The Network Interface Unit (NIU) for the Monsoon Dataflow Processor

Computation Structures Group Memo 290
August 29, 1988

Franz Hutner

This report describes research done at the Laboratory for Computer Science of the Massachusetts Institute of Technology. The author is employed by Siemens AG, West Germany. Funding for the Laboratory is provided in part by the Advanced Research Projects Agency of the Department of Defense under the Office of Naval Research contract N00014-84-K-0099.

Contents

1 Functions

NIU is the connection between the local bus of the Monsoon[1] processor and the network. This network is built of Packet Routing Chips[2] (PaRC), which are interconnected by ECL links and coaxial cable. PaRC is a 4 by 4 routing chip with buffers for max. 16 packets. Each packet consists of 12 16-bit words. The first word contains the routing information together with bits that signal the beginning of a new packet and qualify circuit switched packets. In this case, PaRC and NIU have to send an acknowledgement back to the sending processor when the packet has arrived at its destination. The next 9 16-bit words are data; in the case of the Monsoon processor 72 bits Tag and 72 bits of Value. The last 2 16-bit words are for CRC checking. The input and output port to the network run at 50 MHz; the input port receives the clock together with data from the link. NIU has to deliver a clock signal together with data going out to the network. For this purpose, NIU receives an auxiliary clock signal from the link. The output port to the network runs on this clock and gives it together with the data to the link. On the other side, NIU is connected to the local bus of the Monsoon processor. This bidirectional bus operates at 25-32 MHz. So the local bus interface of the input port and output port run at this clock.

The NIU input port is able to buffer up to 8 incoming packets. It also performs the CRC check, verifies the Processing Element (PE) number and generates handshake signals to the network and local bus.

The NIU output port implements two independent queues for outgoing packets. One is for normal packets, one for circuit switched packets. There is common buffer memory for 15 packets. It can be used for both types. The output port keeps track if a circuit switched packet was sent out. In that case no further circuit switched packet will be sent out until the acknowledgement for this packet is received. If there are circuit switched and normal packets in the queues and NIU is not waiting for a circuit switched acknowledgement, a circuit switched packet will be sent out next. NIU also generates the routing header (by a 8-bit lookup on the most significant bits of the PE number) and the two CRC words which are sent last. Idle patterns will be sent out, if no packets can be sent.

There is also an asynchronous diagnostic port, which allows a service processor to write the status register, load the RAM for lookup and read and reset the counters for statistics. There are counters for each type of error, outgoing packets of both types, cycles while waiting for circuit switched acknowledgement, cycles while input buffer is full, cycles while NIU is blocked to send out packets and clock cycles as a time basis.

2 The Input Port

The input port receives packets like PaRC as 16-bit words together with a clock signal. As data is only valid at the rising edge of the clock, there are edge triggered FFs which store data coming directly from the input pads (module REG16). The output of this register provides stable data words for the checkers (module CHECK) and the memory for incoming packets (module MEM). Bit 15 and 14 are used in the module WORDCNT. Bit 15 of the header is the startbit. It marks the header of a new packet (idlewords have bit 15 = '0') and is used as a start signal for a shift

[1]Papadopoulos, G. M.:*Implementation of a General Purpose Dataflow Multiprocessor. PhD thesis, MIT Department of Electrical Engineering and Computer Science, August 30th, 1988.*

[2]Joerg, C. F.:*Design and Implementation of a Packet Switched Routing Chip for the Dataflow Supercomputer. Masters thesis, MIT Department of Electrical Engineering and Computer Science, December, 1988 (anticipated).*

register in WORDCNT. This shift register of 12 bit provides the write signals for the 9 data words to memory (WORDSEL). Bit 15 of the header is also used to generate a reset signal for the CRC checker (RESCRC) and a signal for the idle check (IDLE). Bit 14 signals that this is a circuit switched packet and that NIU has to generate an acknowledgement which is handed back by the links and PaRC to the sending processor. There is also a block (WPACCNT) within WORDCNT that counts incoming packets modulo 8 and produces the packet select signal for the 8 packet locations in MEM. This counter is incremented by word 11, which is (as word 12) not written to memory. These two words are just used for CRC checking. So a slow asynchronous counter is used.

As incoming 16-bit words arrive at 50 MHz, it was not possible to write them directly into memory (MEM). Instead, these words are alternatingly stored in two parallel, edge triggered registers. Now, there are two input buses (ABUS and BBUS) to each location for a packet (module PACMEM) and every bus is stable for two clock cycles. This gives enough time to use RAM1 cells for memory, which have a low gate count per bit (4). They also have a tristate output. As data are read out as 72-bit words, it was not possible to use one of LSI's hardwired RAMs. This soft-RAM also has the advantage that its shape is variable, which could be important for the layout process, because a large part of the output port consists of similar RAM, too.

There are three checkers parallel to the memory (module CHECK). CRCHK performs the CRC check on incoming data. This module is identical to PaRC's CRC checker. CRC checking is enabled by one input line (USECRC) from the status register. The module IDLECHK is also enabled by a signal from the status register (USEIDLE). There are only two allowed idlewords: '5555' or '2AAA'. All other words received during idle produce an idle error if enabled. PEVERIF compares the PE number contained in the Tag (bits 47-38) with a reference number which can be loaded from the diagnostic port. A mask register in that module allows the checker to disregard some bits (mask bit = '0' means don't care). Like for the other checkers, there is an enable bit in the status register (USEPE). As PE number and mask are 10 bits each, 3 locations are necessary for the 8 bit wide diagnostic port. These locations can be written or read; the write strobes are generated in the module INSTRUMENTATION. All error signals go to INSTRUMENTATION, where they are counted. This module also notifies the processor of the faults.

Whenever a packet is written to memory, the local bus interface must be informed that a new packet has arrived. As this signal (inverted writestrobe N0) goes from one clock domain (received network clock XCLK) to another (local bus clock LCLK), this signal has to be synchronized. This is done by the module SYNC_XI. Its output goes high for just one clock cycle. It has an additional output (WAITING), which is high whenever an event is waiting to be synchronized. This is important if LCLK is much slower than XCLK because there could be more than one event before the first is synchronized. In that case, one event (packet) would be lost. This WAITING signal is used in the module GENWAIT together with a SLOW_MODE bit of the status register to stop incoming packets until the event has been synchronized.

The synchronized signal (UP) is used in GENWAIT to increment a synchronous 4 bit counter. This counter keeps track of the number of packets in memory. It produces both handshake signals for the local bus interface (INRDY) and the network interface (WAIT). INRDY is high if there is at least one packet in memory and goes low as soon as the tag (first half) of the last packet is being read. The timing of the signal WAIT is very critical, as it has to be transmitted over links and cable to the sending PaRC chip. There it has to arrive about 3 clock cycles before the first word of a packet is being sent out to prevent the packet from being sent. This could not be guaranteed by deriving this signal from counter state '8', which means that all buffers are already full. Instead, WAIT goes high when the seventh packet is being written into memory. This may result in the fact that only

7 of 8 available buffer locations are being used (especially in the case of short connections). On the other hand, as WAIT goes high while the seventh packet is received, there is plenty of timing security. So WAIT becomes active if the 4 bit up-down counter in GENWAIT is in the state 7 or 8. As glitches on the WAIT signal must be avoided, this counter changes from '7' to 'F' and back, instead of from '7' to '8'. So it is sufficient, just to derive WAIT from the 3 least significant bits.

When the processor is notified by the INRDY signal that there is one or more packets in the buffer, it will sooner or later start to read from NIU. To do this, NREAD is activated. This enables the output drivers of the bidirectional pads to the local bus, so that the tag of the first packet can be latched in by the processor on the next rising edge of the clock. NREAD is also used to change the state of the module RHPSEL(Read Half Packet SELect). This module produces output enable signals for the memory (NOEHP - Output Enable Half Packet; active low). There are always two output enable lines active; one for each output bus of the memory. They derive from two counters and decoders in RHPSEL, which are alternatingly incremented if NREAD is low. RHPSEL also produces a signal (CDOWN), which is always active when the tag of a packet is being read. This signal is used for two purposes:

- as the decrement signal for the up-down counter in GENWAIT, which keeps track of the number of packets in memory
- as a select signal for one of the two output buses in memory.

As mentioned before, there are always two output enable lines (NOEHP) from RHPSEL active to enable one driver for each bus of MEM. This was necessary, because the tristate outputs of the RAM1 cells were not fast enough to provide sufficient setup time for the local bus directly. So they are always enabled one cycle before their output is used to drive the output pads. This is possible, as packets are always read in the same order in which they arrive (FIFO). So the setup time for data to local bus could be reduced to the time to invert a FF and a driver and switch a 2to1 multiplexer in addition to the time to drive an output pad.

3 The Output Port

This part of NIU is more complex than the input port. The reason is that there are two independent output queues - one for circuit switched packets and one for normal packets. As it is not yet clear what percentage of outgoing packets will be circuit switched packets, it was not optimal to give each queue a fixed number of buffers. Instead, both queues share one memory with a capacity of 15 packets. As it is necessary for both queues to establish FIFO behavior, there must be two real FIFOs for the addresses of the packets. When a packet is sent out, its memory location is free again and must be made available for new packets. It would have been possible to use a FIFO for the addresses of free locations, too. In fact, absolutely the same FIFO as for the two queues could have been used. But there were two objections against that:

- The FIFO with free addresses must be loaded with all 15 available addresses during every HW reset. This is nearly impossible if (gatesaving) RAMs are used for the FIFOs.
- A FIFO contains more information than needed, because it is not necessary that free memory blocks are used in the same order in which they get free.

Therefore a solution with just one FREEBIT per memory location and a fixed priority for occupying blocks was more adequate, as less gates are needed to implement that.

Whenever the processor wants to send out a packet, it first checks the signal OUTRDY. This signal is high as long as there is at least one free location in memory and is generated by the module FREE_BL. FREE_BL also gives out the address (4 bits) of a free location (WBLOCKADR) and the decoded select signal (WBLOCKSEL, 15 bits) for the memory. The next step for the processor is to activate LWR and - if this packet should be a circuit switched packet - CSW. They are used in the module LBCTL (Local Bus ConTroL) to produce the following signals: OCC (OCCupy a location) goes to the module FREE_BL. It causes the FREEBIT assigned to the current address to change to '0' and the address and select signals to stay stable for two clock cycles. Either IN_NO or IN_CS are active for the first clock cycle after LWR, depending on CSW. These signals store the current address either to the circuit switched or to the normal address FIFO. LBCTL also generates two write strobes, WFH and WLH which are active for the high part of the clock during the first (WFH) or the second (WLH) clock cycle after LWR became active. There is also a set of edge triggered FFs to store data from the local bus. Strong drivers generate the data input to all memory locations in parallel. Bits 47:40 also go to D inputs of an address latch for the 256X8 lookup RAM and are stored there with the first write strobe (WFH). The output of the RAM is stored to the current memory location with the second write strobe (WLH). So all the information needed to construct the header for this packet is ready. There is a second address latch, which can be written from the diagnostic port. A line from the status register (LOADRAM) selects one of the two address latches either for normal operation (low) or to load or read the RAM (LOADRAM high).

Both queues for circuit switched and normal packets are implemented as identical modules (ADR_FIFO). The structure of this module is similar to the input port. An address is written to the FIFO by asserting IN for one clock cycle. IN is used as a write strobe for the 16X4 RAM. One clock cycle delayed, it is used as an enable signal for a 4 bit counter (CNT4S), which creates the write address. In order to inform the output side of the FIFO that there is an address in the next memory location, IN has to be synchronized to the output clock domain (OCLK). During normal operation, IN is directly used. But if the processor works at a much lower clock frequency than the network, it might happen that the second half (Value) of a packet is not yet written to memory while the output port already reads that location. Therefore SLOW_MODE, a line from the status register, selects the one cycle delayed version of IN to be synchronized. On the output side of the FIFO, there are two synchronous 4 bit counters. One of them is incremented by the synchronized IN signal. This counter keeps track of the incoming packets. The other counter is incremented each time a packet of this queue is sent out. This counter provides the read address for the FIFO RAM, which contains the address of the next packet of this queue. There is a comparator, too, which produces an EMPTY (and inverted NEMP) signal for this FIFO. As there are only 15 memory locations for packets and each queue is able to store all available 15 addresses, no handshake signal to input side of the FIFO is necessary.

The EMPTY signals of the two queues are used in the module OUTCTL (OUTput ConTroL). This module implements the following scheduling strategy: circuit switched packets are sent out whenever possible. That means, when the circuit switched queue is not empty, the receiving PaRC chip is not asserting WAIT and the acknowledgement for the last sent circuit switched packet has been received (WFACK low). Otherwise, if the normal queue is not empty and WAIT is not active, a normal packet is being sent out. If no packets can be sent out, idle patterns are sent (alternating '2AAA' and '5555').

As the arriving WAITIN signal is not synchronous to OCLK, it has to be synchronized. There is a status bit NO_PAC_OUT which is treated like WAITIN. This way, NIU can be stopped from sending out packets. The state of OUTCTL is changed in two steps. First, either the normal or the circuit switched queue is chosen according to the scheduling strategy. A FF for one of the cases is set. Dependent on the selection, ENB goes high. This selects the address from the circuit switched queue to be decoded and applied to memory and FREE_BL. In that case, the signal WFACK in OUTCTL is set to indicate that NIU is now waiting for the acknowledgement for this circuit switched packet. When it is received at its destination, a circuit switched acknowledge signal is passed back by all PaRC and link chips. As this signal (CSWACKIN) arrives asynchronously, it is synchronized to OCLK. Then it is used to reset WFACK.

In the second step, after either an address from the circuit switched or normal queue has been selected, START is set, unless the internal wait signal (WAITINT) goes high on the same cycle or NWAIT_SINT is active. This signal is derived from WAIT_SYNC which is active if SLOW_MODE is asserted and an event is waiting to be synchronized from OCLK to LCLK in FREE_BL. This is done to keep the address stable until the corresponding FREEBIT in FREE_BL is set by the synchronized signal.

START goes to two modules: RWSEL (Read Word SELect) and O_MEM. RWSEL consists principally of two alternatingly enabled shift registers. They provide the output enable signals to memory (RWOE - Read Word Output Enable) for the 10 first words of an outgoing packet. There are always 2 enable signals active, one for each of the two output buses of the memory. Bit 4 of RWOE also goes to FREE_BL. There it is synchronized and used to set the corresponding FREEBIT. RWSEL also produces WORD9, a signal that is active for just one clock cycle when the last word is read out of memory. It is used to reset OUTCTL. There is also the signal WORD1011, which selects the CRC words for output. Finally, a signal WORD2468 is generated that determines, which of the two output buses is to be selected. In O_MEM, START is used to switch to idlewords if no packet can be sent and to reset the CRC generator after each packet.

O_MEM contains memory for 15 packets, each 144 bits of data and 14 bits for the header (lookup). It was not possible to connect all tristate outputs to just two 16 bit buses, as the load would have been 15 x 5 plus wireload. Even with an enable line one cycle ahead this was not fast enough. So there are 5 groups of 3 locations each. Their outputs are internally connected via two tristate buses. There were two ways to connect the 5 x 2 buses:

- tristate drivers for each group and only one multiplexer to chose one of the two buses.
- two 5-to-1 multiplexers and one multiplexer for the output bus.

The first version had the advantage that the total of wiring is rather low. But there is the problem of high power consumption and noise on the bus if drivers overlap for a short time on the bus. And as there is already a decoded select signal (RBLOCKSEL) available, it seemed apropriate to use AND-OR gates instead of a multiplexer or tristate driver. There is one more multiplexer to select either the first (ABUS) or the second (BBUS) or an idleword to go to the output register. The scan inputs of the output register are used to feed the CRC words back to the output. The enable line SELCRC is connected to WORD1011 of the module RWSEL. If START from OUTCTL is low, the idleword is selected. One clock cycle delayed, it is also used to reset the module that generates the CRC words (CRC32F). This module is directly taken from PaRC.

4 Instrumentation

This module contains the status register, counters for errors and statistics and most of the logic for the diagnostic port.

The **status register** contains the following bits:

USECRC: enables CRC checking

USEIDLE: enables checking of idlewords

USEPE: enables PE number verification

SLOW_MODE: must be set if $LCLK < 22MHz$

CNT_ENA: enables counting of statistics; this signal is synchronized to the apropriate clock domain before being used as an enable signal

NO_PAC_IN: sets WAIT and therefore stops incoming packets

NO_PAC_OUT: is treated like WAITIN; no packets are sent out

NO_CSACK: inhibits sending an acknowledgement for a circuit switched packet (this can be done by PaRC)

CHKACK: if set, an arriving circuit switched acknowledgement is treated as error if NIU is not waiting for it

LOADRAM: address for RAM 256X8 is taken from latch which is loadable from diagnostic port

TESTMODE: selects the lower 8 bits of LBDATA as address for the lookup RAM256X8. This bit also changes the input of every 8(4) bit portion of the event counters to TEST_INC, which can be enabled by a write on the diagnostic port.

The status register can be written and read via the diagnostic port.

The following 8 kinds of **errors** are possible:

EXTERR: external error from a link chip that should be passed to the processor
PERR: PE number wrong
IDLERR: wrong idleword
CRCERR: CRC error
NW_SPACERR: packet written to full input buffer
RD_SPACERR: processor read packet from empty input buffer
WR_SPACERR: processor wrote packet to full output buffer
CSACKERR: circuit switch acknowledgement received, although not waiting for it

There is a 2 bit Gray Counter for every type of error in the module ERRCNT. The counters stick to '10' when more errors arrive (transitions: 00-01-11-10-10-10...). These 16 bits can be read out on two locations of the diagnostic port and the counters can be reset by a write to one location. They are also reset by the general reset NRES. Read location '06' consists of (bits 7:0): IDLERR(1:0), PERR(1:0), CSACKERR(1:0), NW_SPACERR(1:0). WR_SPACERR(1:0), RD_SPACERR(1:0), EXTERR(1:0), CRCERR(1:0) can be read as bits (7:0) of location '07'. To reset all error counters any value can be written to '09'.

INSTRUMENTATION also contains 6 **counters for statistics**:

COUNT_NO: number of outgoing normal packets (24 bits)
COUNT_CS: number of outgoing circuit switched packets (24 bits)
OCYC_BLOCKED_SEND: NIU could send packets, but WAITIN blocks (28 bits)
LCYC_INBUF_FULL: NIU can't receive packets from net (28 bits)
OCYC_WFACK: NIU waits for ack of a circuit switched packet (28 bits)
OCLK: clock cycles as a time basis (28 bits)

These counters are asynchronous and there can be no overflow of the counters for at least 4.02 sec. In order to read the contents, the CNT_ENA bit in the status register must first be set to '0'. There are 20 locations which can be read via the diagnostic port. To test these counters, TESTMODE must be set and the counters should be reset. Now, every 8(4) bit portion of these counters can be incremented by the falling edge of TEST_INC. A write to address '0A' of the diagnostic port activates TEST_INC. So it is possible to set all bits of these counters with just 255 write operations. And resetting TESTMODE won't destroy the contents of the counters, so that subsequent normal operation can generate a carry signal changing the state of the whole counter.

The following locations are reachable by the **diagnostic port**:

READ

'00' status register low
'02' status register high(11:8) : status register high(11:8)
'04' RAM output
'06' error cnt low (NW_SPACERR, CSACKERR, PERR, IDLERR)
'07' error cnt high (CRCERR, EXTERR, RD_SPACERR, WR_SPACERR)
'08' COUNT_NO(7:0)
'0A' COUNT_NO(15:8)
'0C' COUNT_NO(23:16)
'0E' COUNT_CS(7:0)
'10' COUNT_CS(15:8)
'12' COUNT_CS(23:16)
'14' OCYC_BLOCKED_SEND(7:0)
'16' OCYC_BLOCKED_SEND(15:8)
'18' OCYC_BLOCKED_SEND(23:16)
'1A' LCYC_INBUF_FULL(7:0)
'1C' LCYC_INBUF_FULL(15:8)

'1E' LCYC_INBUF_FULL(23:16)
'20' LCYC_INBUF_FULL(27:24):OCYC_BLOCKED_SEND(27:24)
'22' OCYC_WFACK(7:0)
'24' OCYC_WFACK(15:8)
'26' OCYC_WFACK(23:16)
'28' OCLCK(7:0)
'2A' OCLCK(15:8)
'2C' OCLCK(23:16)
'2E' OCLK(27:24) : OCYC_WFACK(27:24)
'30' PE reference reg low
'31' mask reg low
'32' PE reg high(9:8=3:2) : mask reg high(9:8=1:0)

WRITE

'00' status register low
'01' status register high(3:0)
'02' address latch for RAM
'03' RAM
'04' PE reference reg low
'05' mask reg low
'06' PE reg high(9:8=3:2) : mask reg high(9:8=1:0)
'07' global reset for NIU (NRES)
'08' reset all event counters
'09' reset error cnt
'0A' TEST_INC increments all 8(4) bit portions of the event counters

5 Timing Specification

All delay times given here are worst case commercial (wccom) and a load of 50 pF is assumed for all signals on the local bus. It is important to realize that these specifications derive from delay estimates based only on the number of receivers. No physical layout or floorplan is regarded. Therefore the whole timing has to be revised after the floorplan has been developed.

5.1 Network Interface

All timing behaviour of NIU to network-side is identical to PaRC. Only WAIT, the outgoing handshake signal for incoming packets, differs. It gets active while word 9 of the seventh packet is being received. Therefore, the whole transmission time of packet 8 is available to stop PaRC sending a ninth packet.

