. You can't access internal registers through the Slave port.

   In the Dorado design, we have chosen to configure the Master port permanently for data input, and the Slave port for data output. Thus, for encryption, the port configuration bits in the Mode register are set to 01b, (Dual-port, Master Clear, Slave Encrypted) before issuing a Start Encryption command (41x), whereas for decryption, the port configuration bits in the Mode register are set to 00b, (Dual-port, Master Encrypted, Slave Clear) before issuing a Start Decryption command (40x).

   In the Dorado design, no provision is made for reading data out of the Master port. The command codes show that the data which can be read from the Master port consists of the initialisation vectors, IVE and IVD, either in clear or ciphered, or the internal Status register. However, it is never necessary to read the IV's, and the Status register doesn't convey much information which couldn't be deduced by other means. Therefore, not

being able to read the Master port is no great loss.

d. The AMD literature for the Z8068 is not as clear as it might be, and it was necessary to contact the chip's design engineer for clarification of a number of points. Four points are particularly important:

    1. MDS', SDS', and MAS' *must* change synchronously with CLK.

    2. MFLG' and SFLG' will change synchronously with CLK.

    3. After power-up, you *must* issue a hardware reset (MAS' and MDS' active simultaneously).

    4. Any write of a command to the chip *must* be followed by at least 6 clocks delay before performing any other reads of writes on any of the ports. This is the reason for the delay counter/timer, described below.

e. The AMD specification for the Z8068 indicates a maximum clock period of 250nS. After considering a design where synchronisers are provided to permit the DES units to operate on their own 250nS clock independent from the Dorado, it was concluded that the additional complexity was unjustified since the small difference in clock speeds means that many delays will be induced by the phase differences. Thus, the DES chips are operated at Dorado Clock0'/8, namely 256nS. The AMD design engineer expressed confidence that most DES devices would continue to run if the Dorado clock rate was increased, possibly down as far as 240nS. If we ever actually do this, we may also purchase speed-selected versions of the Am Z8068 from AMD.

Now let us consider the overall operation of the DES logic. As mentioned above, data in the DES input fifo is tagged with the two low-order bits of the TIOA it was written with, corresponding to writes to TIOA's of *341-343*. Writes to *340* do not enable the input fifo.

Words tagged with *341* or *342* are routed into the DES chips; words tagged with *343* are loaded into the delay counter/timer.

DES data emerges from the input fifo, is converted by the multiplexor into bytes, and latched again. Thus, whilst the last byte is being written into the DES chips, the DES input finite-state machine can cause the next word to be read out of the DES input fifo, enabling fully-pipelined operation.

The byte data is then converted into TTL and fed into the Master ports of the two AMD Am Z8068 DES chips. The parity is checked (in TTL) just as the data enters these devices, and errors are signalled as bit 14 in the Board Status Register, SR14.

Words tagged with *341* are considered to be *control words*, and consist of an internal DES chip register address in the high-order byte, and either a command code or a device mode in the low-order byte. The address byte is entered into the DES chips using MAS', whilst the command or mode byte is entered using MDS'.

Words tagged with *342* are considered to be *data words*, and consist simply of two bytes of data. Both bytes are entered into the DES chips using MDS', MAS' remaining inactive.

The DES chips operate in units of *blocks*, which are 8 bytes (64 bits) long. The devices are pipelined, allowing a block to be loaded into the chips whilst a second is being encrypted, and a third is being read out of the Slave port. See the pipeline timing diagrams on page AUX03 for more detail.

As the data is read from the Slave ports of the DES chips under the control of the DES output finite-state machine, parity is generated (using 74S280 parity generators) before the outputs of the duplicate encryptors are compared for differences. The IC's used are three 74LS266 open-collector quad exclusive-OR gates. Differences indicate hardware errors, and are noted as bit 13 of the Board Status Register, SR13.

Thus, in the Dorado design, parity is carried through right to the point where the data enters the duplicate DES chips, and is generated again immediately the data emerges from the chips, and before the outputs are compared for error detection.

