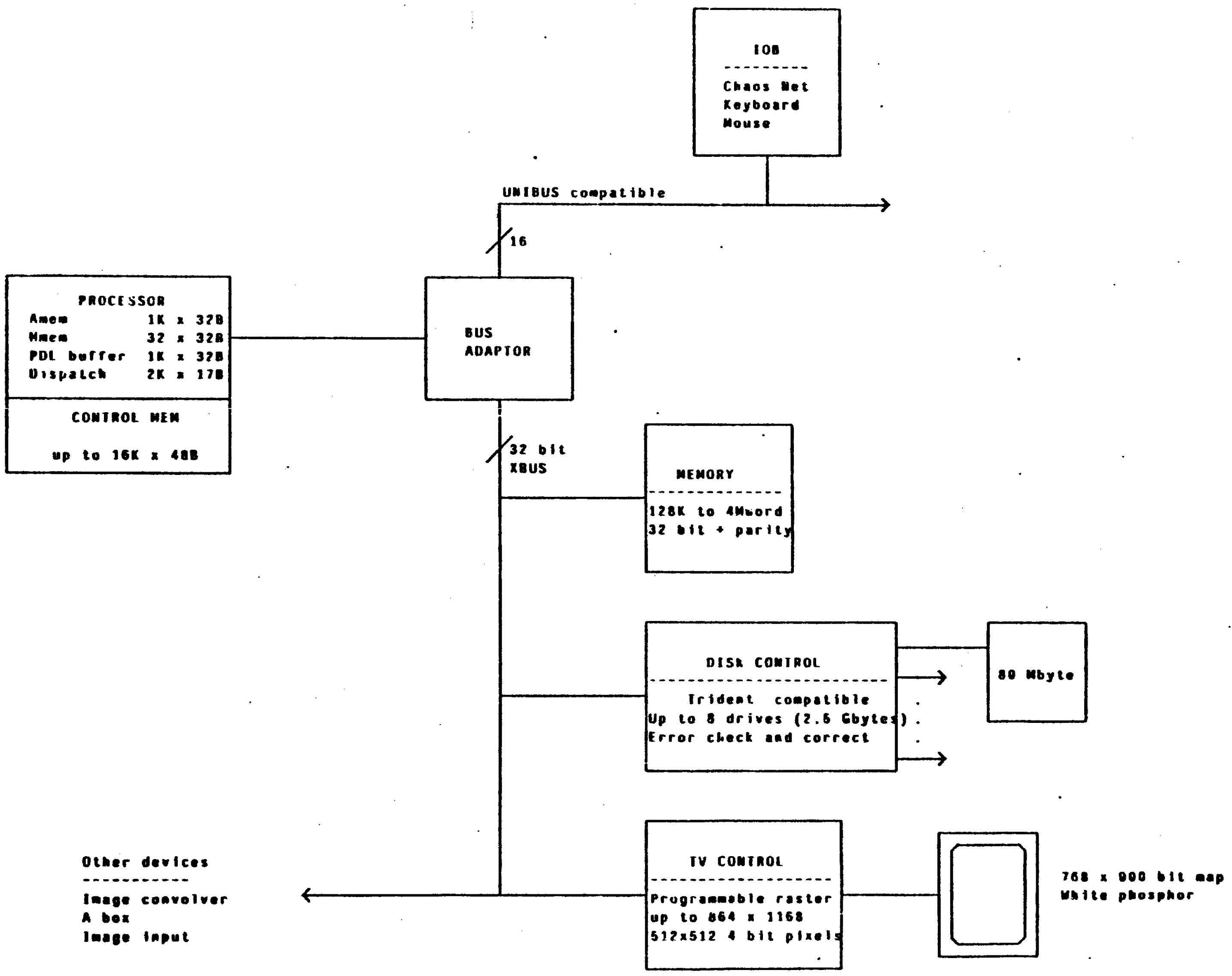
LM-2

Prints



| Lisp Machine | System Configuration | 18-SEP-1981 08:42 | AI: LMDOC; FIG1 |
|---|---|---|---|