5.2 Local Bus Interface

OUTRDY_P stable 25.5 nsec after rising edge of ext. clock

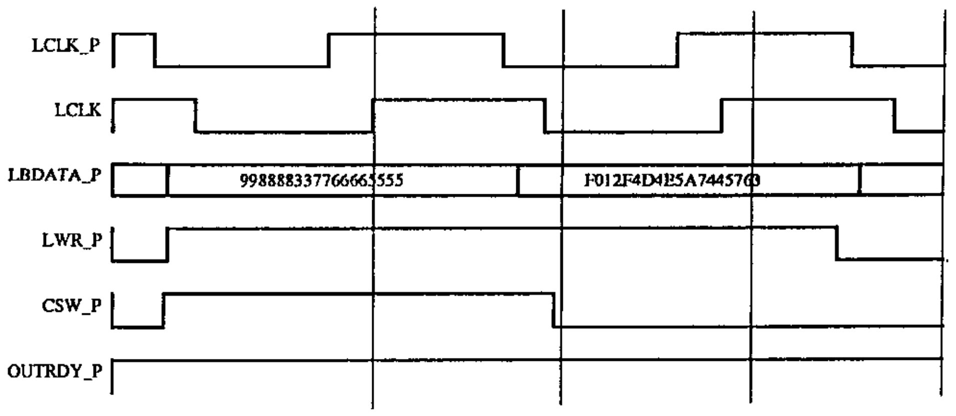


Figure 1: Write to NIU on local bus

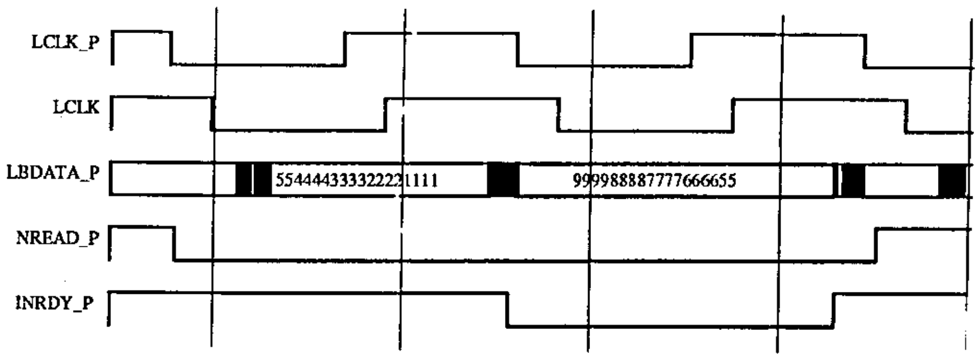


Figure 2: Read from NIU on local bus

INRDY_P	stable 20.6 nsec after rising edge of ext. clock
LWR_P	requires 11.3 nsec setup time before rising edge of ext. clock (12.8 nsec before int. clock)
CSW_P	requires 4.4 nsec setup time before rising edge of ext. clock (5.9 nsec before int. clock)
NREAD_P	requires 13.5 nsec setup time before rising edge of ext. clock (15 nsec before int. clock)
LBDATA_P	output enabled 11.1 nsec after NREAD; new data stable 20.3 nsec after rising ext. clock
LBDATA_P	requires 1.5 nsec setup time before rising edge of ext. clock (3 nsec before int. clock)
LCLK_P	10.8 nsec < pulsewidth < 21.9 nsec at 32 MHz.
	Worst case/typical process parameters (factor 1.5) were considered.

5.3 Diagnostic Port

WRITE
DIAGADR Tsetup 10 nsec; Thold 0 nsec
DIAGDAT Tsetup 0 nsec; Thold 10 nsec
DIAGWN Twidth 28 nsec
READ
DIAGADR Tsetup 0 nsec; Thold 10 nsec

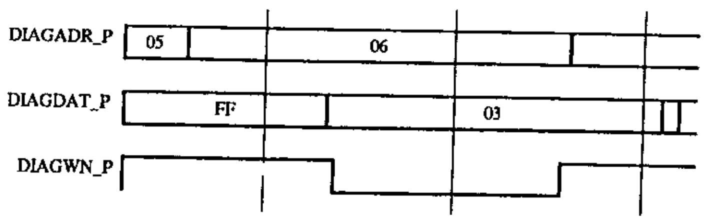


Figure 3: Write on diagnostic port

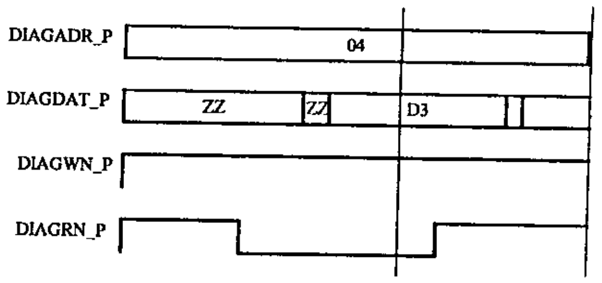


Figure 4: Read from diagnostic port

DIAGRN to DIAGDAT_P: 26.4 nsec

6 Functional Tests

6.1 Submodules

In order to shorten the the time to debug the whole design, all critical submodules were simulated before they were used in the chip. This also results in a short cycle time for changes and resimulation, as the time to link a circuit depends very much on the complexity. The following modules were functionally tested.

WORDCNT

Initialization: NRES active; ENA and CSW low (2 clock cycles)

Idle: NRES high (inactive); ENA low; CSW don't care ⇒ WORDSEL remains '000', IDLE and RESCRC stay high and CSWACK is low. WPACSEL stays at '01'

Normal packet: ENA high; CSW low ⇒ IDLE and RESCRC change to low (same cycle); WORDSEL takes the values 001-002-...-800-000; CSWACK remains low; RESCRC goes back to high with '400' (word 10) and IDLE with '800' (word 11); WPACSEL changes to '02' with '800'.

CS_packet: ENA and CSW high ⇒ IDLE and RESCRC change to low; CSWACK is high for one clock cycle (together with '001'); WORDSEL takes the values 001-002-...-800-000; WPACSEL changes to '04'

Inhibit acknowledgement: NO_CSACK changed to high; circuit switched packet ⇒ CSWACK remains low; WPACSEL changes to '08'

Directly following packets: ENA is already high when the first packet finishes ('800') ⇒ WORDSEL changes from '800' to '001'; IDLE doesn't go high; RESCRC is active for one cycle ('800').

Sync_signal N0 goes low with the first word of each packet for one cycle

The behaviour of all signals is examined by this simulation with the exception of WPACSEL. This will require more packets (at least 8) and will therefore be handled in the final simulation for the whole chip.

GENWAIT

Initialization: NRES active; UP and DOWN low; WAITING, SLOW_MODE and NO_PAC_IN low (2 cycles).

Idle: NRES inactive; UP and DOWN low; WAITING, SLOW_MODE and NO_PAC_IN low ⇒ no changes.

Packet arrived: UP high for one cycle ⇒ INRDY goes high (counter '1')

More packets: UP high for 3 cycles ⇒ counter '4'

Packet while read: UP and DOWN active for one cycle ⇒ counter '4'

Read: DOWN active for two cycles ⇒ counter '2'; INRDY still active

Read and arriving packets: UP and DOWN active for 2 cycles ⇒ counter '2'

More packets: UP high for 6 cycles ⇒ counter '8'; WAIT got active after 5 cycles (counter '7') and is still high; LCYC_INBUF_FULL is high while LCLK is low and WAIT is high;

Idle: UP and DOWN low; no changes besides LCYC_INBUF_FULL.

Read: DOWN active for one cycle ⇒ counter '7'; WAIT still high

Packet arrived: UP high for one cycle ⇒ counter '8'; WAIT still high

Read: DOWN active for two cycles ⇒ counter '6'; WAIT low; LCYC_INBUF_FULL remains low.

Too_many pack arrived: UP high for 3 cycles ⇒ NW_SPACERR gets high and nonconsistent status (must be reset).

Reset: NRES active for two cycles ⇒ counter '0'

Read from empty buffer: DOWN active for one cycle ⇒ RD_SPACERR is high for one cycle and nonconsistent status (must be reset).

Asynchronous components of WAIT overlapped with the former simulation is the test that WAIT must be additionally high if WAITING and SLOW_MODE or NO_PAC_IN is high.

This simulation covers all combinations for idle, UP, DOWN and both as well as the two possible errorcases. It also proves that WAIT is high (and remains high without spikes) at states '7' and '8' and transitions between them. It also checks the additional components of WAIT and the statistics signal.

RHPSEL

Initialization: NRES and READ are low for two clock cycles ⇒ CDOWN is low and NOEHP 'FFFC' (bits 0 and 1 active).

Idle: NRES inactive; READ low ⇒ no changes

Read: READ active for two clock cycles (Tag and Value) ⇒ CDOWN goes immediately high and down after the first rising edge of the clock. There is a spike after the second clock, but CDOWN is only used synchronously. NOEHP changes from 'FFFC' to 'FFF9' (bits 1 and 2 active) and to 'FFF3' (bits 2 and 3 active)

Idle: READ low ⇒ no changes

Read_4: READ active for 8 cycles (4 packets) ⇒ CDOWN high immediately and then on every second cycle (and spike at the end). NOEHP changes 'FFF3'-...-'F3FF' (always two bits enabled)

Idle: READ low ⇒ no changes

Read_3: READ active for 6 cycles (3 packets) ⇒ NOEHP changes 'F3FF'-...-'FFFC' (initial status; 8 packets read)

Idle and read: like after initialization

IDLECHK

Check_enabled ENA high; '5555','2AAA' ⇒ ok; 'AAAA', '5554', '2A8A', '5D55' ⇒ IDLERR high;

Check_disabled ENA low; 'FFFF' and '0000' ⇒ IDLERR remains low.

Check_enabled '2AAA' and '5555' ⇒ ok; 'AAAA' ⇒ IDLERR high.

PEVERIF

Initialization: set reference '3FF' and mask '3FF' (no don't care bits) DIAGADR is set to '0', '1' and '2' to verify the written reference and mask

Right PEnumber: DIN '3FF' and SELPE high ⇒ PERR low.

Disabled: DIN '000' and SELPE low ⇒ PERR low.

Enabled and error: DIN '2FF' and SELPE high ⇒ PERR high.

Error masked: set mask to '300' (lower 8 bits don't care); '3F0' ⇒ PERR low; '30F' ⇒ PERR low; '0FF' ⇒ PERR high.

This functional test contains all principially different cases: PE equal reference, PE not equal reference, PERR not enabled, error masked out, load and verify the registers. There is not a complete test of all possible bitwise error combinations. But as the checker is drawn with buses, it is not possible that some bits are right connected and some wrong.

LBCTL

Initialization: NRES active; LWR, CSW and CSWACKIN low for two cycles ⇒ all output signals low.

Idle: LWR, CSW and CSWACKIN low ⇒ all output signals low.

Write: LWR high for two cycles (Tag and Value) ⇒ OCC goes immediately high; IN_NO goes high for the next cycle and WFH is high for the first positive pulse of LCLK. WLH is high for the next positive pulse of LCLK. There is also a spike on OCC at the end of LWR, but OCC is only used synchronously.

CSWACK: CSWACKIN goes high for a short time; synchronized ⇒ CSWACK goes high for one cycle.

Write cspacket: LWR is high for two cycles; CSW only for the first ⇒ IN_CS is high for the next cycle and WFH and WLH like before.

Idle: LWR, CSW and CSWACKIN low ⇒ all output signals low.

Consecutive writes: LWR is active for several cycles ⇒ OCC is going high immediately and then on every second cycle; IN_NO or IN_CS go high depending on CSW; WFH and WLH are high alternatingly for the positive pulses of LCLK.

This simulation contains all transitions of the circuit. However, there are certain timing constraints especially for LWR and CSWACKIN, which are not covered by this functional test.

ADR_FIFO

Initialization: NRES active; SLOW_MODE, IN, OUT low ⇒ EMPTY goes high

Adr_in: ADRIN set to '0' and IN high for one cycle (ADRIN is stable for the next cycle too) ⇒ ADROUT gets the value '0' and EMPTY goes low.

Filling Buffer: 15 more addresses are written into ADR_FIFO. Each address is applied to ADRIN for two cycles; IN is active for each first cycle.

Read: while filling buffer, the first address '0' is taken out of ADR_FIFO. Then ADROUT takes the value of the next address in the queue 'F'.

Read until empty: There are still 15 addresses in the queue. They are read out on following cycles by setting OUT high for 15 cycles. The addresses appear after each rising edge of OCLK in the order in which they were written. EMPTY goes high after the last address ('8') has been taken out of the fifo.

SLOW_MODE is set. Then ADR_FIFO is reset and one address '8' written into the fifo. EMPTY goes low like before, but one LCLK cycle later.

The final simulation of the whole chip contains more than only one cycle of filling and reading out. But the boundary conditions like queue full (15 packets), queue empty and SLOW_MODE are covered by this.

FREE_BL

Initialization: NRES active for 2 clock cycles. OCC, SLOW_MODE and REL are low; RBLOCKADR is undefined. ⇒ WBLOCKADR and WBLOCKSEL are also undefined. OUTRDY, WR_SPACERR and WAIT_SYNC are low.

Occupy one address: OCC is active for one clock cycle; on the next rising edge WBLOCKSEL takes the value '0001' and WBLOCKADR '0'.

Idle: OCC and REL remain low ⇒ all outputs unchanged

Consecutive occupying: 15 more locations are occupied on every second clock cycle

Release while occupying: one address RBLOCKSEL '0001' is released and re-occupied on the next cycle

OUTRDY: changes to low after 12 locations were occupied, one released and 4 more occupied.

Release location '4000' ⇒ OUTRDY high

Occupy block ('4000'); OUTRDY low

Release location '0800' ⇒ OUTRDY high

Release location '0001'

Occupy two loacations ⇒ OUTRDY low

Occupy next loacation ⇒ WR_SPACERR goes high for one cycle (no location free)

SLOW_MODE ⇒ WAIT_SYNC is high whenever a signal is waiting to be synchronized

OUTCTL

Initialization: NRES active for 3 clock cycles.

Normal packet: NO_EMP low; CSP_EMP high; WAITIN, WAITSYNC and NO_PAC_OUT low ⇒ ENB remains low, START goes high after the second cycle, WFACK remains low and COUNT_NO changes to high

Termination WORD9 releases ENB and on the next cycle START; COUNT_NO changes to low and NORM_OUT is high for one cycle.

Suppressed CSACKERR: CHKACK low and CSWACKIN while WFACK low

CS_packet: CSP_EMP low (also NO_EMP) ⇒ ENB is set and START on the next cycle; WFACK and COUNT_CS go also high

CSWACKIN releases WFACK together with WORD9; no acknowledgement error

WORD9 terminates ENB, START and COUNT_CS and activates CSP_OUT.

Waiting normal packet is selected immediately after former selection is released. START goes high after one cycle

CSWACKIN results in a CSACKERR, because WFACK is low.

WAITIN is synchronized (2 cycles) and inhibits the selection of a waiting normal packet

NO_PAC_OUT has the same effect as WAITIN.

WAIT_SYNC also inhibits the selection of a new packet, but delays NORM_OUT until the falling edge of WAIT_SYNC; START terminates as usual with WORD9

WAITIN even suppresses START if applied two cycles ahead; the selection cs/normal packet is fixed then. That means, if a cs packet gets available while waiting and a normal packet was selected, the normal packet will be sent out.

RWSEL

Initialization: START is low for 3 cycles ⇒ WORD2468 is high, WORD9, WORD1011 are low and RWOE is '003'.

Idle: as long as START is low, no changes in the outputs.

START goes high ⇒ WORD2468 toggles with every clock, RWOE takes the values 003-006-00C-...-300-201-003. WORD9 is high with '201'; WORD1011 for the next two cycles. START is supposed to change to low for the second half of WORD1011.

Idle: START remains low for one cycle; no changes.

Consecutive START: there is always one low cycle between two packets. This does not mean that there are idle patterns sent out!

ERRCNT

Initialization by NRES and then by ERRES ⇒ ERRCNT(ADR0 low) = '00' (single counters 00); ERRCNT(ADR0 high) = '00'

Events on the lower half of ERRCNT ⇒ ERRCNT(ADR0 low) = '55' (single counters 01); ERRCNT(ADR0 high) = '00'

Events on the upper half of ERRCNT ⇒ ERRCNT(ADR0 low) = '55'; ERRCNT(ADR0 high) ='55'

Events on the lower half of ERRCNT ⇒ ERRCNT(ADR0 low) = 'FF' (single counters 11); ERRCNT(ADR0 high) = '55'

Events on the upper half of ERRCNT ⇒ ERRCNT(ADR0 low) = 'FF'; ERRCNT(ADR0 high) ='FF'

Events on the lower half of ERRCNT ⇒ ERRCNT(ADR0 low) = 'AA' (single counters 10); ERRCNT(ADR0 high) = 'FF'

Events on the upper half of ERRCNT ⇒ ERRCNT(ADR0 low) = 'AA'; ERRCNT(ADR0 high) ='AA'

Events on the lower half of ERRCNT ⇒ ERRCNT(ADR0 low) = 'AA' (single counters 10); ERRCNT(ADR0 high) = 'AA'; counters stick

Events on the upper half of ERRCNT ⇒ ERRCNT(ADR0 low) = 'AA'; ERRCNT(ADR0 high) = 'AA'

CNT8A

Initialization: NRESCNT is low; EAN is high and CNT is low. TESTMODE is not selected. ⇒ DOUT is '00' and NCY is low.

Counting: four events happen, then ENA changes to low and DOUT doesn't change anymore. Then ENA goes high again and counting starts.

Disabled again and the counters are reset.

Normal counting again; NCY goes high.

TESTMODE is set after CNT stops. Now TEST_INC increments the counter 5 times and TESTMODE is reset afterwards.

Counting starts again. Finally, DOUT changes to '00' and NCY falls. This would increment the following counter (type CNT8B).

Some more pulses are counted.

6.2 Functional Test of the Complete Chip

Extensive simulation for the complete circuit is necessary, because a lot of the possible problems derive from the interaction of the modules. There are also a number of less critical modules, which were not simulated independently. But their correct behaviour must also be proved. The period of LCLK was chosen to be 30 nsec (33 MHz), XCLK and OCLK 20 nsec (50 MHz). There may be new problems when LCLK is only 22 MHz (cycle 46 nsec; no SLOW_MODE) or when LCLK is below that (SLOW_MODE set). These cases are treated in independent simulations.

Initialization: prepare NIU for normal operation: all checkers enabled, counters enabled and reset, load (here only one location '33') lookup RAM; set reference for PE verification and mask (no don't care bits). HW reset for at least 3 clock cycles.

Verification: the values written before are read back.

In the following, the functional tests of the input buffer and the output buffer as well as the instrumentation are described independently. Nevertheless, in the simulation of the complete NIU chip they are overlapped.

6.2.1 Stimuli for the Output Port

Idle: LWR_P low; no packets in the output queue ⇒ no changes in state; idle patterns (alternatingly '5555' and '2AAA') sent out to network and OUTRDY_P is high.

Normal packet from processor: first Tag and then Value applied to the bus and LWR_P high for two cycles; CSW_P low. OUTRDY_P remains high. WBLOCKADR '0' is occupied and Tag and Value stored in there. IN_NO stores the address '0' to the ADR_FIFO for normal packets. NORM_EMP changes to low. RBLOCKSEL takes the value '0001' and START changes to high. Then RWOE takes the values 003-006-...201-003. So the two buses ABUS and BBUS in O_MEM are driven alternatingly and take the (inverted) values of the 16 bit words contained in the packet. The header is first taken from ABUS. WORD2468 selects the bus; in case of high ABUS. After word 9 SELCRC goes high for two cycles of OCLK. This selects the CRC words to be sent out. After word 9, NORM_OUT switches the ADR_FIFO to the next address. REL goes high and causes the current address '0' to be released in FREE_BL.

4_normal packets from processor directly following each other. WBLOCKADR '0' is re-used for the first packet; '1', '2', '3' for the other three. The packets are sent out in the order of their arrival and with no idlewords between them. The addresses '0', '1', '2', '3' are released in this order. NORM_EMP goes back to high after address '3' has been taken out of the FIFO and RBLOCKSEL is undefined again.

Disable counting of statistics: '07' is written into location '00' of the diagnostic port (status register low) ⇒ ENA_CNT = 0; This is an important test, because it must be possible to disable counting during normal operation without changing the contents of the counters. As the diagnostic port is asynchronous, ENA_CNT is synchronized to both clock domains in which events are counted. The transitions must not be at the same time als the rising edges of the counted events.

CS_packet from processor stored in location '0'; CSP_EMP goes low; ENB changes to high and the packet is sent out to the network. NIU is now waiting for the acknowledgement for this cs packet.

Normal packet from processor stored in '1'; NORM_EMP goes low. After the cs packet has been sent out, the transmission of this normal packet starts immediately with no idle pattern between them. Then location '1' is released again and '1' is taken out of the queue for normal packets. RBLOCKSEL is undefined again and idle patterns are sent out now.

Enable counting of statistics: '17' is written into location '00' of the diagnostic port (status register low) ⇒ ENA_CNT = 1; Disabling and enabling of the counters is done several times during simulation.

Sequence of packets from processor: 2 normal packets (location 0 and 1), 1 cs packet (2), 10 normal packets (locations 3, 4, 0(free again), 5, 6, 7, 1, 8, 9, A), 1 cs packet (3), 4 normal packets (stored in B, C, D and 4) and 1 cs packet (location E). Now, all locations are occupied and OUTRDY_P changes to low after the Tag of the last packet was written to NIU.

Transmission: The packets are sent out in the following order (locations): 0 - 1 - 3 ('2' is a cs packet and WFACK is still high) - 4 (CSWACKIN_P is high for a short time; resets WFACK) - 0 (was selected before WFACK was reset; after '0' is released, there is a free location again and OUTRDY_P goes back to high) - 2 (cs packet; CSWACKIN_P received immediately) - 3 (another cs packet) - 5 (CSWACKIN_P received again) - E (last cs packet; CSP_EMP changes to high) - 6 - 7 - 1 (WAITIN_P changes to high 2 cycles before START would go high for '8'. START is suppressed, but the selection of '8' for RBLOCKSEL ('0100') remains).