Finally, the bytes are converted back into ECL, latched for conversion back into 16-bit words, and written into the DES output fifo, all under the control of the DES output finite-state machine.

When at least 4 16-bit words (ie. at least 1 64-bit DES block) have accumulated in the DES output fifo, a Dorado DES task-wakeup is generated and microcode transfers the data back into memory.

3.3 DES Input Finite-State Machine

The DES input finite-state machine, shown on page AUX22, controls all of the data paths from the DES input fifo to the Master ports of the DES encryption chips.

The DES chips are clocked at an eighth of the basic Dorado Clock0' rate, namely 256nS. The DES input finite-state machine is clocked at twice the DES clock rate, namely 128nS, allowing for a grain of two FSM cycles per DES cycle. Thus, it is necessary for DesClock to be an input to the IFSM, so that the IFSM can be made to run in-phase with DesClock. (This is reflected in the software, where, if IFSM gets out-of-phase, either an idle IFSM cycle is inserted (eg. in the IDLE state), or an IFSM error is generated).

Upon GlobalReset, the upper MC149 in the logic diagram is disabled, which effectively forces the IFSM into state 0, the RESET state. The software in the IFSM then tries to transit into state RESET1, but will be unsuccessful until GlobalReset is removed. All outputs from IFSM are active-high, so the software can default them to LOW.

The DES IFSM transits through four states (RESET, RESET1, RESET2, and RESET3) before arriving in the IDLE state. In the IDLE State, the IFSM loops waiting until the two-bit EVENT input (which is derived from the TIOA tags and the DataValid' indication coming from the input fifo), indicate that there is a valid data word waiting in the fifo's output data latch.

If the tag is 01b (ififoHasDesData), the IFSM loads the two bytes of data into the DES chips' Master port, one byte at a time.

If the tag is 10b (ififoHasDesControl), the IFSM loads the upper byte (an internal register address) into the DES chips using MAS', then the lower byte (either a command code or a mode) using MDS'.

Finally, if the tag is 11b (ififoHasTimerData), the IFSM loads the lower byte of the word into the counter-timer, the upper byte being discarded.

If any errors are detected by the IFSM, the output IfsmError is generated and appears as bit 12 in the Board Status Register, SR12.

Two points are worthy of note:

a. The AMD specifications show that the duty-cycle for MDS' is not 50%. Thus, the MC231 is required to extend the period MDS' is active accordingly. All other DES signals behave nicely with 50% duty-cycles.

b. This complicates the DES chip reset procedure, since it is necessary to activate MAS' and MDS' together to perform a reset. The is solved by providing a separate DesReset output from IFSM, which overrides PreMAS and PreMDS by means of the MC105 gates.

3.4 DES Delay Counter-Timer

The 8-bit counter-timer, on page AUX24, is clocked with DesClock. To implement the delays required after loading commands into the DES chips, the delay desired is written into the DES Ififo using TIOA=*343.*

When the IFSM sees a data word with a tag of 11b (ififoHasTimerData), it loads the lower byte of the word into the counter-timer, the upper byte being discarded.

The MC106 gates, which gate the EVENT inputs to the IFSM on page AUX24, ensure that the IFSM will not see further valid data in the IFIFO output latch until the Counter-Timer has counted down to zero (Counter=Zero').

3.5 DES Output Fifo

Refer to page AUX16. The DES output fifo, Ofifo, is very similar to the input fifo discussed above. Again, the memories are dual-ported with a MC158 multiplexor which is switched by FHCP such that write cycles take place ifrom t1 to t2, and read cycles take place from t2 to t3.

Again, the type of the last cycle is remembered by the lower MC135, and this piece of state allows the fifo-full case to be distinguisghed from the fifo-empty case.

Dorado DES task-wakeups are generated when the upper two bits of the read counter and the upper two bits of the write counter differ, indicating that there is at least 4 words of information in the Ofifo. The DesWakeup signal will go away when the microcode reads sufficient data out from the Ofifo to change this.