The clock can be stopped at the end of either phase, for several reasons. Usually the clock stops at the end of the read phase, referred to as "wait". This leaves the clock in the inactive high state, and leaves the latches on the memories open. The clock can wait because the machine was commanded to halt by the diagnostic interface, because a single-step commanded by the diagnostic interface has completed, because of an error such as a parity error, because of the statistics counter overflowing, or because of a memory-wait condition. This latter condition happens if a main memory cycle is initiated while a previous cycle is still in progress, or if the program calls for the result of a main memory read before the bus controller has granted the bus access needed to perform that read cycle. During a clock wait, the processor clock stops, but the clock to the rest of the system (the bus interface and XBUS devices), continues to run, allowing them to operate. When the processor finishes waiting the processor clock starts up in synchrony with the external clock.

The clock can also stop at the end of the write phase, referred to as "hang". This is used only during memory reads. If the processor calls for the result of a read which is in progress but has not yet completed, it hangs until the data has arrived from memory and sufficient time has passed for the data to flow through the data paths and appear on the output bus. This is also sufficient time for the parity of the data to be checked. In the case of a hang, both clocks stop, which allows them to restart synchronously without any extra delay. In this way, the speed of the processor is adjusted to exactly match the speed of the memory.

ε4The Bus Interfaceε0

The Bus Interface connects the CADR machine to two busses, the Unibus and the Xbus. The Unibus is a regular pdp11 bus, used to attach peripheral devices, especially commercial devices designed for the PDP11 line. The Xbus is a 32-bit bus used to attach memory and high-performance peripheral devices, such as disk. The bus interface also includes the diagnostic interface, which allows a unibus operator, such as a pdp10, a pdp11, or another lisp machine, to control the operation of the machine, hardware to pass interrupts from the Unibus and the Xbus to the processor, the logic which arbitrates the Xbus, and the logic which arbitrates the Unibus in the absence of a pdp11 on that bus.

The Bus Interface allows the CADR machine to access memory on the Xbus and devices on the Unibus, allows independent devices on the Xbus to access the Xbus (only), and allows Unibus devices to access Xbus memory (through a map since the Unibus address space is not big enough.) Buffering is provided when the Unibus accesses the Xbus. to convert a 32-bit word into a pair of 16-bit words.

<<More to come>>

Cover how to program the various frobs from the Unibus, initialization, map structure.

.page
.sect
ε4The Xbusε0

The Xbus is the standard 32 bit wide data bus for the CADR processor. Main memory and high speed peripherals such as the disk control and TV display are interfaced to the Xbus. Control of the Xbus is similar to the Unibus, in that transfers are positively timed and (as far as the devices are concerned) asynchronous. The bus is terminated at both ends with resistive pullups of 390 ohms to ground and 180 ohms to +5 volts, for an effective 123 ohm termination to +3.42 volts. At ground, each termination draws 28 ma. for a total load of 56 ma. The bus is open collector, and may be driven with any device capable of handling the 56 ma. load. The recommended driver is the AMD 26S10, which also provides bus receivers.

A typical read cycle begins with placing the address for the transfer on the -XADDR lines and the parity of the address on the -XBUS.ADDRPAR line. The -XBUS.RQ line is then lowered, initiating the request. The responding device places the requested data on the 32 -XBUS lines and the parity of the data on the -XBUS.PAR line. Should it not be convenient for the device to produce parity (as in the case of I/O registers), the device may assert -XBUS.IGNPAR to notify the bus master that the transfer should not be checked for correct parity. The responding device then asserts -XBUS.ACK, which remains asserted until the -XBUS.RQ signal is removed by the master.

Write requests proceed identically, except that the master asserts -XBUS.WR and the data to be written on the -XBUS lines along with the address lines. All bus masters are required to produce good parity data on writes.

Deskewing delays are the responsibility of the bus master. In particular, it is the responsibility of the bus master to assert good address, write, and data lines 80 ns. prior to asserting -XBUS.RQ, and these lines must be held until the -XBUS.ACK signal drops in response to the master dropping -XBUS.RQ. Responding devices are allowed to assert -XBUS.ACK at the same time they drive read data onto the -XBUS lines. Thus, masters should delay 50 ns. after receiving -XBUS.ACK before dropping -XBUS.RQ and strobing the data. Responding devices are required to drop -XBUS.ACK immediately after -XBUS.RQ is no longer asserted.