WAITIN_P changes to low and the earlier initiated transmission of the packet stored in '8' starts. Next, the packets from '9', 'A' and 'B' are sent out to the network. While 'B' is sent,

WAITIN_P is set high again. Transmission of 'B' is completed, then the next packet from the normal queue (location 'C') is selected, but START remains low. When WAITIN_P returns to low, START comes up and the before selected packet is sent out. Then 'D' and '4' are sent out.

Processor writes two packets in a row; the second is a cs packet. Both are sent out to the network in the same order; WFACK is set again and both addresses '0' and '1' are released.

Processor writes 15 normal packets into the output port. As WAITIN_P is low, packets are sent out immediately and locations are re-used (0, 1, 2). OUTRDY_P remains high. While the third packet is being sent, WAITIN_P goes high and delays sending of the next packet. WAITIN_P goes high at the end of the transmission of the 7th packet and suppresses START.

WAITIN_P changes back to low and the remaining 8 packets are sent out in the same order in which they were written to the buffer (they are all normal packets). NORM_EMP and CSP_EMP are both high (the output buffer is again empty); idle patterns are sent.

HW_reset because of the input buffer (constant '5555' is sent out as idleword).

WAIT_IN is set and the processor writes first 8 cs packets in a row. As WAIT_INT is active, no packets are sent out. 8 of the 15 locations are now occupied (0,1,2,3,4,5,6,7).

Processor writes 8 more packets out to NIU: one cs packet, one normal and 6 more cs packets. For the first 7 packets the available locations '8...E' are used. After writing the Tag of the seventh packet to NIU, OUTRDY_P changes to low. But as the processor writes one more packet, WR_SPACERR goes high and increments the according error counter. FREE_BL delivers WBLOCKADR = '0' and WBLOCKSEL = '0000' if no location is free. Therefore none of the former packets is overwritten, because no block in memory is selected. But the EMPTY signal of the FIFO would change to high if there are 16 addresses in one queue. Therefore and because the 16th packet is lost, NIU has to be reset.

But here NIU was not reset and starts to send the first cs packet after WAIT_IN goes back to low. Then WFACK is active and the only normal packet (location 9) is sent. Then the normal queue is empty. As no CSWACKIN_P was received since the last cs packet was sent out, WFACK is still high and only idle patterns can be sent.

Stop of the clocks to go to TESTMODE (see instrumentation).

6.2.2 Stimuli for the Input Port

Idle: idle patterns received ⇒ no change in internal signals; nothing stored.

CS_packet received from network: WWORDSEL takes the values '000-001-002-...-400-800-000' as the 16 bit words are stored in memory; WPACSEL changes from '01' to '02 on '800' of WWORDSEL. UP causes INRDY_P to go high. WAIT_P stays low (buffer not yet full). As the CRC words are wrong (self-made !!) and USECRC is set, CRCERR goes high and increments the according error counter.

Normal_packet from network: written into location '02'; WPACSEL switches to the next location; no errors - CRC words right

Read one packet: NREAD_P low for two cycles; LBDATA_P drives bus; NOEHP changes twice ⇒ Tag (first cycle) and Value (second cycle) of the first packet are given out to the bus. DOWN is active for one cycle; INRDY_P remains high (still one packet in buffer). There is a short pulse on DOWN after the second cycle, but this has no effect, because DOWN is only used synchronously.

Wrong idle pattern from network: '555D'. As USEIDLE is set, IDLERR is high for the whole time '555D' is received.

CS_packet received from network. CSWACKNET_P is produced; WPACSEL changes to '08' and a CRC error is produced for the wrong CRC words. As the PE number was also wrong, PERR also goes high for one cycle.

CS_packet again received from network; new WPACSEL is '10'. There are now 3 packets in the input buffer.

Packets are received from the network: One normal, then one idle pattern, then one cs packet followed by two idle patterns, and two more normal packets without idle patterns between. Then WAIT_P changes to high (there are now 7 packets in the input buffer) while the first CRC word of the 7th packet is received, but (assuming WAIT arrives too late on the sending side to stop the next transmission) one more packet is received. WPACSEL changed '10-20-40-80-01'.

Processor reads 4 packets in a row: NOEHP changes 'FFF3-FFE7-FFCF-FF9F-FF3F-FE7F-FCFF-F9FF-F3FF' on every cycle of LCLK. WAIT_P returns to low after the Tag of the first packet is read, because the 8th packet is not yet completely received. So the counter changes from 7 to 6. Two cycles later, the 8th packet increments the counter. But as the processor continues to read, UP and DOWN are active at the same cycle ⇒ WAIT_P remains low.

Packet received from network; 5 packets in the buffer.

Processor reads two more packets from input buffer

Packet received from network; 4 packets in the buffer.

Processor reads one more packet from input buffer

Packet received from network

Processor reads seperately 3 packets; INRDY_P remains high, WAIT_P low. One packet is left in the input buffer.

3_packets are received from the network; the second is a cs packet and produces an acknowledgement. WAIT_P remains low; 4 packets are in the buffer.

3_packets are received from the network; the second is a cs packet and produces an acknowledgement. WAIT_P goes high at the end of the last packet; the buffer contains 7 packets.

Processor reads one packet; WAIT_P goes low as there are now 6 packets in the buffer.

2_packets are received from the network; the second is a cs packet and produces an acknowledgement. WAIT_P goes high at the end of the first packet; the buffer contains 8 packets now.

Processor reads 9 (!) packets from input buffer. After the Tag of the second packet is read, WAIT_P changes to low. Then six more packets are read normally from the input buffer. After the Tag of the last packet, INRDY_P goes low. Normally, the processor should stop reading now, but one more packet is read. This procuces a RD_SPACERR and the controller gets confused: INRDY_P and WAIT_P go high.

HW_reset is necessary after reading from the empty input buffer.

Clocks stopped to simulate TESTMODE for Instrumentation.

6.2.3 Stimuli for the Instrumentation

Initialization as described in the beginning.

Errors are detected during operation: CRCERR causes ERR_P to go high, but there is no immediate reaction. Then a CSACKERR, and an IDLERR occur. There is also an EXTERR_P coming from a link. Then a PERR and 4 more CRCERR happen before the errorcounters are read and reset.

Counting of statistics is enabled during the initialization. But it is very important that the counter can be disabled while NIU performs normal operation whithout destroying the contents. So '07' (disable) and '17' (enable) were written to status register low ('00') alternatingly during the simulation.

Read error counters: First, counting is disabled (write '07'→'00'). Then, error count low and high are read ('06': '54', '07': '06'). '0101 0100' ⇒ 1 idle error, 1 PE error, 1 cs acknowledgement error and 0 network space error. '0000 0110' ⇒ 0 write space error, 0 read space error, 1 external error and 3 or more CRC errors. Then the error counters are reset.

Read statistics counters: addresses '0A', '0C'... '2E' give the contents of the 20 counter portions. For example, '0C':'0A':'08' give the number of normal packets sent out. ⇒ '000013' = 19 packets. Then counting is enabled again.

NRES resets the error counters and ERR_P changes to low.

CS_ACKERR happens and sets ERR_P shortly before the error counters are reset by a write to the diagnostic port.

WR_SPACERR occurres and causes ERR_P to go high again.

Stop all clocks to do a test increment for all counter portions

TESTMODE is set ('09'→'01') to split up the counters into 8 or 4 bit portions. This increments some of the counter portions, depending on their input lines. Therefore the counters should normally be reset after changing to test mode (but it isn't done here).

TEST_INC is activated some times by writing don't-care-values to address '0A'. The counter portions are incremented by each falling edge of TEST_INC (4 bit portions in their upper and lower parts independently).

TESTMODE is reset again. Important: there are no changes on the states of all counters. So it is possible to bring the counters into a defined state using TESTMODE, disable TESTMODE and use this state for further sampling. This is a possible way to test these long counters.

Disable counting, read address '06' (error count low; no errors here) and reset the error counters.

Read all counter locations from the diagnostic port.

6.2.4 Slow Processor Clock

If LCLK_P is below 22 MHz, the FREE_BIT according to every location in O_MEM may not be released again after a packet has been sent out. The reason is that RBLOCKSEL has already changed before the release signal is synchronized to LCLK and combined with RBLOCKSEL. Therefore, SLOW_MODE must be set. Now OUTCTL waits with deselecting RBLOCKSEL until REL is synchronized. Also WAIT_P is activated whenever one packet has arrived to stop PaRC from sending another packet before the count signal for the first one has been synchronized. To show the correct behaviour, version 'S' of the simulation was performed.

Initialization is done like in the main simulation with the exception that SLOW_MODE is set ('1F'→'00')

Processor writes one packet into the output buffer. A free memory location is occupied ('0') and Tag and Value are written there. The address is written to the normal queue. Then NORM_EMP changes to low and OUTCTL starts the transmission of the packet. In order to release the FREE_BIT of the memory location, the signal REL is synchronized to LCLK. This may take some time as LCLK is slow. Therefore the signal WAIT_SYNC is active whenever SLOW_MODE is set and REL is being synchrionized. WAIT_SYNC has the effect that START goes back to low after the transmission of a packet but RBLOCKSEL is kept stable. This is done by delaying the signals NORM_OUT, CSP_OUT until WAIT_SYNC goes back to low. Also ENB (selects address from cs or normal queue) is kept stable until WAIT_SYNC changes to low.

Packet is received from network. CSWACKNET_P is high, because the packet was circuit switched. WAIT_P goes high while word 2 is received. In SLOW_MODE, WAIT_P must come early enough after every packet to stop PaRC from sending the next packet immediately after the first, because the count signal for the packet must be synchronized to LCLK (UP) before that.

Processor writes 3 packets in a row to NIU. As WAIT_SYNC is still active from the last sent packet, the first of the new packets is not sent immediately. Only after the falling edge of WAIT_SYNC START comes up again and one packet is sent. WAIT_SYNC changes to high again and delays the next START until FBL/RELINT has set the matching FREE_BIT. Then the next packet is sent and the same happens. Meanwhile, the processor writes one more

cs_packet. This packet is available to be sent out when WAIT_SYNC of the former normal packet goes back to low. Therefore, not the last normal packet, but this cs packet is transmitted next. WAIT_SYNC again goes high until the synchronisation is done. Finally, the last normal packet is sent. Then both queues are empty. In parallel, another

packet is received from the network. Again, WAIT_P goes high while word 2 is received and idle patterns are received then. The processor reads the first packet from the queue and INRDY_P is low until the next packet is received.

Processor writes 6 packets in a row to NIU. The first and the 5th are cs packets. As WFACK is still active, the second packet is first sent out, then the third. There are always gaps between the packets, because the REL signal must always first be synchronized. After the four normal packets are sent out, no more packets can be sent because the acknowledgement for the first cs packet has not yet been received.

CSWACKIN_P resets WFACK. In the meantime, WAITIN_P is high and inhibits sending of packets. Then WAITIN_P goes low and allows the next cs packet to be sent out. Again, WAIT_SYNC goes high and keeps RBLOCKSEL constant until REL is synchronized.

5_more packets are received from the network. After the first 3 packets there are always idle patterns as long as WAIT_P is active or longer. All these packets are counted as usual. But the fifth packet starts too early and the signal that is synchronized to the LCLK domain is reset together with that of the fourth packet. This causes the input counter running on LCLK to be lower than that on XCLK side ⇒ packets are lost!

The simulation is carried on, but normally a HW reset must be performed. There is no HW error detected in that case - NW_SPACERR is only signaled if the buffer is full and a packet arrives. But here the buffer is not full - only sychronisation is too slow.

6.2.5 Lower Clock Boundary

If the frequency of LCLK is below 22 MHz, SLOW_MODE must be set and the controllers for input and output behave in a different way. But there is still to prove that NIU operates correctly at 22 MHz for LCLK without SLOW_MODE, as main simulation uses 33 MHz for LCLK. This is done in simulation version 'C'. There is just one central point: the synchronized RELease signal must be used together with the RBLOCKSEL (OCLK domain) signal to set the according FREE_BIT. Only after this is done RBLOCKSEL may change. But it was easier to change a large part of the main simulation to the slower clock rate than to construct just this worst case.

RBLOCKSEL and (FBL)RELINT are used to manipulate the D input of an edge triggered FF, which is clocked by LCLK. Therefore RBLOCKSEL must not change before (FBL)RELINT is high and a rising edge of LCLK occurres (plus hold time). The most tight case during this simulation (in total 31 packets sent out) was at timestep 56178: RBLOCKSEL changed 7.9 nsec after the according rising edge of the clock.

7 Floorplan

All timing behaviour described up to here is based on calculations that only consider the number of receivers. The length of wires is not yet known, as none of the cells nor the I/O pads are placed. Therefore it is also not yet sure if the proposed design will fit on the planned 75k master. LSI Logic's program 'lpace' provides this kind of information. It allows to place functional blocks on the chip and checks the resulting routability. There is also an 'autoplace' function, but it showed that the results are not satisfying. Also the pad and pin connections had first to be established. This is not quite trivial, as there are 72 8mA output drivers for the local bus. One pair of VSS/VDD pads can only drive 5 of these BD8T cells. So they were spread around the lower left side, the lower side and the lower right side. Additional VDD/VSS pairs were established compared to PaRC between the corners and the centers of a side. Important: as there was not much time left to do the pinout and the floorplan and especially as the 'check' command of the program 'lbond' did not work, pinout and floorplan should be checked carefully again to make sure that no design rules are hurt.

Network input and output pads were placed in the upper left and right corner. There are not as many critical drivers, so the VDD/VSS configuration looks like PaRC in these areas.

The strategy for placing the modules on the chip was as follows: output memory (instance name OM) in the upper right corner next to the network output pads; the input memory (IM) in the lower left corner near to the local bus pads. The controllers and smaller modules of the output and input port below and left of OM; especially LBCTL (instance name LBC) in the center of the lower side, because it contains the input register from the local bus. The input register from the network is placed togeter with the checkers in the upper left corner next to the input pads. Below that INSTRUMENTATION and on the left hand side the RAM for lookup and the pads for the diagnostic port. The network clocks were placed next to the according data pads. The pad cells for LCLK and NREAD (output enable for the 72 bidirectional pads to the local processor bus) were placed in the middle of the lower side. So the maximal skew on NREAD is only 2 nsec - even without being specified as a clock signal. It showed that the path from the network output register to the CRC generator and back to the output register is too long if this submodule (U10) of OM is not placed. So all submodules of OM had to be placed, but then all signals were fast enough. Despite this overspecification, routability shown by 'lpace' is 98 and that means that there should be no problems at all with layout. But this number is a bit doubtfull, because a routability of just 85 was shown in an earlier session with a quite similar placement. In general, it was often hard to understand why this program sometimes complained and sometimes not.

Based on this placement, new delay times were calculated and the simulations were repeated. There were no more problems after (OM)U10 - the output register to the network and the CRC generator - had been placed seperately. Most internal signals even became faster. The following timing specifications are based on the current floorplan (only the critical ones are listed):

Local Bus

OUTRDY_P	stable 24.5 nsec after rising edge of ext. clock
INRDY_P	stable 21.7 nsec after rising edge of ext. clock
LWR_P	requires 9.0 nsec setup time before rising edge of ext. clock (10.5 nsec before int. clock)
NREAD_P	requires 12.0 nsec setup time before rising edge of ext. clock (10.5 nsec before int. clock)
LBDATA_P	output enabled 13.0 nsec after NREAD; new data stable 21.0 nsec after rising ext. clock

Especially both times of LBDATA_P could be improved. The relatively long output enable time is due to the high fanout (long wiring and high load on the driver). Both effects could be minimized be introducing a second pin and driver for the NREAD signal. These two inputs should then be placed in the lower left and lower right corner. Also the time to change from one output data to the other on LCLK_P can be brought down by placing the DRV8I drivers IM/U3 and IM/U4 nearer to IM (at the moment they are on the other side - just because there is not enough time left to move them).

8 Test Strategies

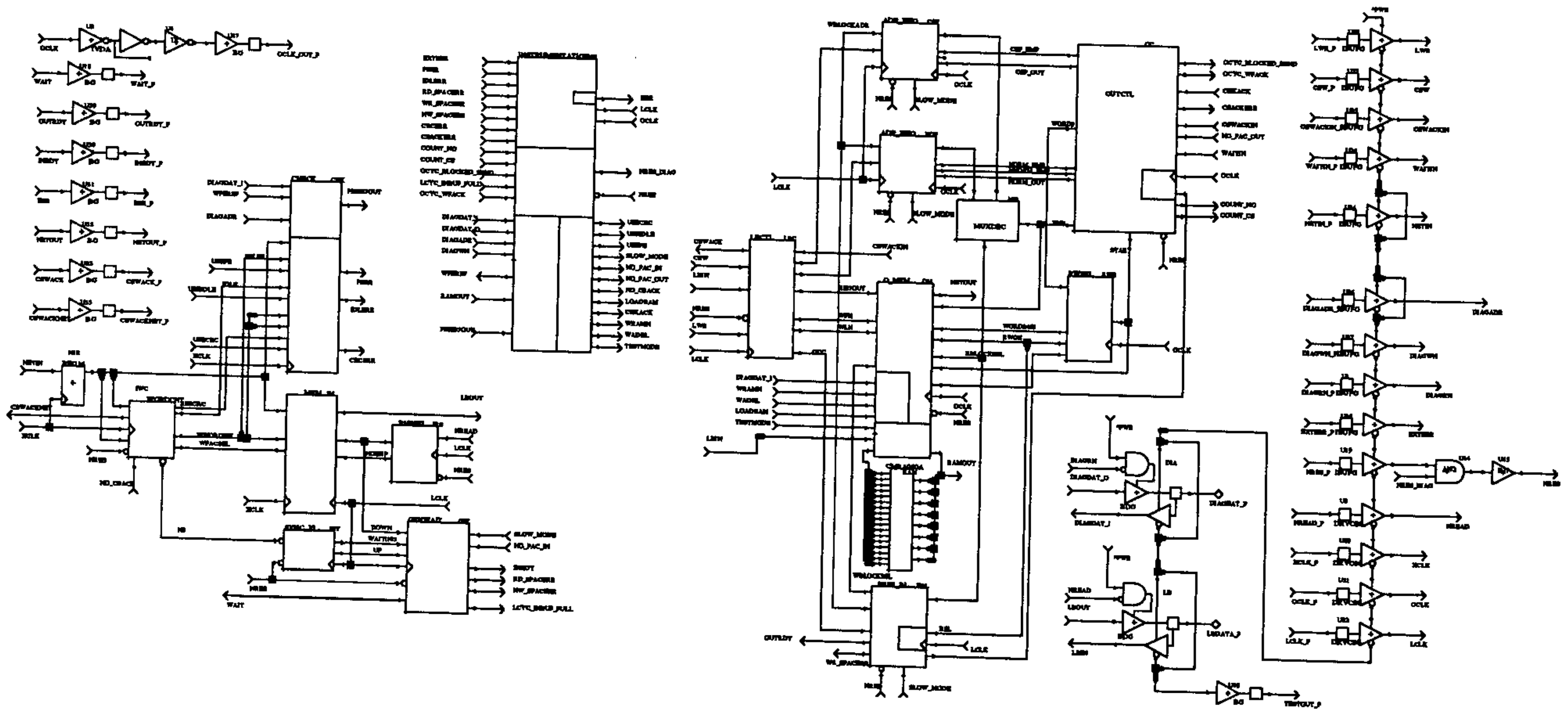
As most parts of NIU consist of memory (soft RAM), most stuck-at faults should be found just by sending packets with random data to and from NIU. Most of the controllers are tested by this too, as the right select and output enable signals are essential for the correct transmission of even just one packet.

More critical are the modules which generate the handshake signals: GENWAIT and FREE_BL. In order to check these, the input and output buffers have to be filled and emptied again. Even more critical are the address FIFOs. They contain small hardwired 16X4 RAMs. LSI Logic provides testpattern for those macro cells, but only if they are directly accessable. At the moment, NIU does not provide this. It should be possible to test these RAMs nearly completely by filling and emptying the output buffer once only with normal and once only with cs packets. This procedure will have to be repeated several times to write some different addresses to all locations in the RAM. In general, faultgrading would be a very useful tool, because it is likely that the fault coverage after this kind of test is already very high - with the exception of two blocks: The large lookup RAM 256X8 and the counters for statistics (24 or 28 bits long). There is a way to connect all inputs and outputs of the RAM to external pins: TESTMODE (in status register) must be set. This connects the address input of the RAM to LBDATA_P(7:0). Data input of the RAM is connected to DIAGDAT_P anyway and WRN of the RAM can be activated by writing to address '03'. The output can be read from location '04' of the diagnostic port. For testing the statistics counters, there is even more HW support: if TESTMODE is set, the counters are broken up into portions of 8 (or 4) bits. Each of these portions can be incremented independently by a write to location '0A' of the diagnostic port. So it is possible to bring all 28 bits of these counters with just 255 write operations in the state 'FFF FFFF'. If NIU returns to normal operation afterwards and some events happen, the right function of all carry signals within a counter can be proved by reading the contents later (it must be in the state '000 00XX').

As there are still some pad cells and pins available, it would also be possible to define some more test pins for signals which are not easy to watch during normal operation. As the gate count is not too high at the moment, it seems also possible to spend additional HW for the two RAMs in the FIFOs to make them accessable from external pins and use LSI Logics testvectors for them.

9 File Structure

This documentation can be found in the directory '/jj/hutner/doc' in the file 'niu.tex'. All files for the design itself are in the directory '/jj/hutner/chip'. There are all the files '*.def', which contain the schematics . The toplevel block of the design is 'CHIP', so there are a lot of files with that name, most of them for the different versions of simulation. There are also the cross reference listing 'CHIP.WXRFLST' (contains the delay times based on the current floorplan), the '.SYSLOAD' file that contains the assumed load on the output pins. In 'CHIP.OPTSPEC', clock signals are specified and 'CHIP.BLOCKS' contains the block structure (output of 'lpace'). There are also the '0000BD.*' files. They are the output and input files of 'lbond' and 'lpace'. Finally, there are the libraries 'NIU.*' and the simulation files for the submodules.