A word of data will fall out of Ofifo into the right-hand bank of MC176 latches whenever the upper MC135 status register on page AUX15 indicates the latches are empty. Further reads are inhibited until the microcode picks up the data (TIOA=Fifo' AND bIoin').

3.6 DES Output Finite-State Machine

The DES output finite-state machine, OFSM, is quite simple. The OFSM loops in an IDLE state waiting until the Des Slave port flag, DesSFLG, indicates that the slave port has a block of data ready to be read out. It then picks up a byte of data at a time until DesSFLG goes inactive, and loads them into the byte-to-word conversion latches before writing them into the output fifo. Again, the grain of OFSM is 2 states per DesClock, so as to easily generate DesSDS with a 50% duty cycle. No special action is required on a GlobalReset, except that the machine is forced into state 0, the IDLE state.

3.7 64-bit Checksum Unit

The Dorado design provides additional hardware to enable a higher degree of protection against particular forms of cryptographic attack. Consider the equations which describe the cipher-block-chaining algorithm, CBC:

On encryption,

```
C[0] = DES[ P[0] EXOR IVE,    EKey ]          -- 1
C[i] = DES[ P[i] EXOR C[i-1], EKey ]          -- 2
```

On decryption,

```
P[0] = DES[ C[0], DKey ] EXOR IVD             -- 3
P[i] = DES[ C[i], DKey ] EXOR C[i-1]          -- 4
```

Where

```
P[i]   = Plaintext block i,
C[i]   = Ciphertext block i,
EKey   = Encryption key
DKey   = Decryption key
IVE    = CBC initialisation vector for encryption
IVD    = CBC initialisation vector for decryption
```

Notice, from equation 3, that plaintext block i is only a function of ciphertext block i and ciphertext block i-1. Thus, ciphertext blocks can be substituted in the datastream and they may pass unnoticed at the destination.

We solve this problem by checksumming the plaintext at the source, and sending the checksum as the the last ciphertext block. On decryption at the destination, the resulting plaintext is re-checksummed and compared with the decrypted checksum sent from the source, ie:

On encryption,

```
C[i+1] = DES[ Checksum[ P[0..i] ] EXOR C[i], EKey ]
```

On decryption,

```
P[i+1] = DES[ C[i+1], DKey ] EXOR C[i]
       = Checksum[ P[0..i] ]
```

This algorithm, invented by Andrew Birrell, is implemented in hardware on page AUX18. Either the 16-bit plaintext word in the output latch of the input fifo, or the 16-bit plaintext word about to be written into the output fifo is selected by the MC158 multiplexors, depending on whether we are encrypting or decrypting (a bit in the control register). The word is exclusive-or'd with the word from the corresponding 16-bit position in the shift register and written back into the shift register.

When encrypting, after the DES chips have finished encrypting the last block, the microcode sets the ChecksumReset bit in the control register, and reads the 4 checksum words from the checksum unit and writes them back into the Des input fifo. As this is done, the checksum is reset to zero, which it will remain until the ChecksumReset bit is cleared. The checksum is, in turn, encrypted, and is later transferred by the microcode from the DES output fifo back into memory as the last block.

When decrypting, the checksum unit accumulates the checksum of the plaintext words as they are written into the DES output fifo. The last block through the DES chips will be the encrypted checksum sent by the source, and, when exclusive-or'd with the recomputed checksum, should equal zero. When decryption completes, the microcode reads the 4 checksum words from the checksum unit and verifies this.

4. Random Bit Generator

Refer to page AUX26. The random bit generator has four main components, from top to bottom:

a. A white-noise source based on a KN1201 noise diode.
b. An operational amplifier to raise the noise signal level.
c. An ECL comparator, to turn the noise into a bit stream.
d. An isolated power supply, based on a PG505 DC-DC converter.

5. Electronic Stable Storage

To be added.

6. 10 megabits/second **Ethernet Controller**

To be added.