Normal bus master arbitration between the CADR processor and the Unibus requests is handled by the bus interface. Devices on the Xbus which must become bus master, such as the disk control, do so by asserting the -XBUS.EXTRQ signal. When the bus becomes free, the bus interface responds by asserting -XBUS.EXTGRANT. This signal is daisy chained between bus master devices on the Xbus, coming in on the -XBUS.EXTGRANT.IN pin and leaving on the -XBUS.EXTGRANT.OUT pin. Within each device, the decision is made whether or not to pass the grant onto the next device. Unlike the Unibus structure, the decision on whether to pass grant and the act of becoming bus master happen synchronously with a master clock signal distributed on the -XBUS.SYNC line.

When a device initiates a request, it immediately asserts -XBUS.EXTRQ. At the falling edge of -XBUS.SYNC it clocks the request signal into a D flip flop which we will call REQ.SYNC. When -XBUS.EXTGRANT.IN goes low, the device asserts -XBUS.EXTGRANT.OUT unless it has either the REQ.SYNC flip flop set, or is already the bus master. At the next falling edge of -XBUS.SYNC the device which has both -XBUS.EXTGRANT.IN and REQ.SYNC set becomes bus master. The device should immediately assert -XBUS.BUSY and may immediately begin asserting address lines for a transfer. -XBUS.BUSY may be dropped asynchronously, after the slave device drops -XBUS.ACK in response to the master's request.

The -XBUS.EXTGRANT.IN signal must be terminated with a resistive pullup of 180 ohms to +5 volts within each device which does not simply connect it to -XBUS.EXTGRANT.OUT.

Signal review:

data lines:

| | |
|---|---|
| -XBUS0 through -XBUS31 | 32 data lines, low when data is a one |
| -XBUS.PAR | parity of the 32 data lines. Required for writes |
| -XBUS.IGNPAR | ignore parity signal, may be asserted by any device for a read |

address lines:

| | |
|---|---|
| -XADDR0 through -XADDR21 | 22 address lines, low for address bit a one |
| -XADDR.PAR | <<this needs to be decided... is this required?>> |

cycle control lines:

| | |
|---|---|
| -XBUS.RQ | Asserted by the master to request a read or write<br>Minimum of 80 ns following stable -XADDR, -XBUS.WRITE and -XBUS data |
| -XBUS.ACK | Asserted by the slave in response to -XBUS.RQ<br>No delay necessary following assertion of good read data |
| -XBUS.WR | Asserted by the master during a write cycle. |

mastership control lines:

| | |
|---|---|
| -XBUS.BUSY | Asserted when a device other than the bus interface is bus master. Only the bus interface examines this line. Asserted on a -XBUS.SYNC clock edge, dropped asynchronously after -XBUS.ACK drops |
| -XBUS.EXTRQ | Asserted when a device other than the bus interface wishes to become bus master.<br>Asserted asynchronously, may be removed asynchronously after the device becomes master, but before dropping -XBUS.BUSY |
| -XBUS.EXTGRANT.IN | The daisy-chained mastership grant signal. Must be pulled up with 180 ohms to VCC in the device. |
| -XBUS.EXTGRANT.OUT | Asserted initially by the bus interface, synchronously with the -XBUS.SYNC edge. The signal may be subject to synchronizer lossage, since it is a clocked version of -XBUS.EXTRQ which is not synchronous with -XBUS.SYNC |

Miscellaneous:

| | |
|---|---|
| -XBUS.INIT | When low, resets all devices. This is low during power on and off, and when the machine is reset. |
| -XBUS.SYNC | Synchronization clock for mastership passing and other desired purposes.<br>Devices become bus master synchronous with the edge of this signal. The request will normally follow the edge by 80 ns. |
| -XBUS.INTR | Driving this low requests an interrupt.<br>All devices are required to initialize to a non-interrupt enable condition, and are required to have interrupt |

enable and disable bits which can selectively enable interrupts from that device. The "requesting interrupt" state must be readable in one of the device control register bits.

XBUS.POWER.OK

This line is HIGH when power is stable. It remains low for <<xx>> seconds after power comes on, and goes low <<xx>> seconds before power is turned off.