M.I.T.

LL10075

DRAWN BY: FRANZ HUTNER

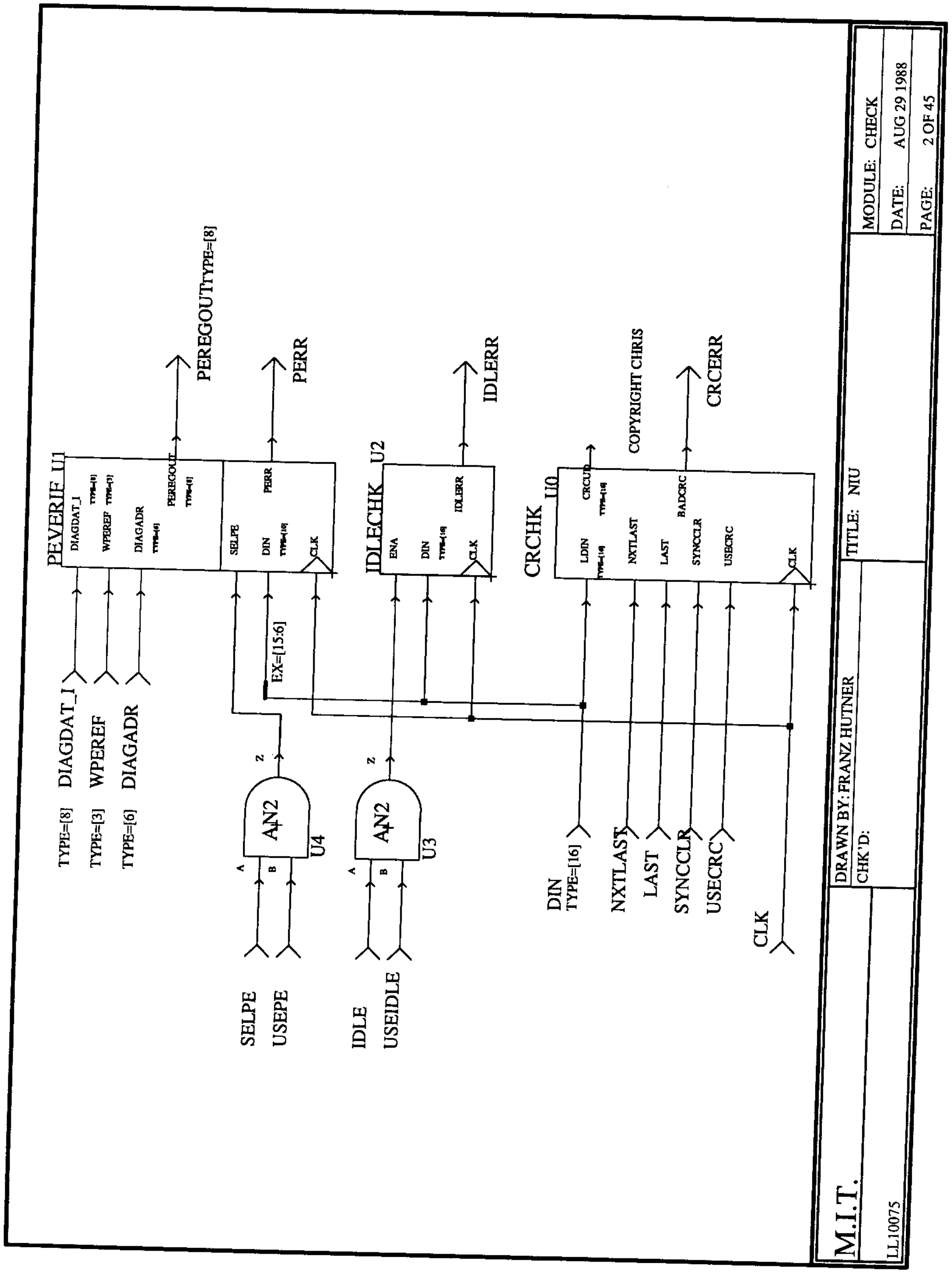
CHK'D:

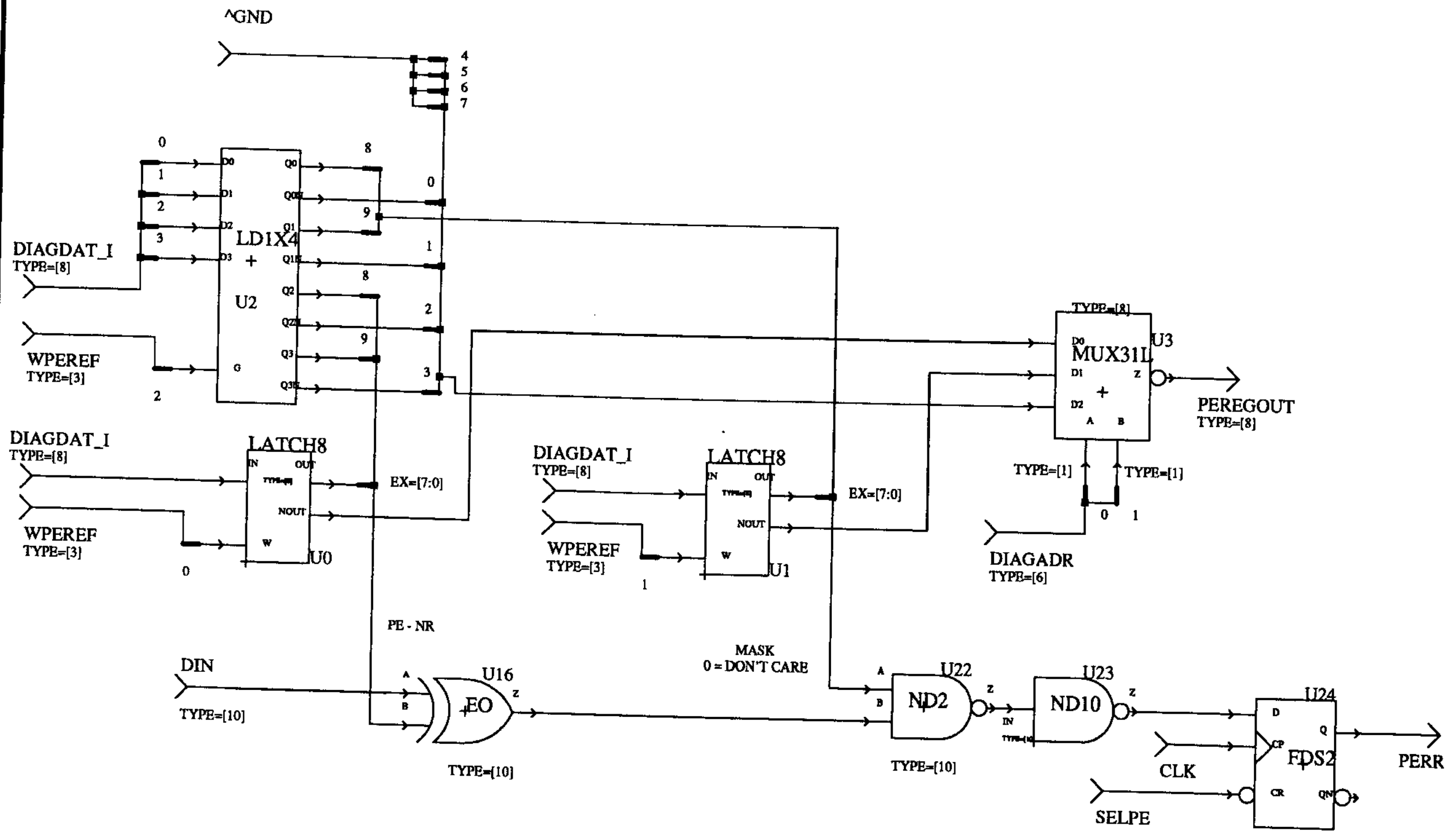
TITLE: NIU

MODULE: CHIP

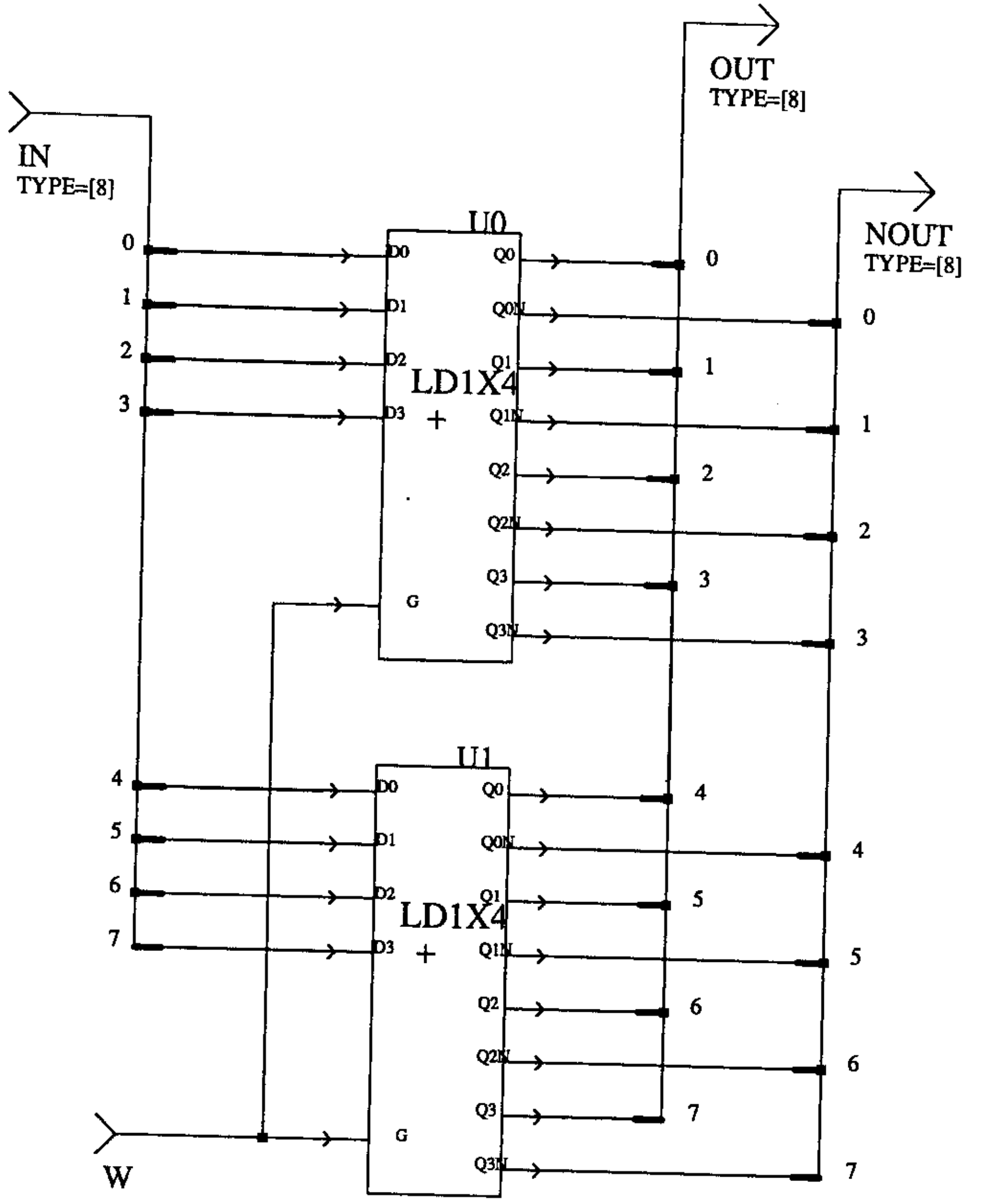
DATE: AUG 29 1988

PAGE: 1 OF 45

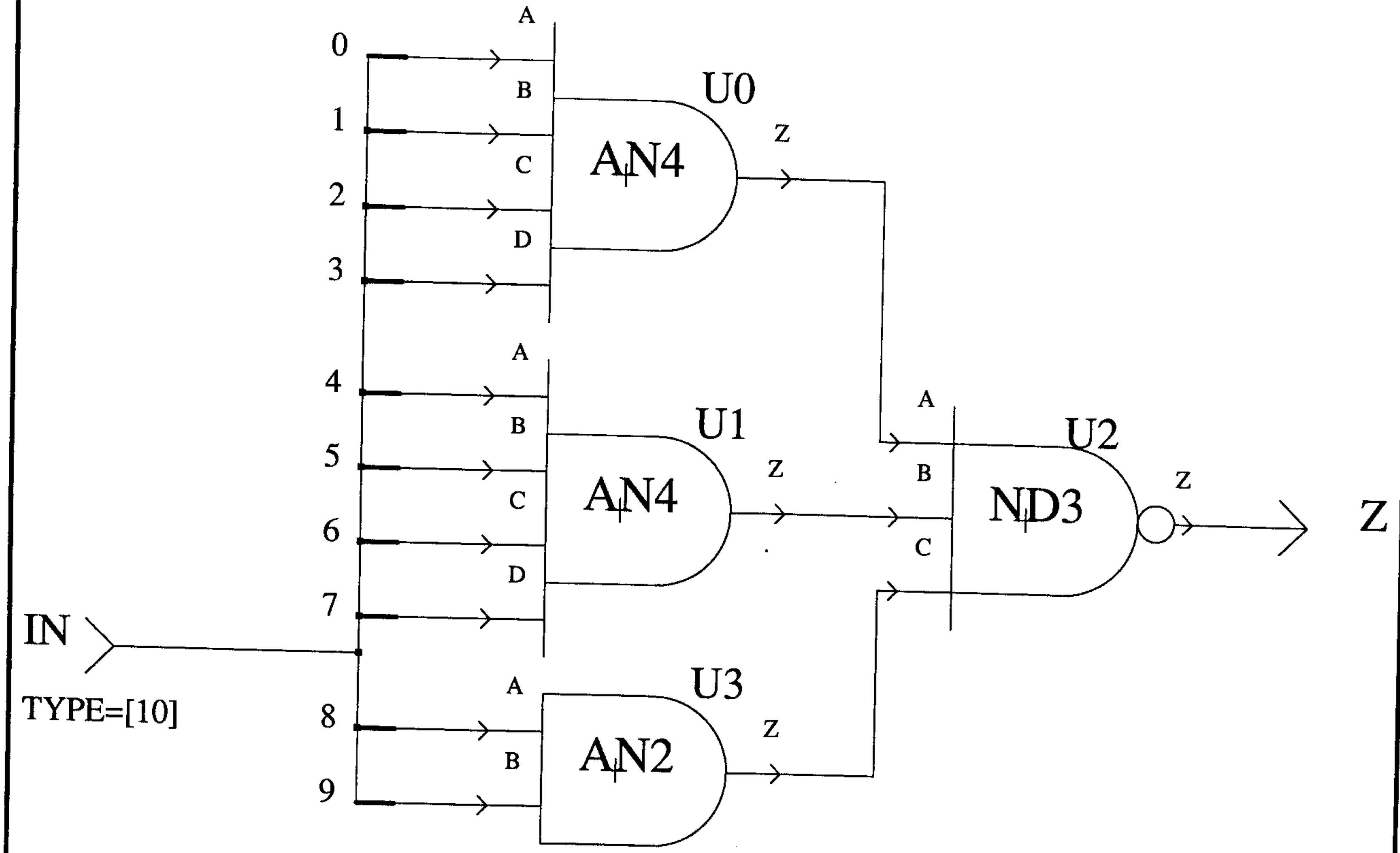

PEVERIF U1
DIAGDAT_I
WPEREF
DIAGADR
PEREGOUT
SELPE
DIN
PERR
CLK
PEREGOUT TYPE=[8]
PERR
IDLECHK U2
ENA
DIN
IDLERR
CLK
IDLERR
CRCHK U0
LDIN
NXTLAST
LAST
SYNCCLR
USECRC
CRCUD
BADCRC
CLK
COPYRIGHT CHRIS
CRCERR
TYPE=[8] DIAGDAT_I
TYPE=[3] WPEREF
TYPE=[6] DIAGADR
EX=[15:6]
AN2 U4
AN2 U3
SELPE
USEPE
IDLE
USEIDLE
DIN TYPE=[16]
NXTLAST
LAST
SYNCCLR
USECRC
CLK
M.I.T.
LL10075
DRAWN BY: FRANZ HUTNER
CHK'D:
TITLE: NIU
MODULE: CHECK
DATE: AUG 29 1988
PAGE: 2 OF 45



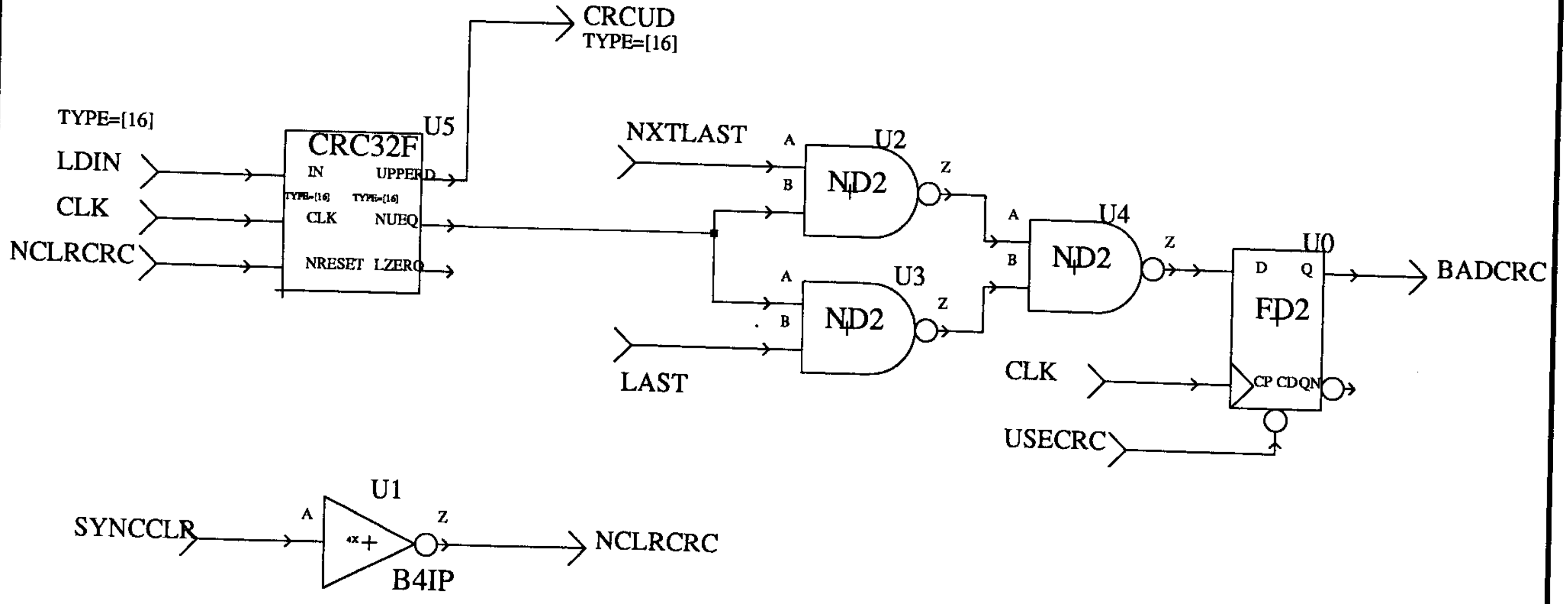
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: PEVERIF
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 3 OF 45



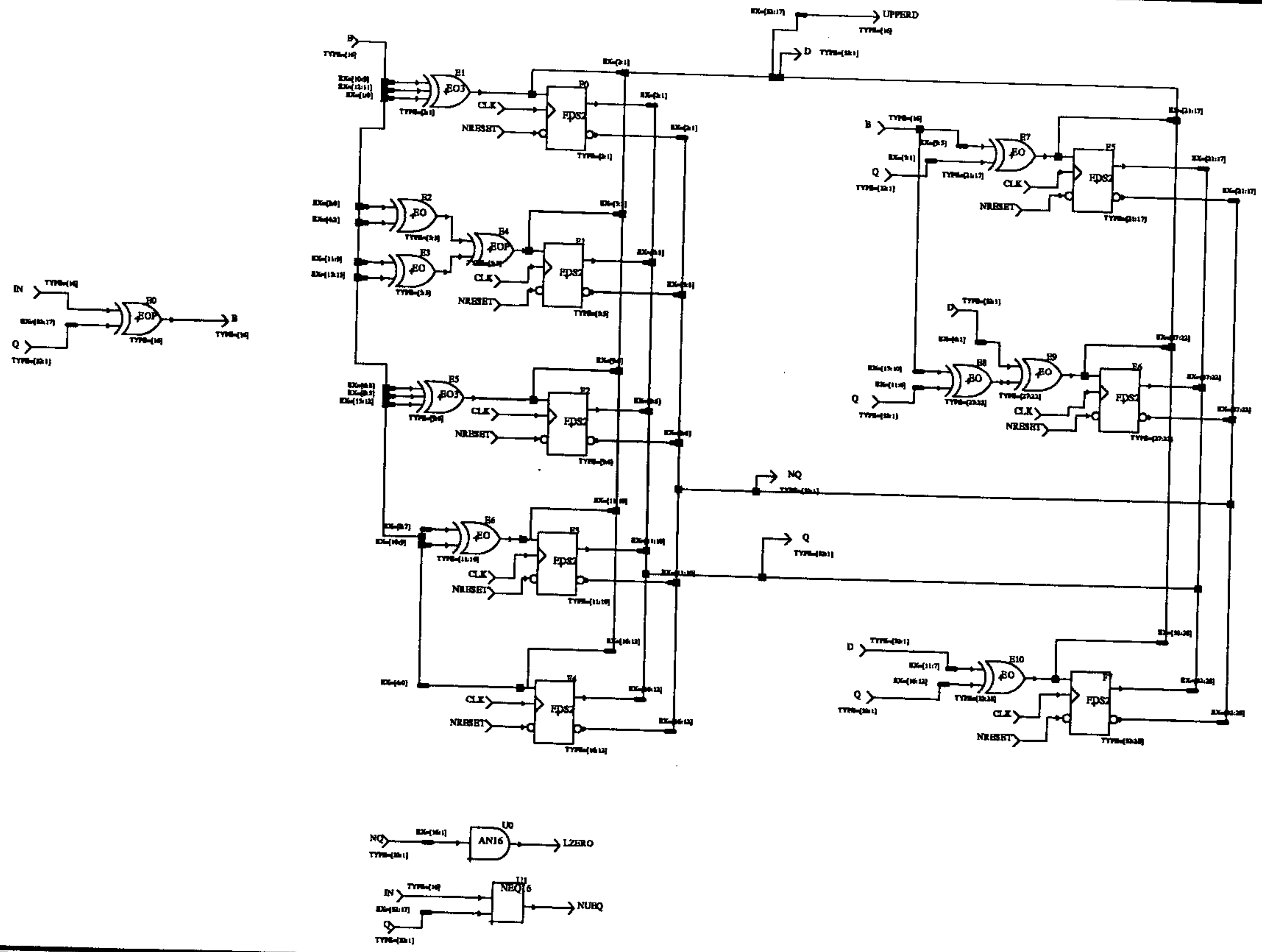
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: LATCH8
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 4 OF 45



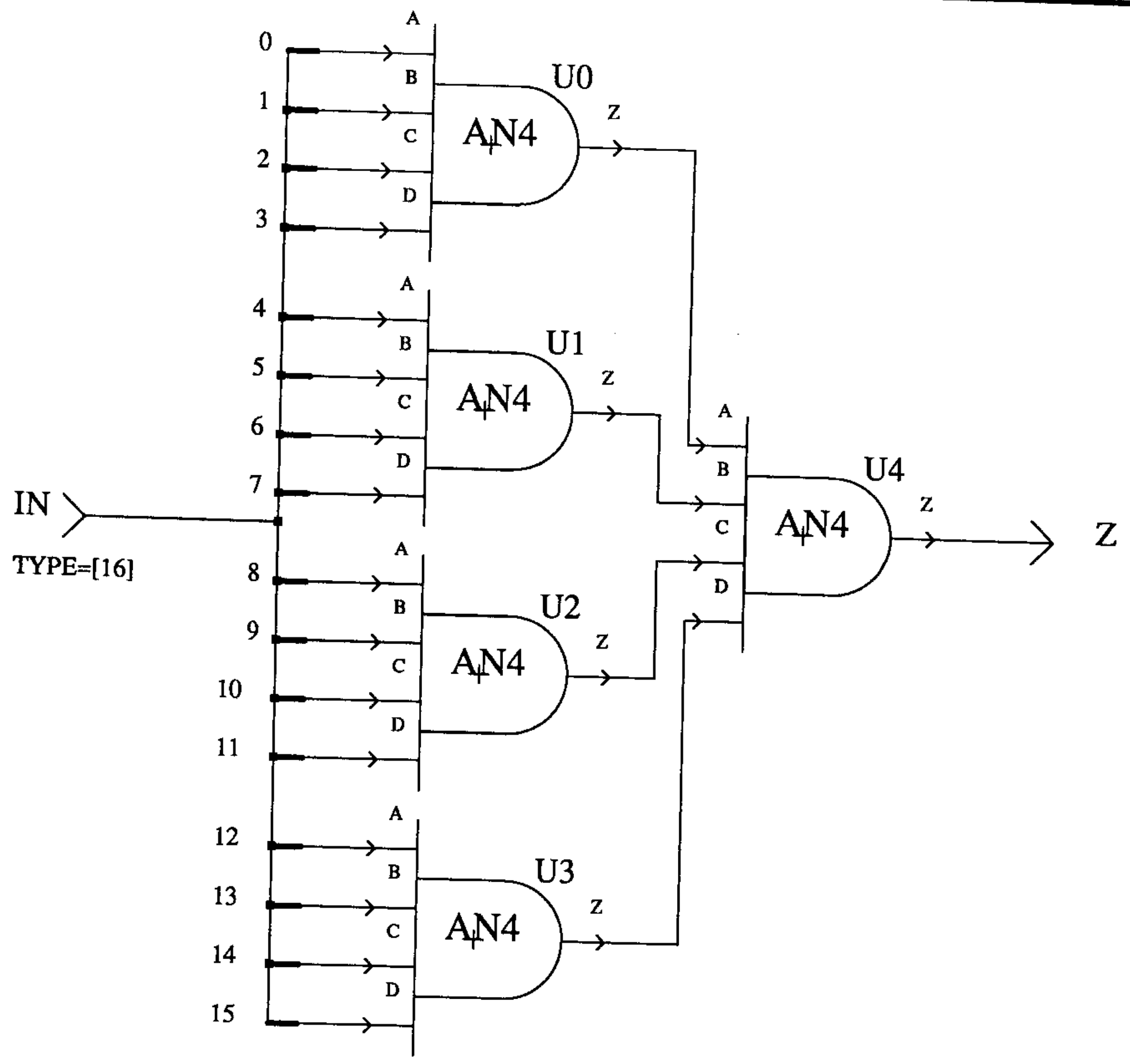
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: ND10
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 5 OF 45



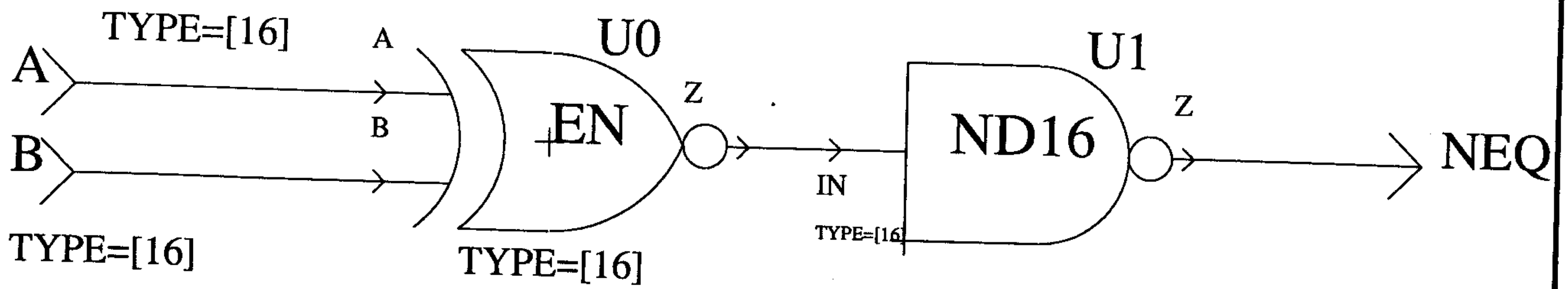
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: CRCHK
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 6 OF 45



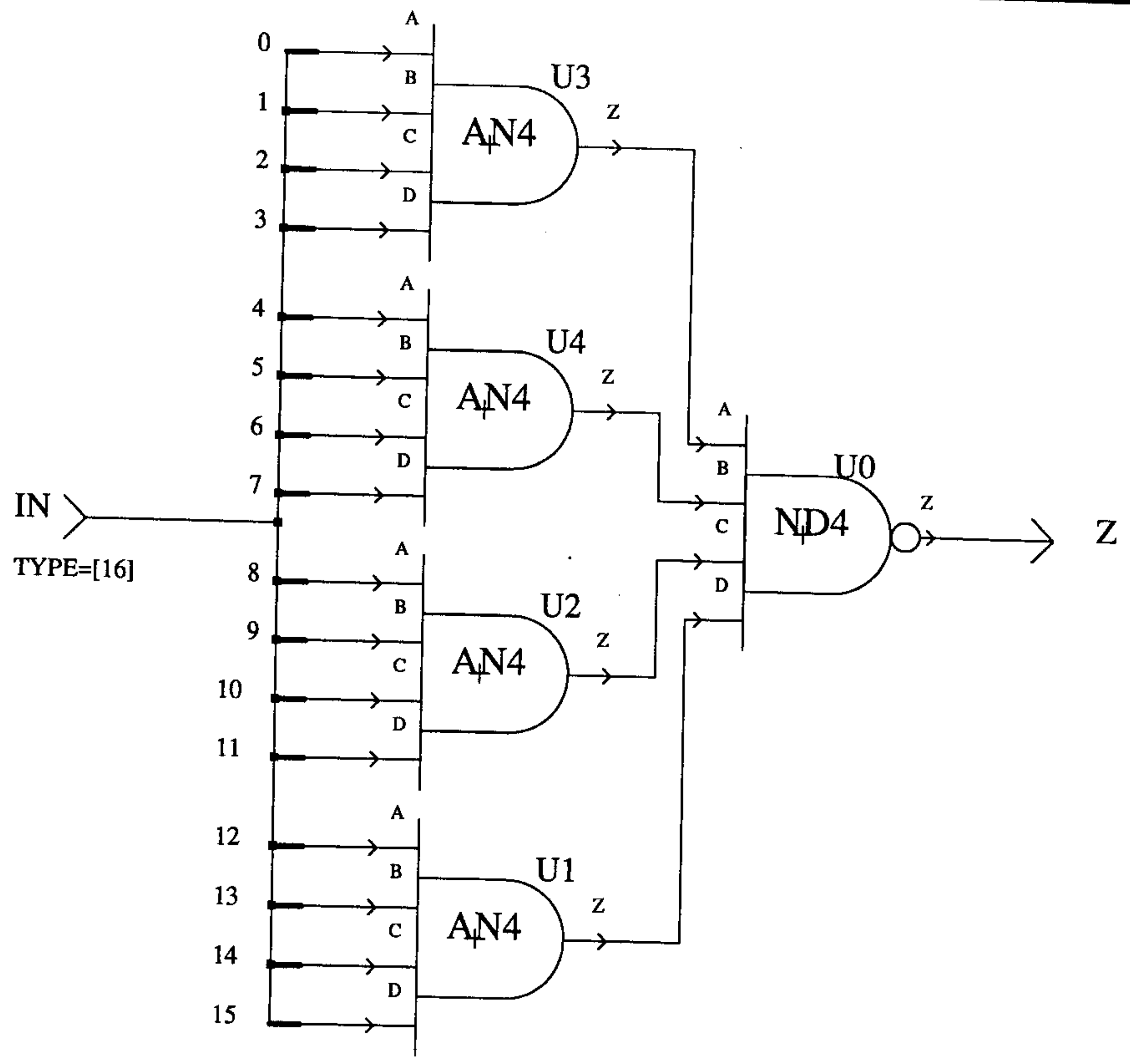
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: CRC32F
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 7 OF 45



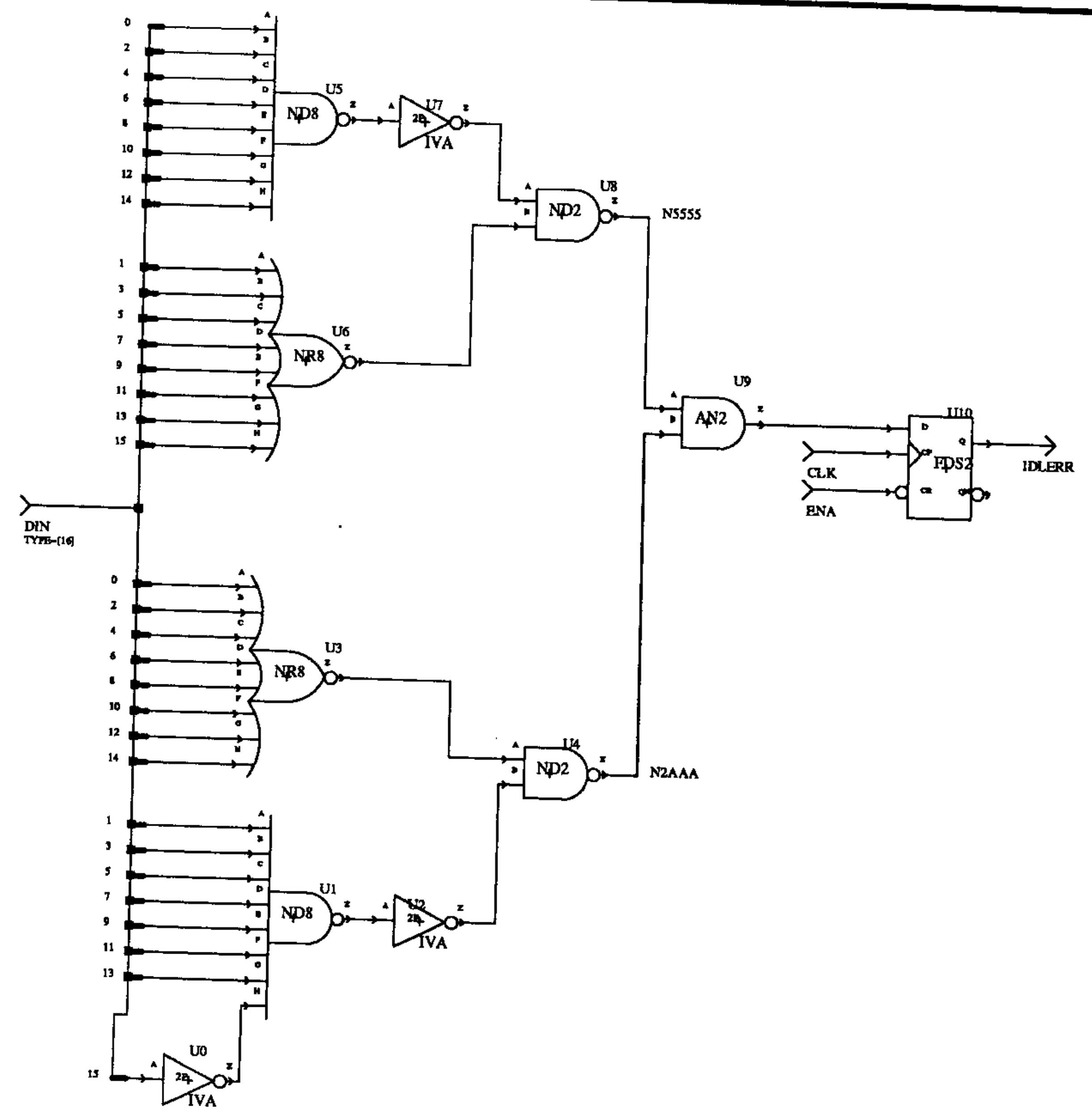
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: AN16
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 8 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: NEQ16
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 9 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: ND16
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 10 OF 45



M.I.T.

LL10075

DRAWN BY: FRANZ HUTNER

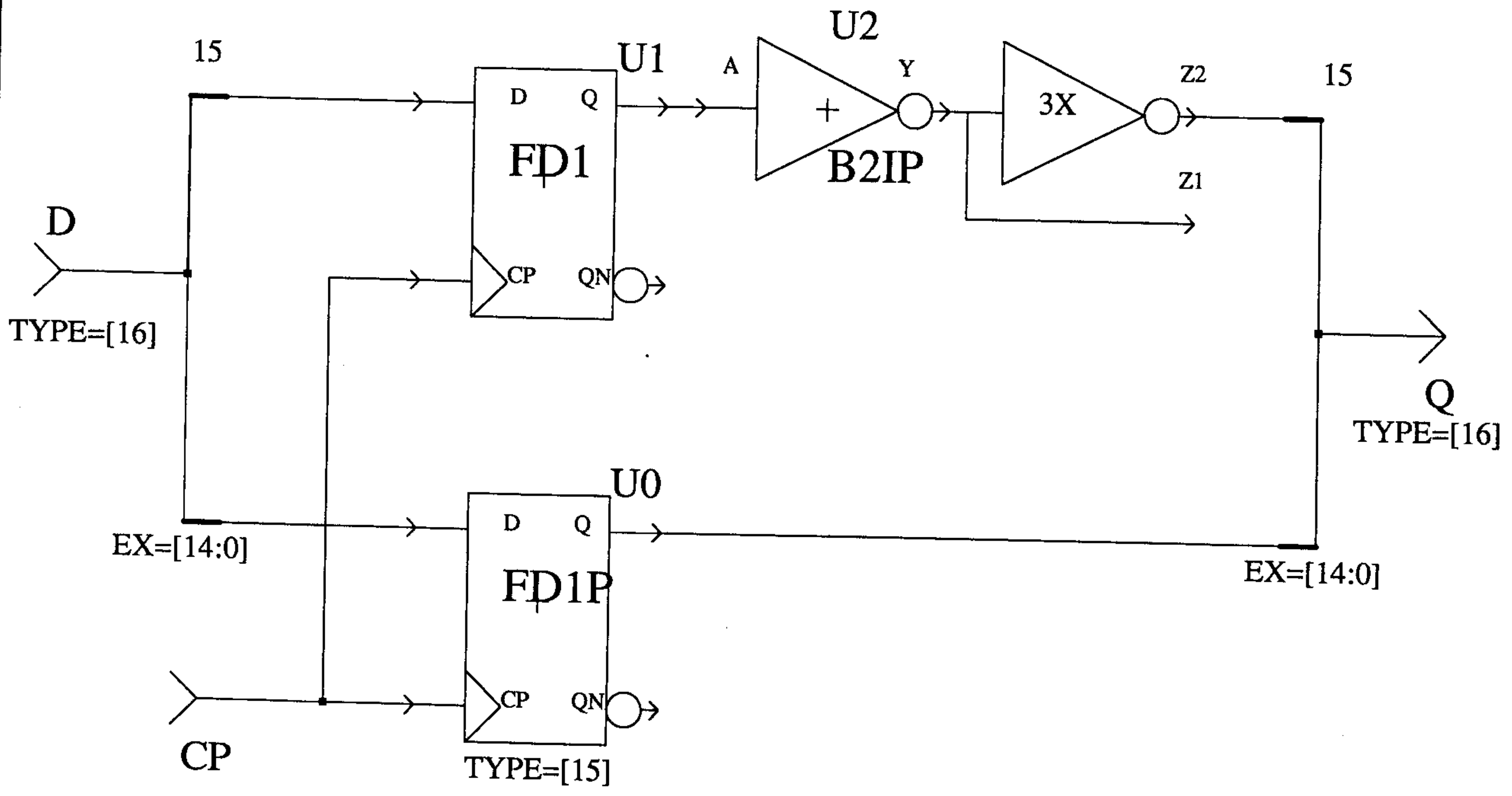
CHK'D:

TITLE: NIU

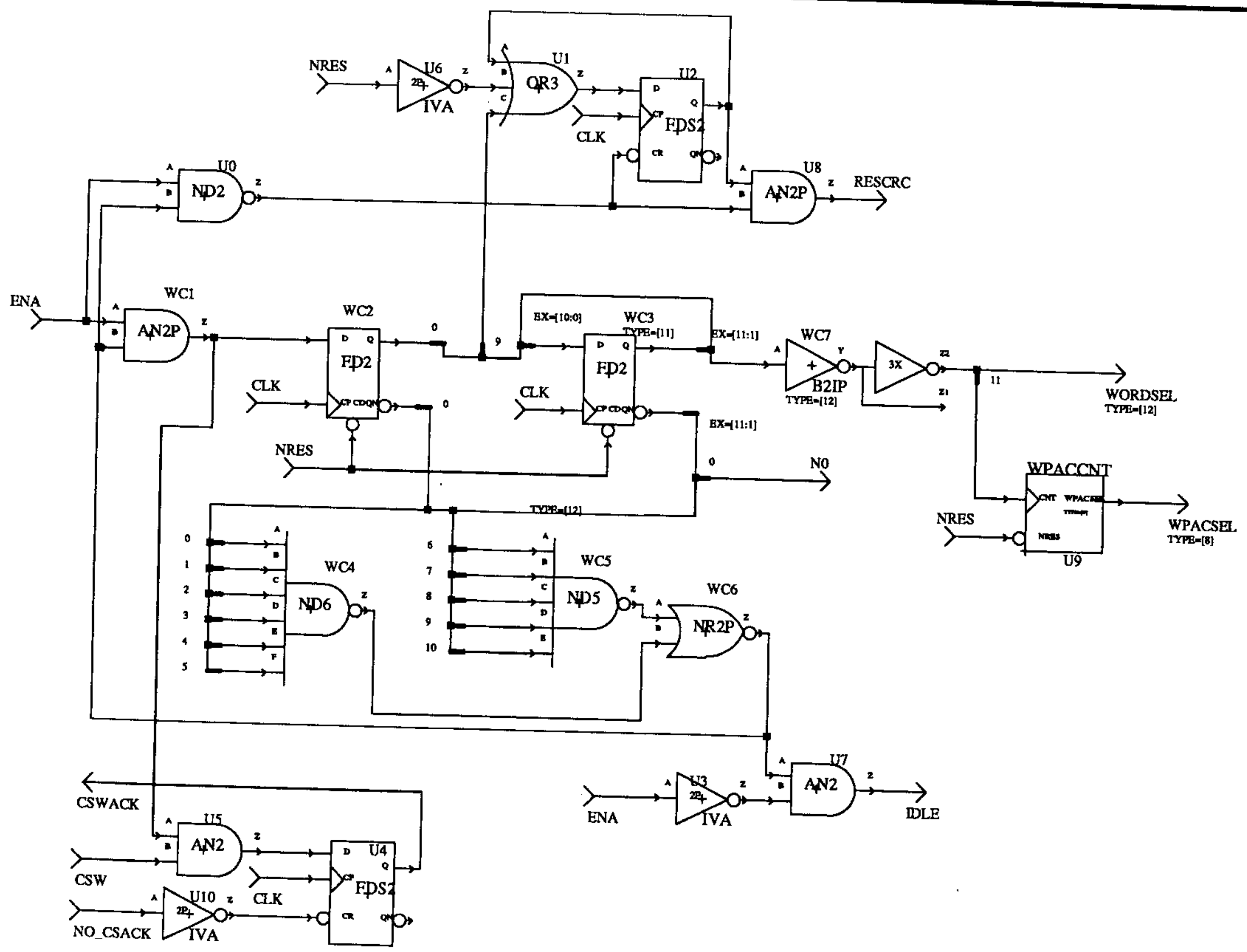
MODULE: IDLECHK

DATE: AUG 29 1988

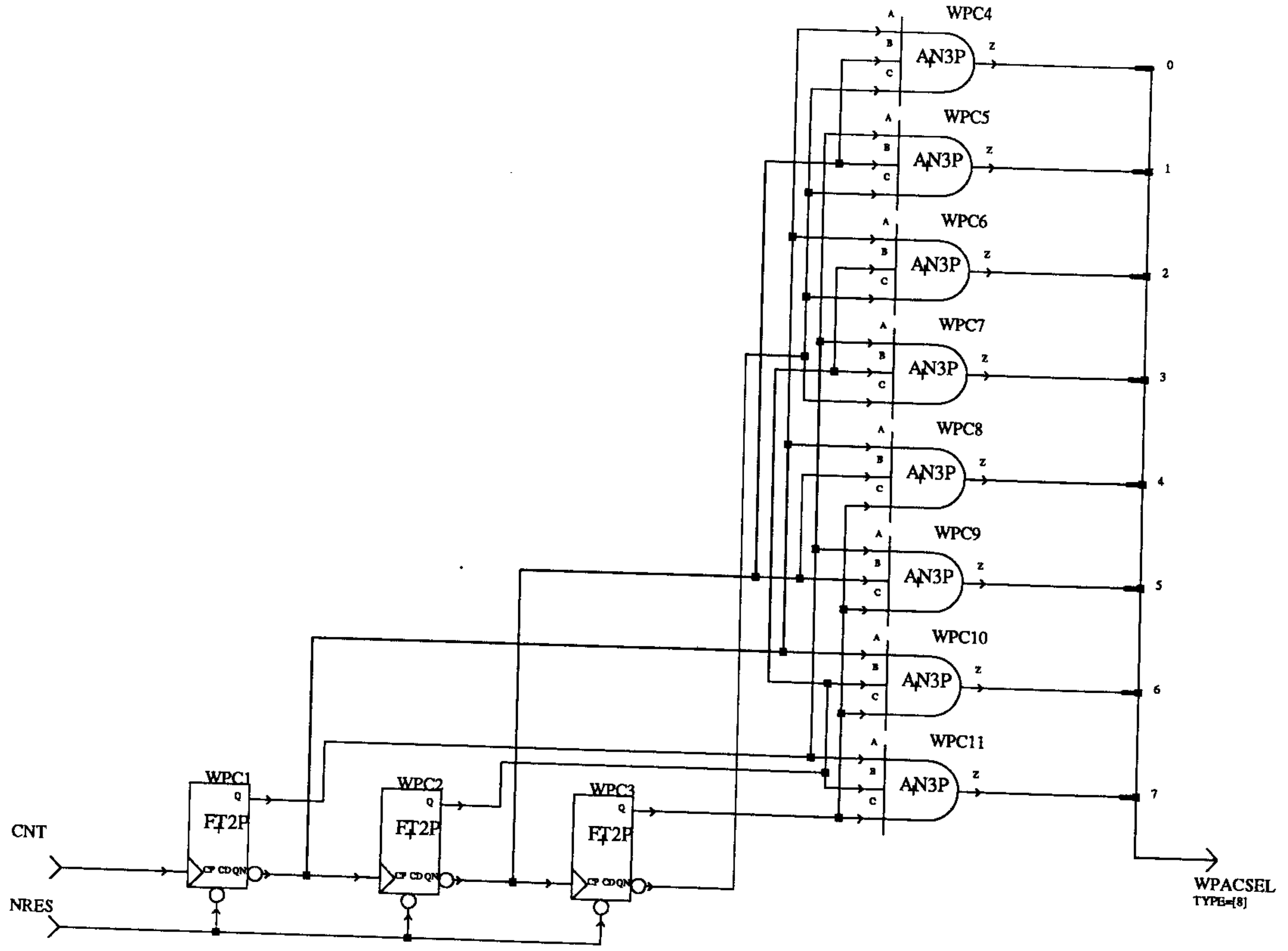
PAGE: 11 OF 45



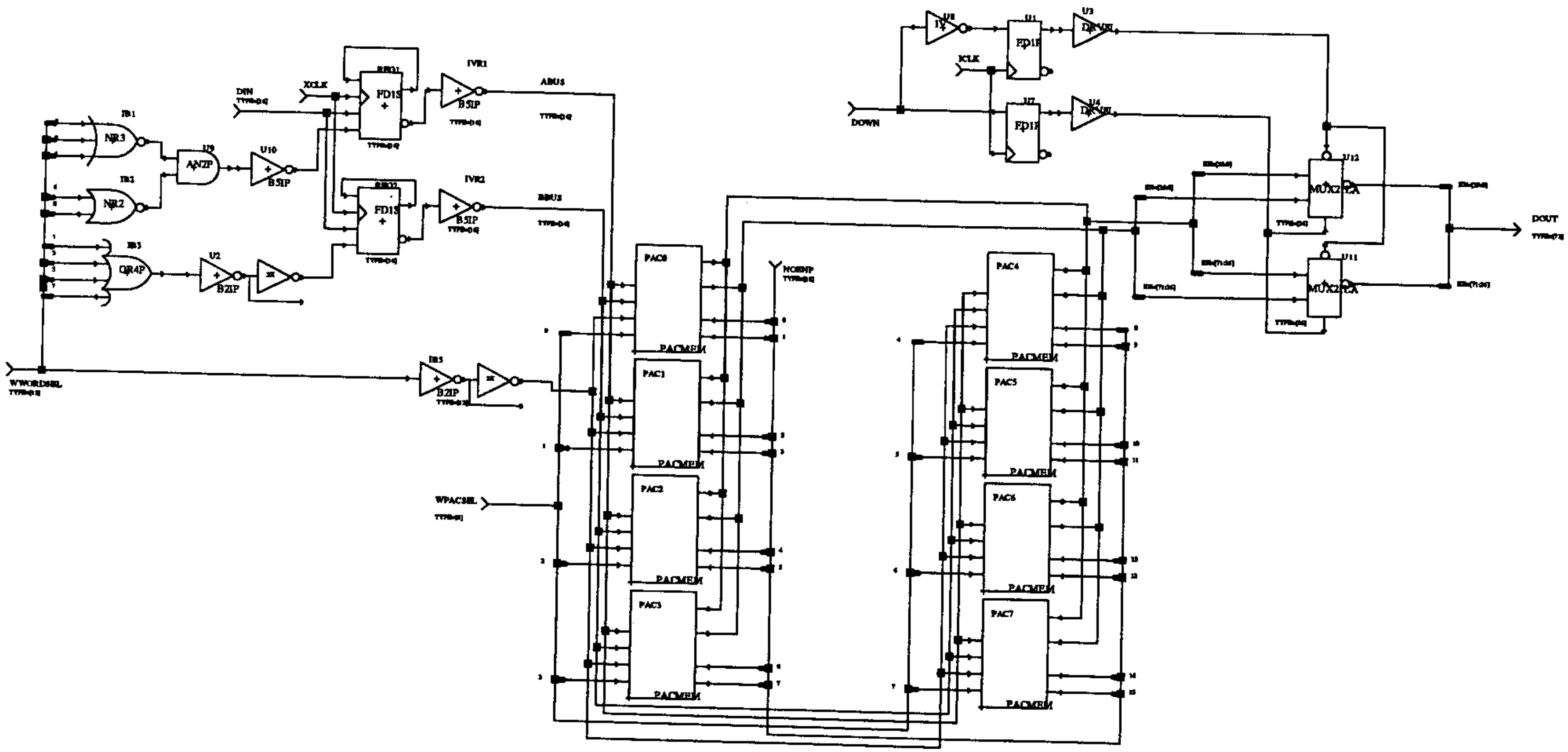
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: REG16
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 12 OF 45



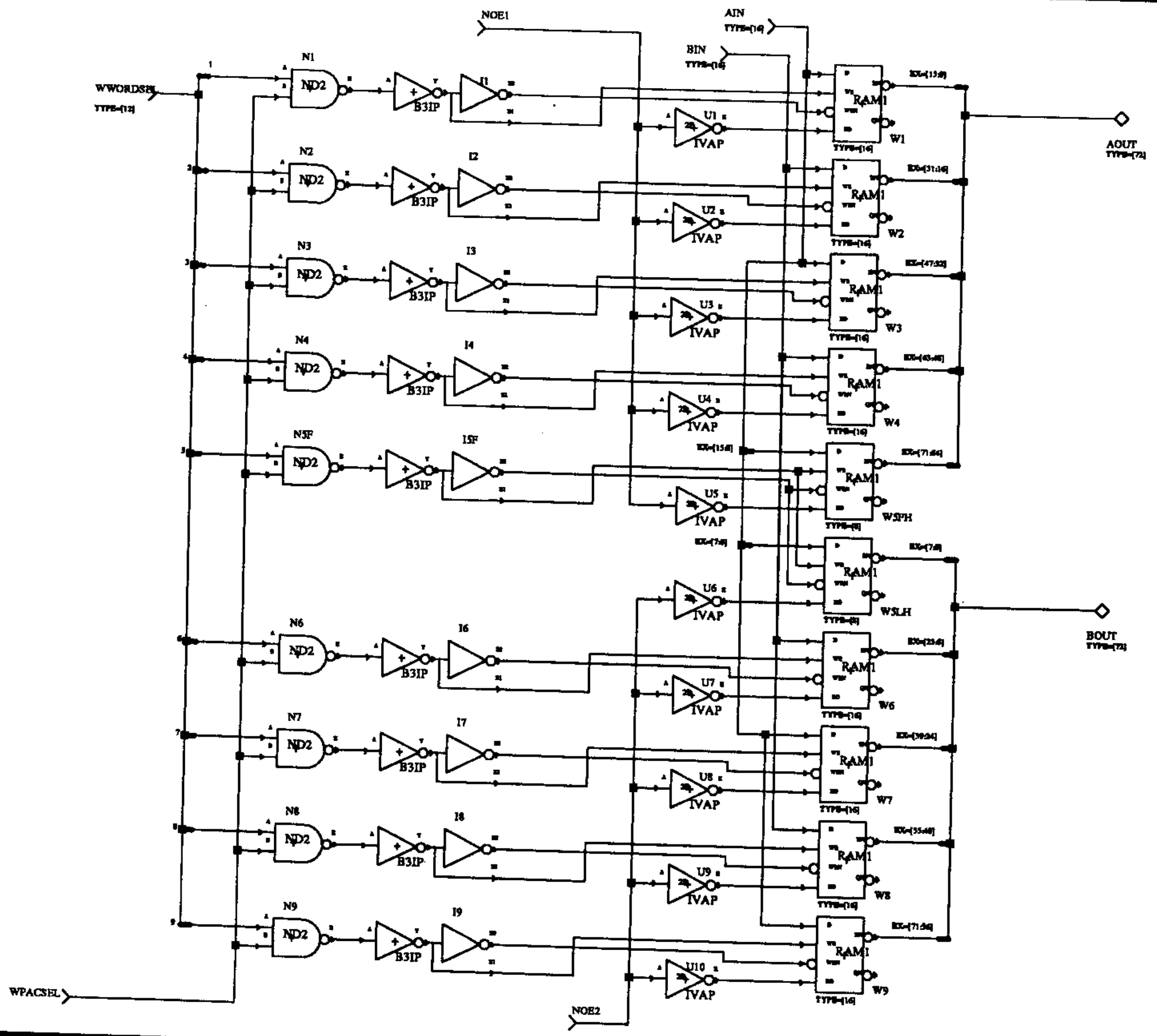
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: WORDCNT
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 13 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: WPACCNT
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 14 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: MEM
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 15 OF 45



M.I.T.

LL10075

DRAWN BY: FRANZ HUTNER

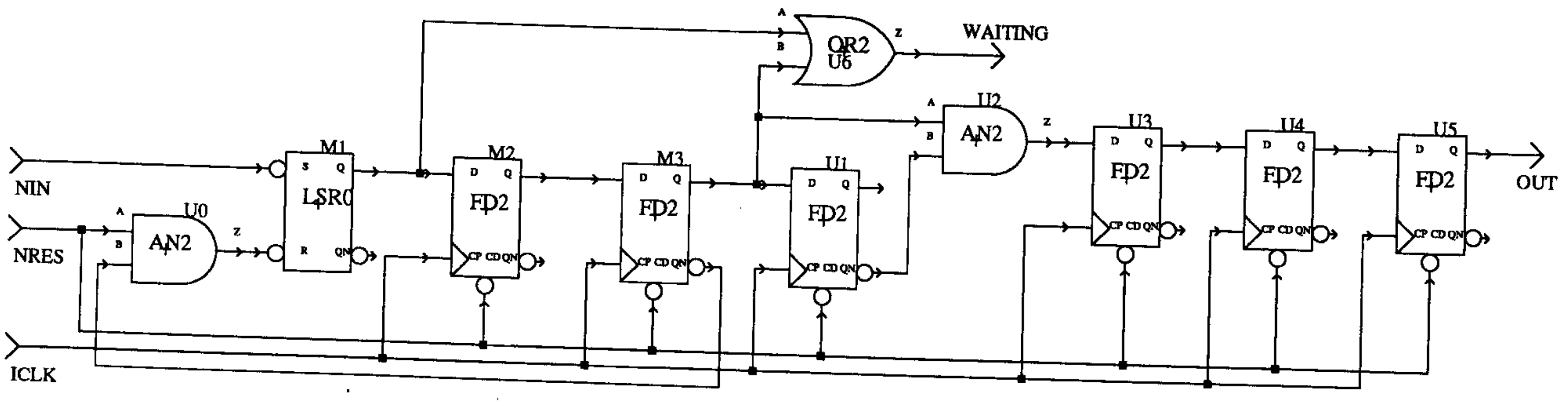
CHK'D:

TITLE: NIU

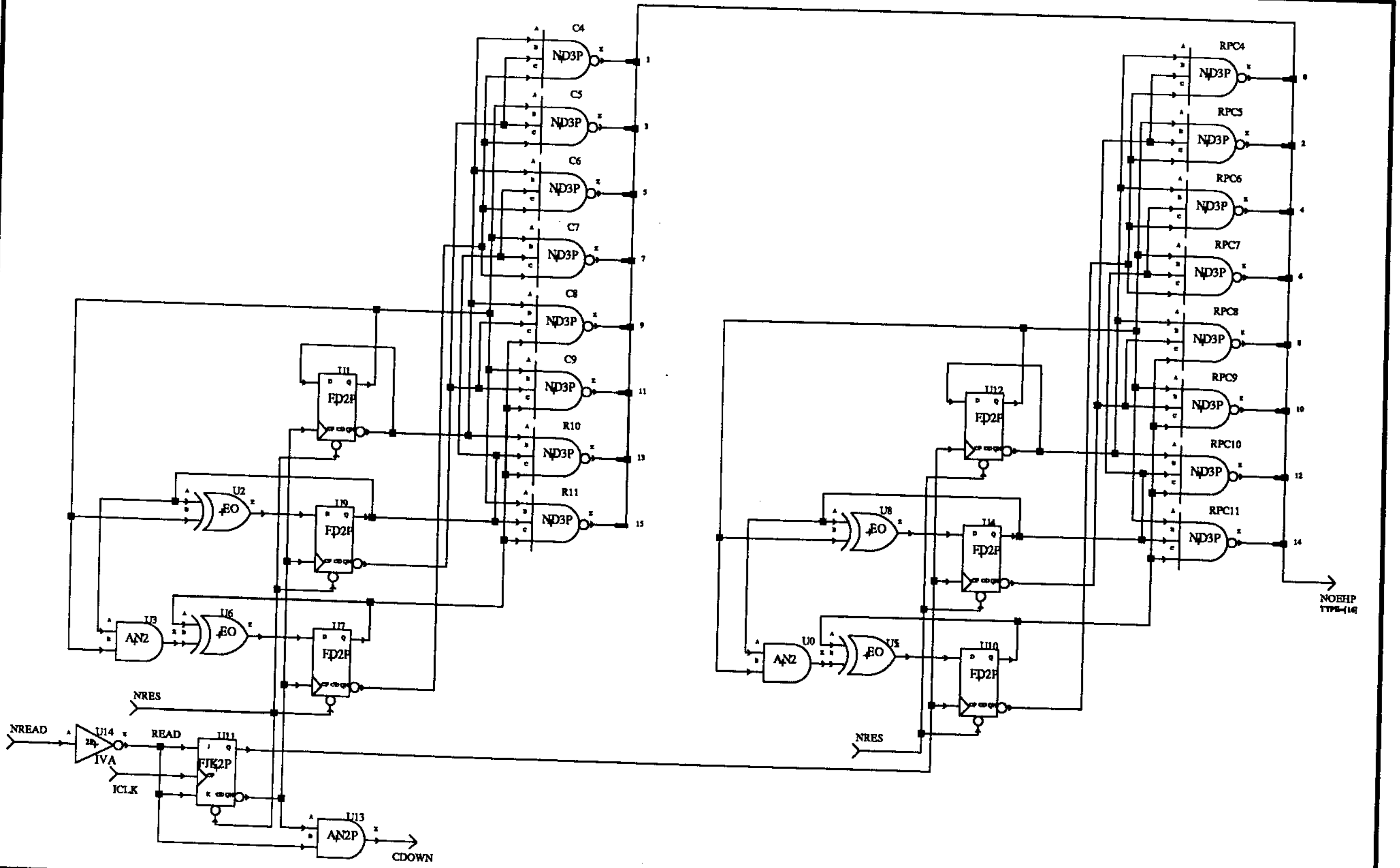
MODULE: PACMEM

DATE: AUG 29 1988

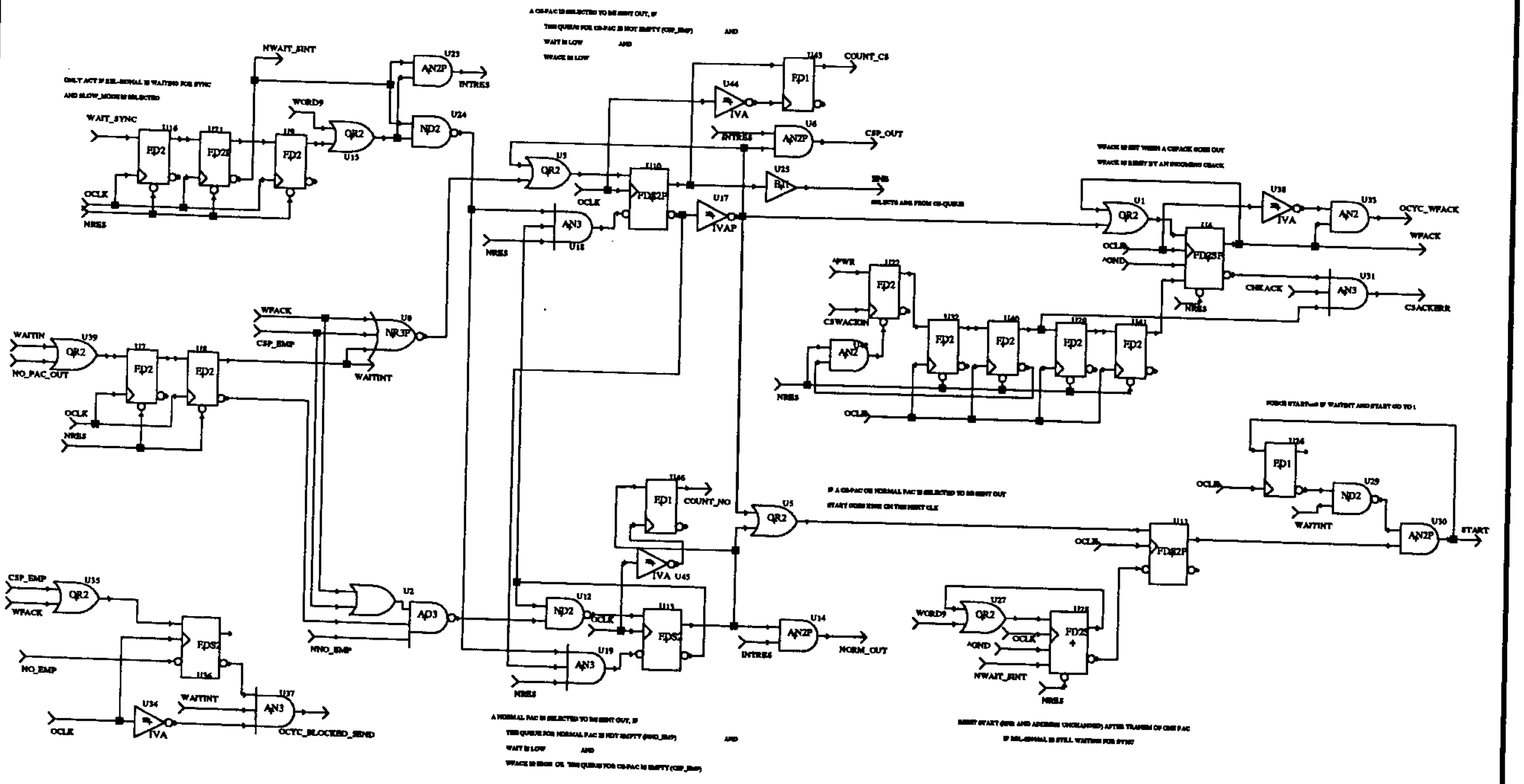
PAGE: 16 OF 45



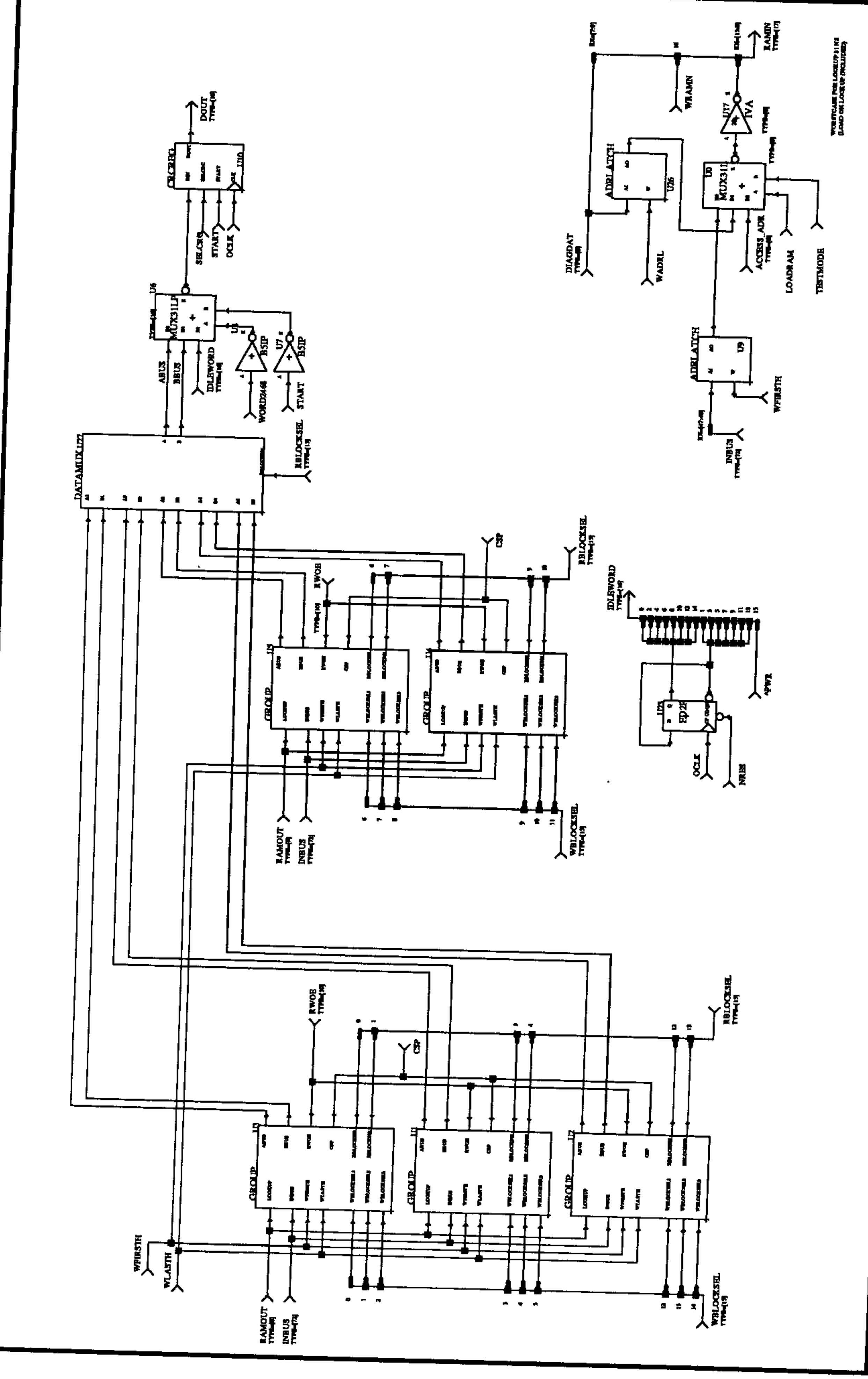
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: SYNC_XI
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 17 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: RHPSEL
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 18 OF 45


A CS-PAC IS SELECTED TO BE SENT OUT, IF
THE QUEUE FOR CS-PAC IS NOT EMPTY (CSP_EMP) AND
WAIT IS LOW AND
WFACK IS LOW
ONLY ACT IF SEL-SIGNAL IS WAITING FOR SYNC
AND SLOW_MODE IS SELECTED
WAIT_SYNC
NWAIT_SINT
WORD9
INTRES
OCLK
NRES
COUNT_CS
CSP_OUT
SNB
SELECTS ADS FROM CS-QUEUE
WFACK IS SET WHEN A CSPACK GOES OUT
WFACK IS RESET BY AN INCOMING CSACK
OCYC_WFACK
WFACK
CHKACK
CSACKERR
APWR
CSWACKIN
AGND
WAITIN
NO_PAC_OUT
CSP_EMP
WAITINT
FORCE START=0 IF WAITINT AND START GO TO 1
IF A CS-PAC OR NORMAL PAC IS SELECTED TO BE SENT OUT
START GOES HIGH ON THE NEXT CLK
COUNT_NO
START
NNO_EMP
NO_EMP
NORM_OUT
OCYC_BLOCKED_SEND
A NORMAL PAC IS SELECTED TO BE SENT OUT, IF
THE QUEUE FOR NORMAL PAC IS NOT EMPTY (NNO_EMP) AND
WAIT IS LOW AND
WFACK IS HIGH OR THE QUEUE FOR CS-PAC IS EMPTY (CSP_EMP)
RESET START (SEL AND ADDRESS UNCHANGED) AFTER TRANSM OF ONE PAC
IF SEL-SIGNAL IS STILL WAITING FOR SYNC

M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: OUTCTL
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 19 OF 45



M.I.T.

LL10075

DRAWN BY: FRANZ HUTNER

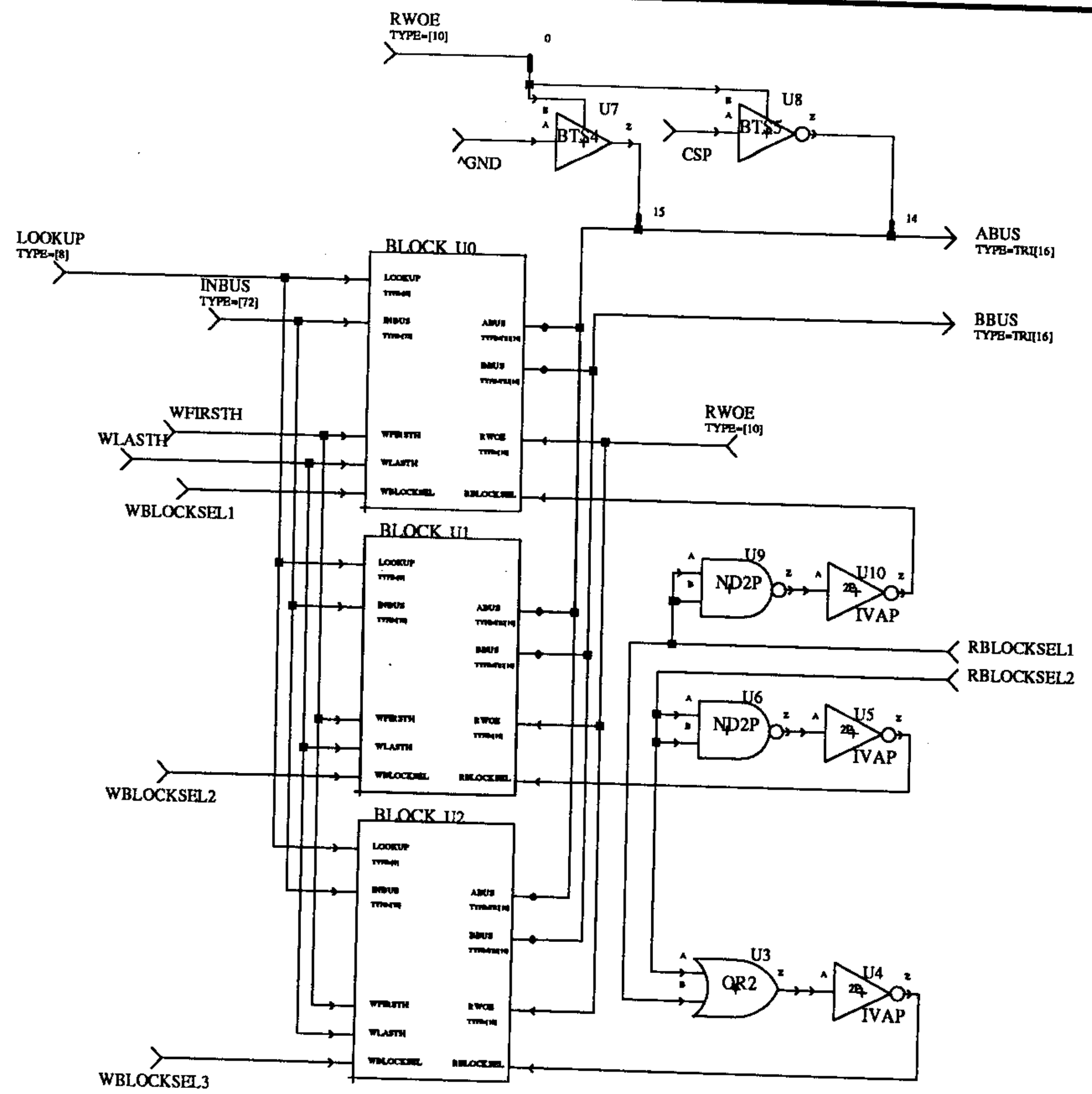
CHK'D:

TITLE: NIU

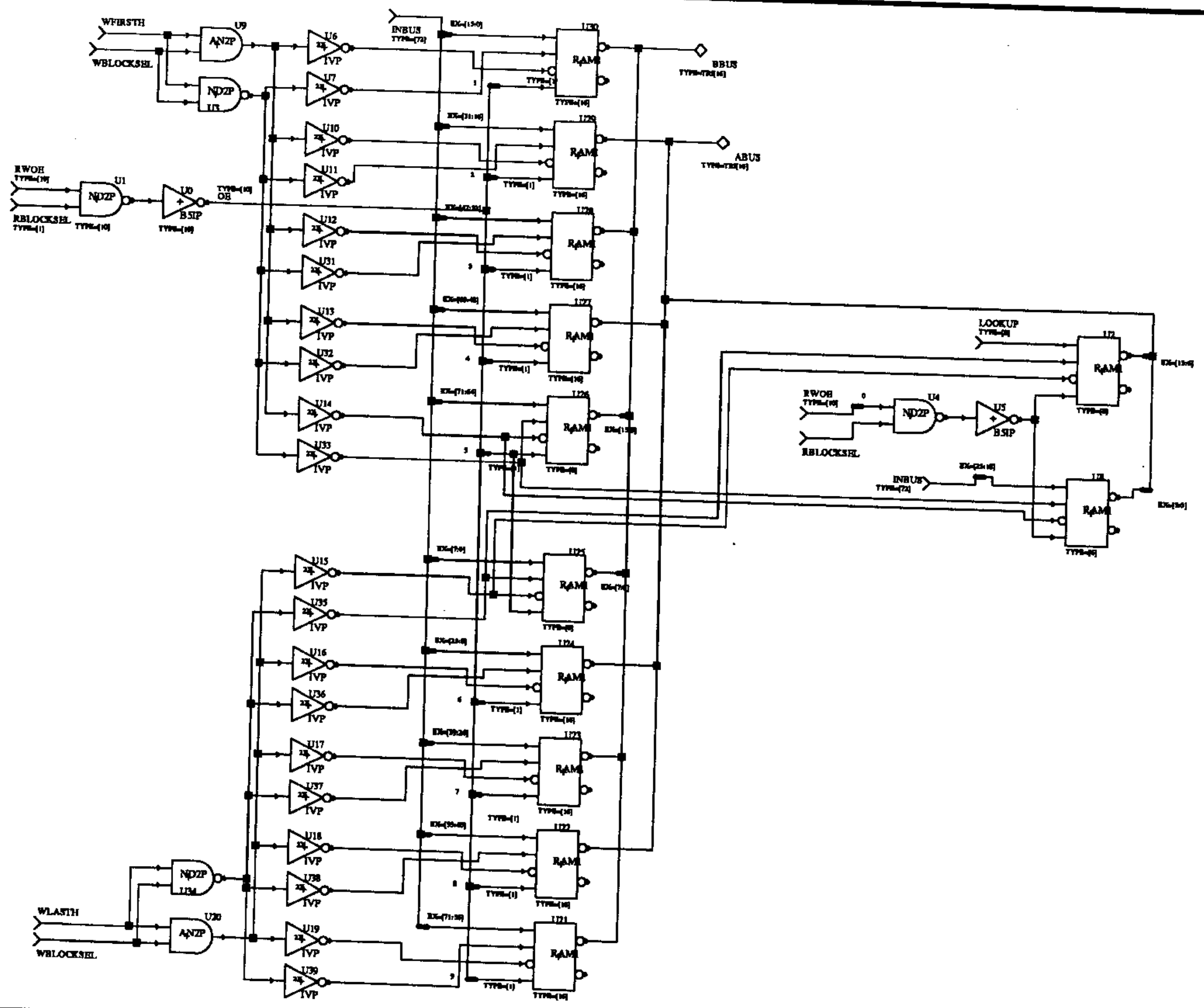
MODULE: O_MEM

DATE: AUG 29 1988

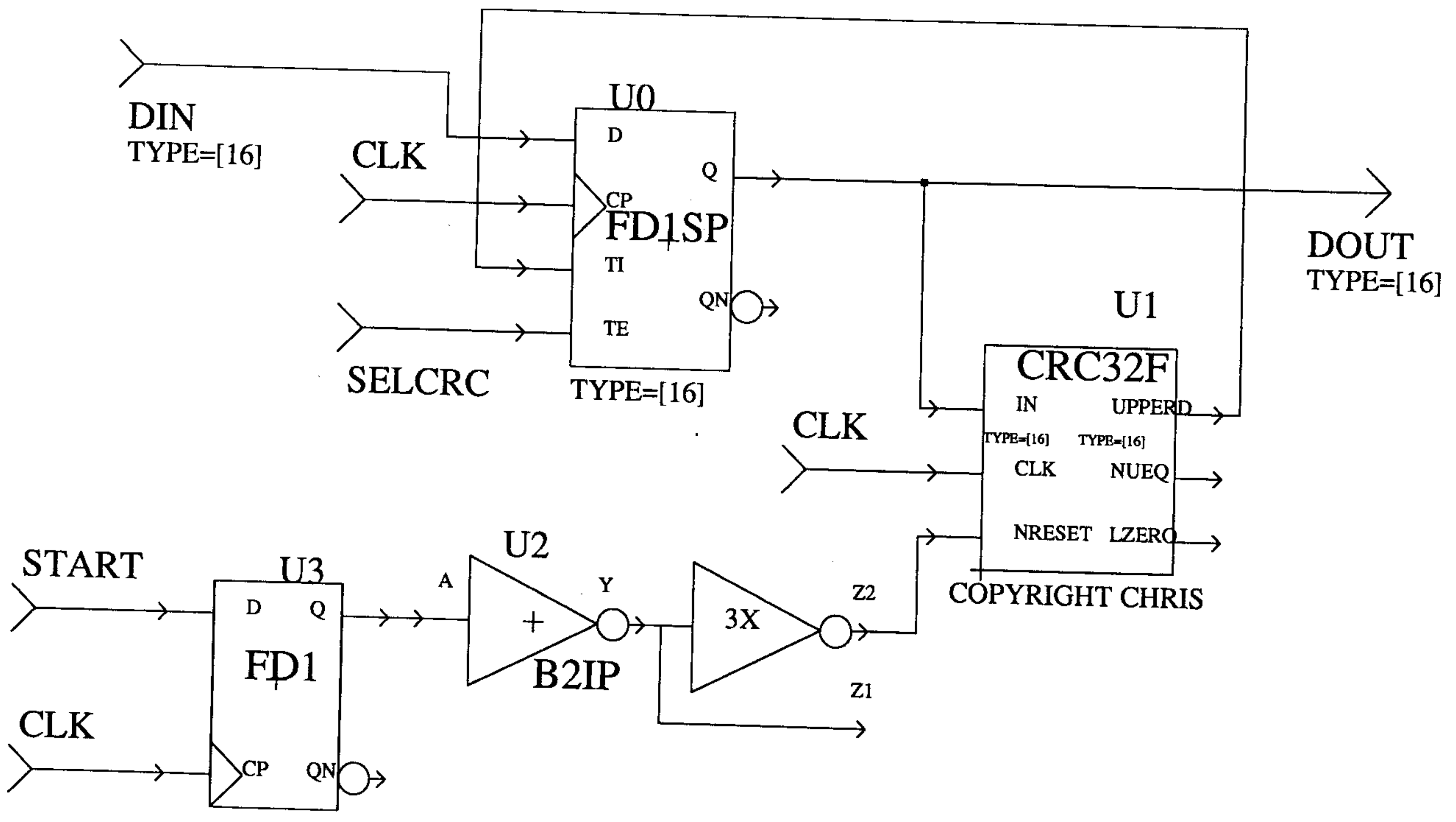
PAGE: 20 OF 45



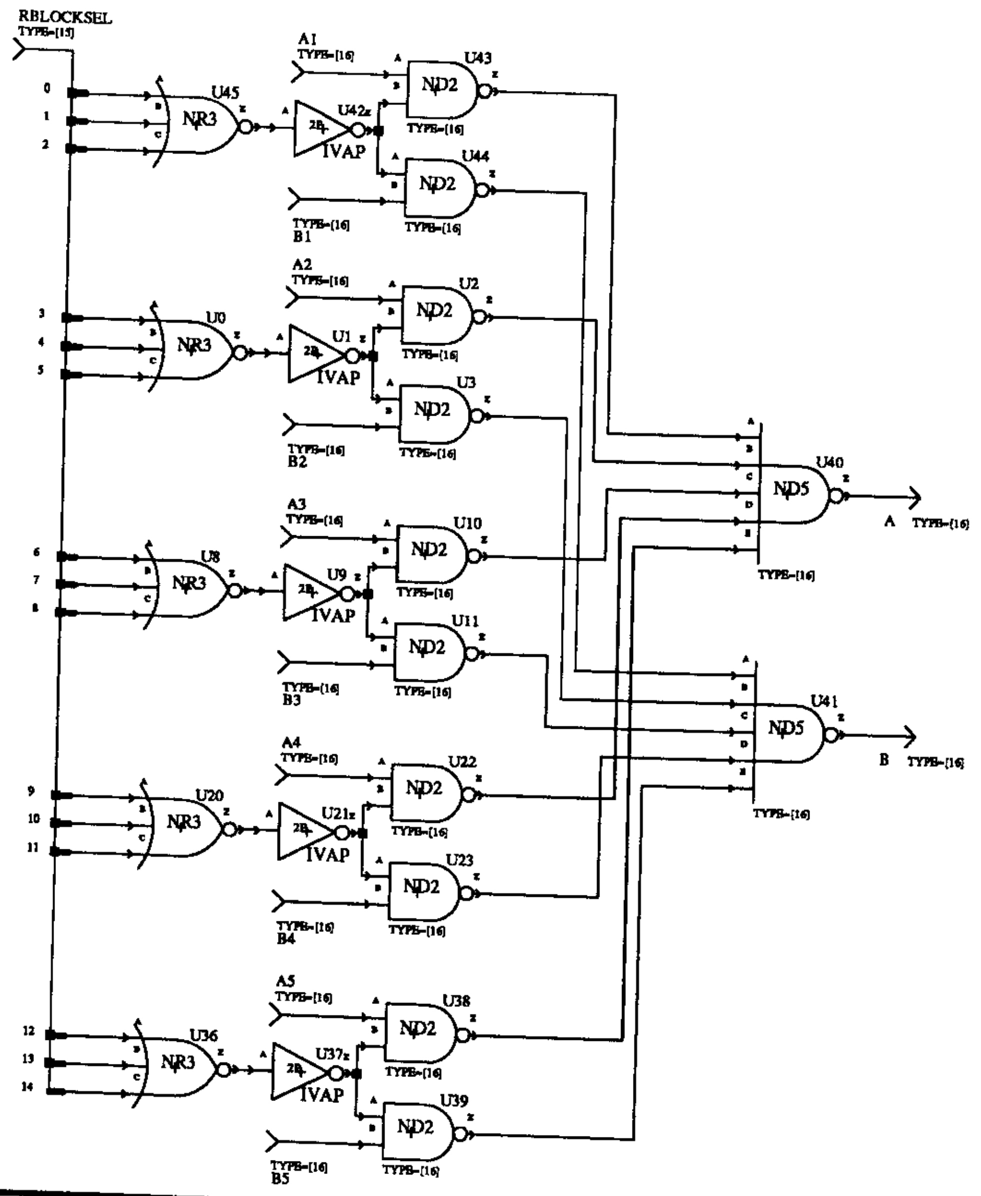
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: GROUP
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 21 OF 45



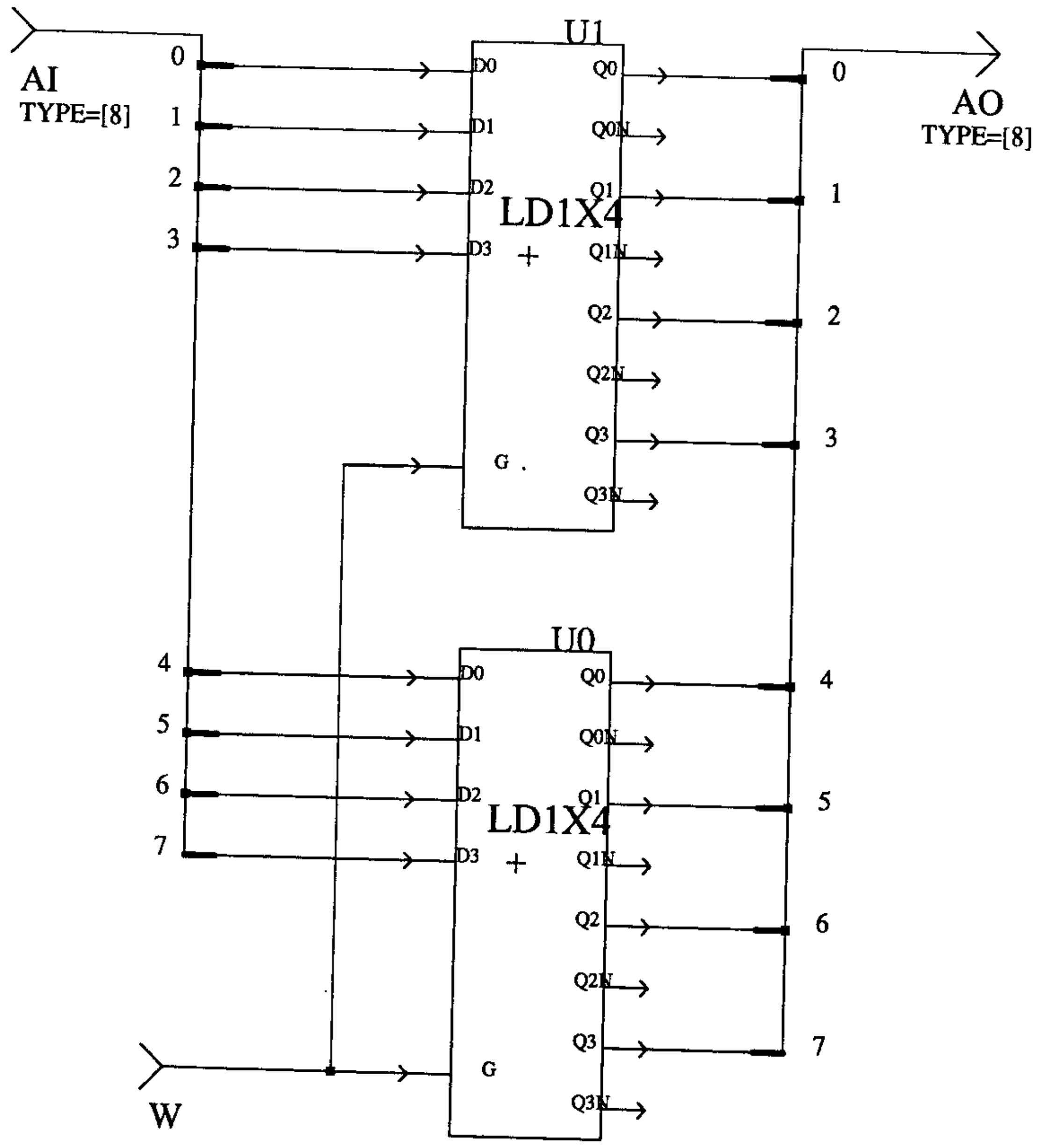
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: BLOCK
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 22 OF 45



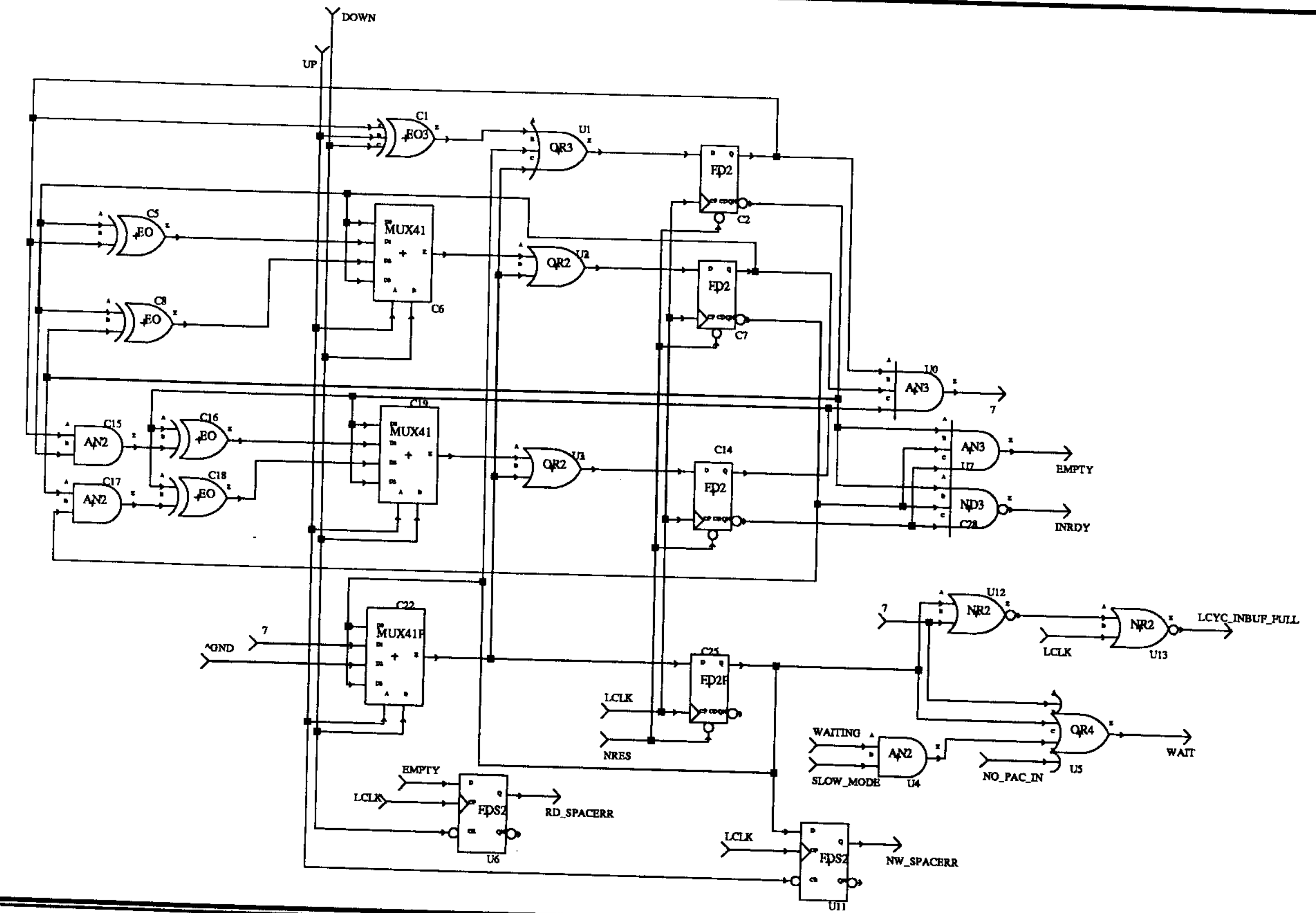
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: CRCREG
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 23 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: DATAMUX
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 24 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: ADRLATCH
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 25 OF 45



M.I.T.

LL10075

DRAWN BY: FRANZ HUTNER

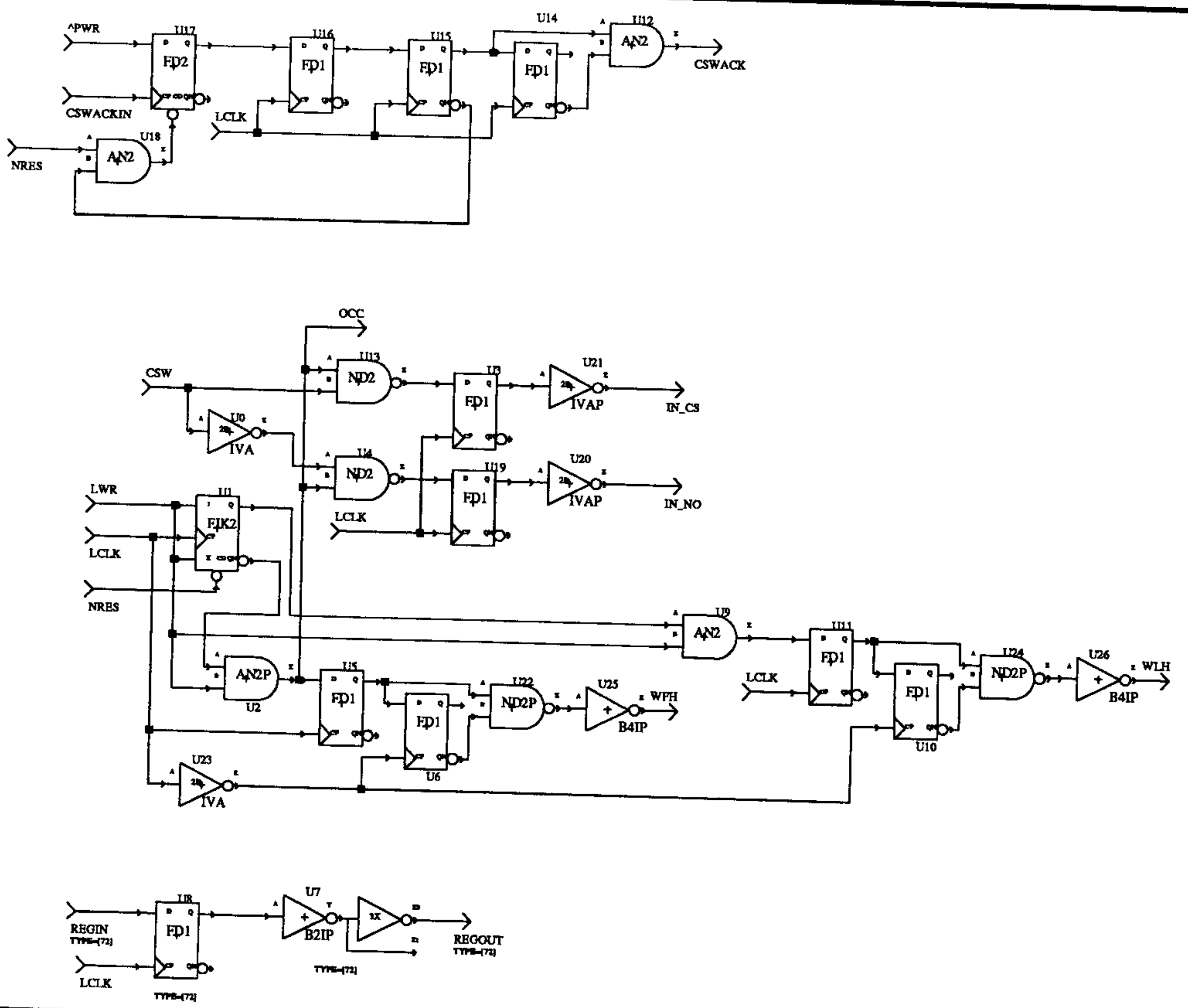
CHK'D:

TITLE: NIU

MODULE: GENWAIT

DATE: AUG 29 1988

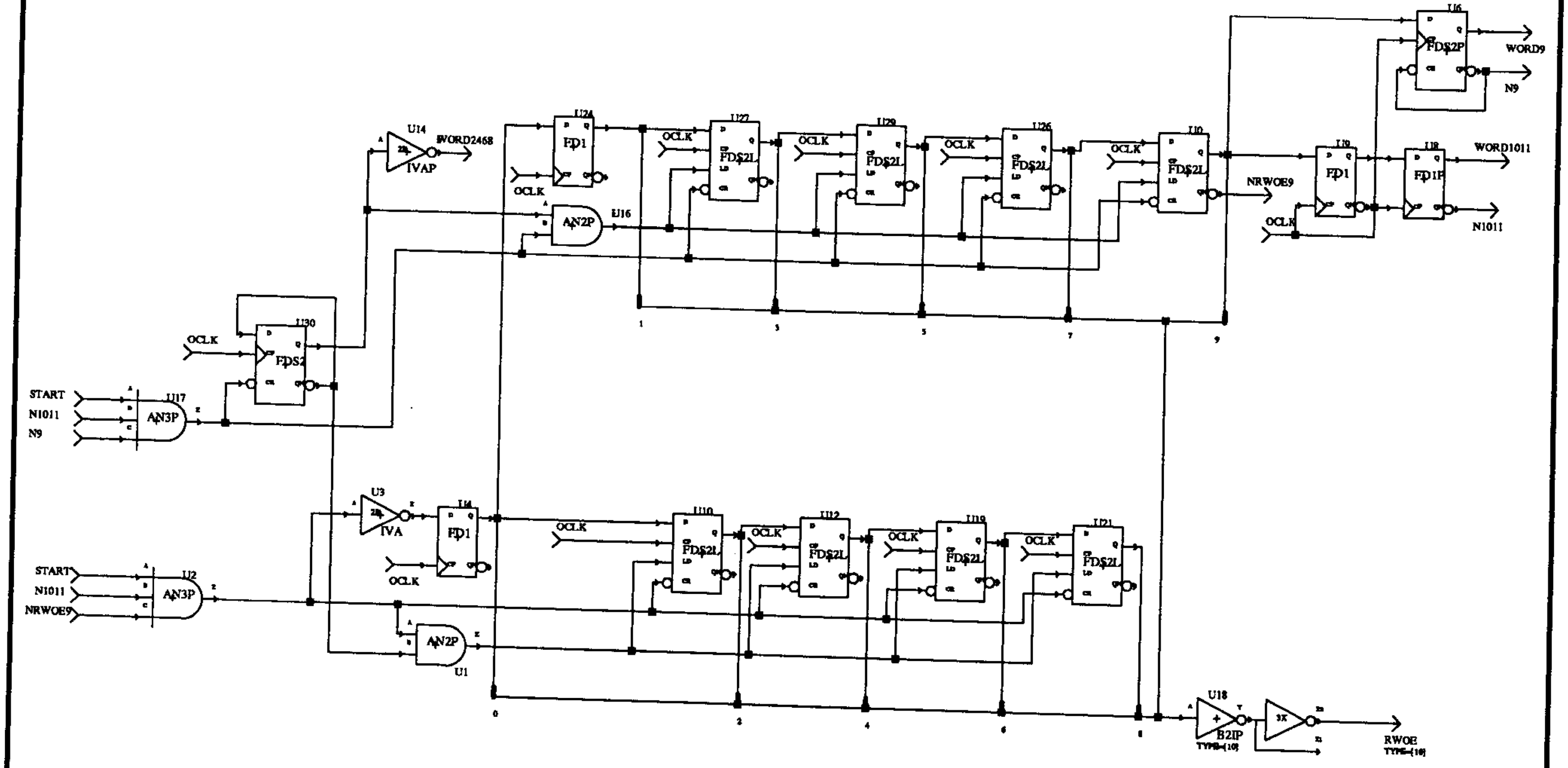
PAGE: 26 OF 45



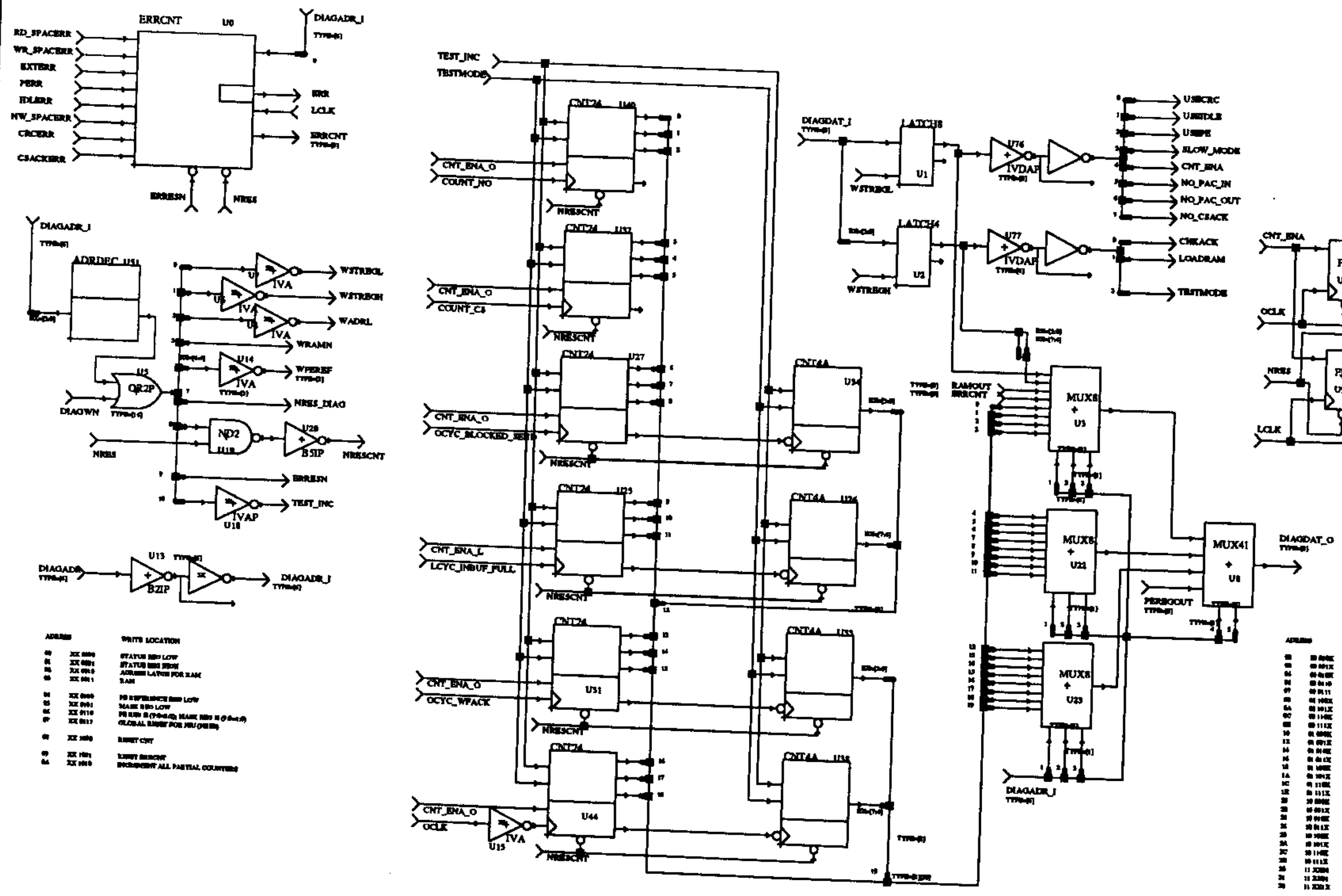
M.I.T.	DRAWN BY: FRANZ HUTNER
LL10075	CHK'D:

TITLE: NIU

MODULE: LBCTL
DATE: AUG 29 1988
PAGE: 27 OF 45



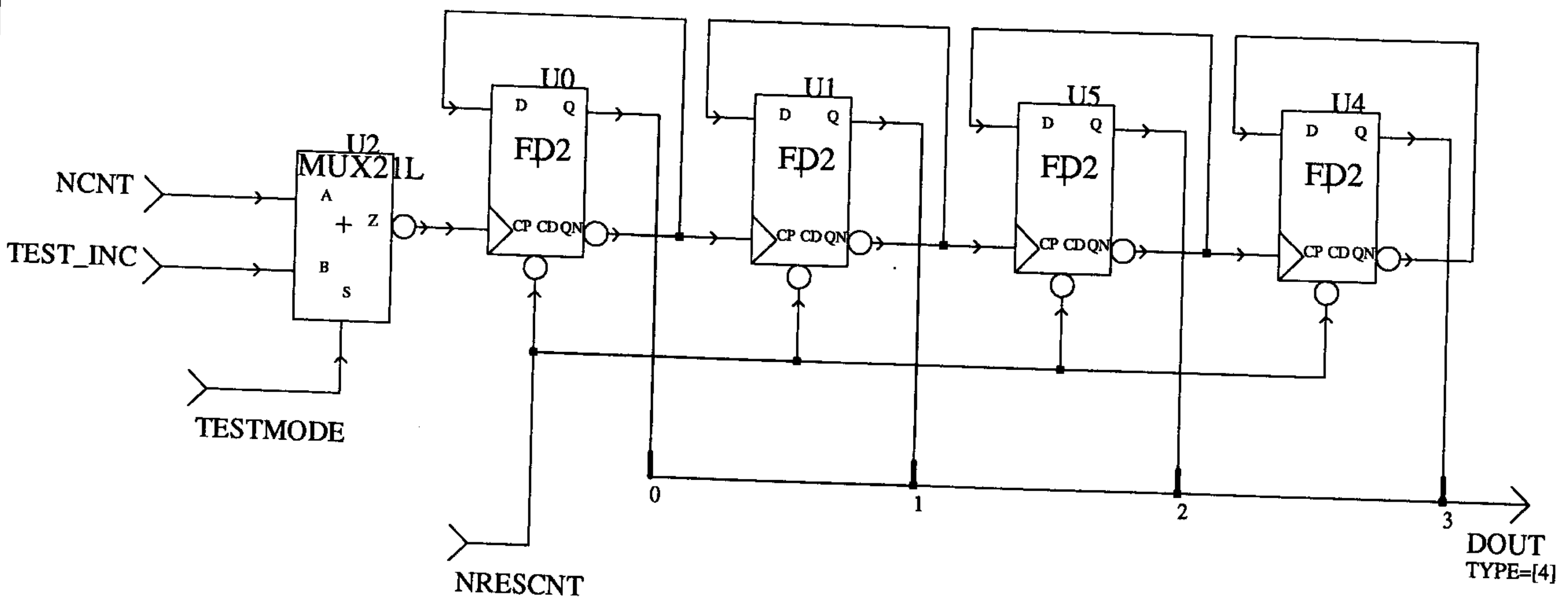
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: RWSEL
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 28 OF 45



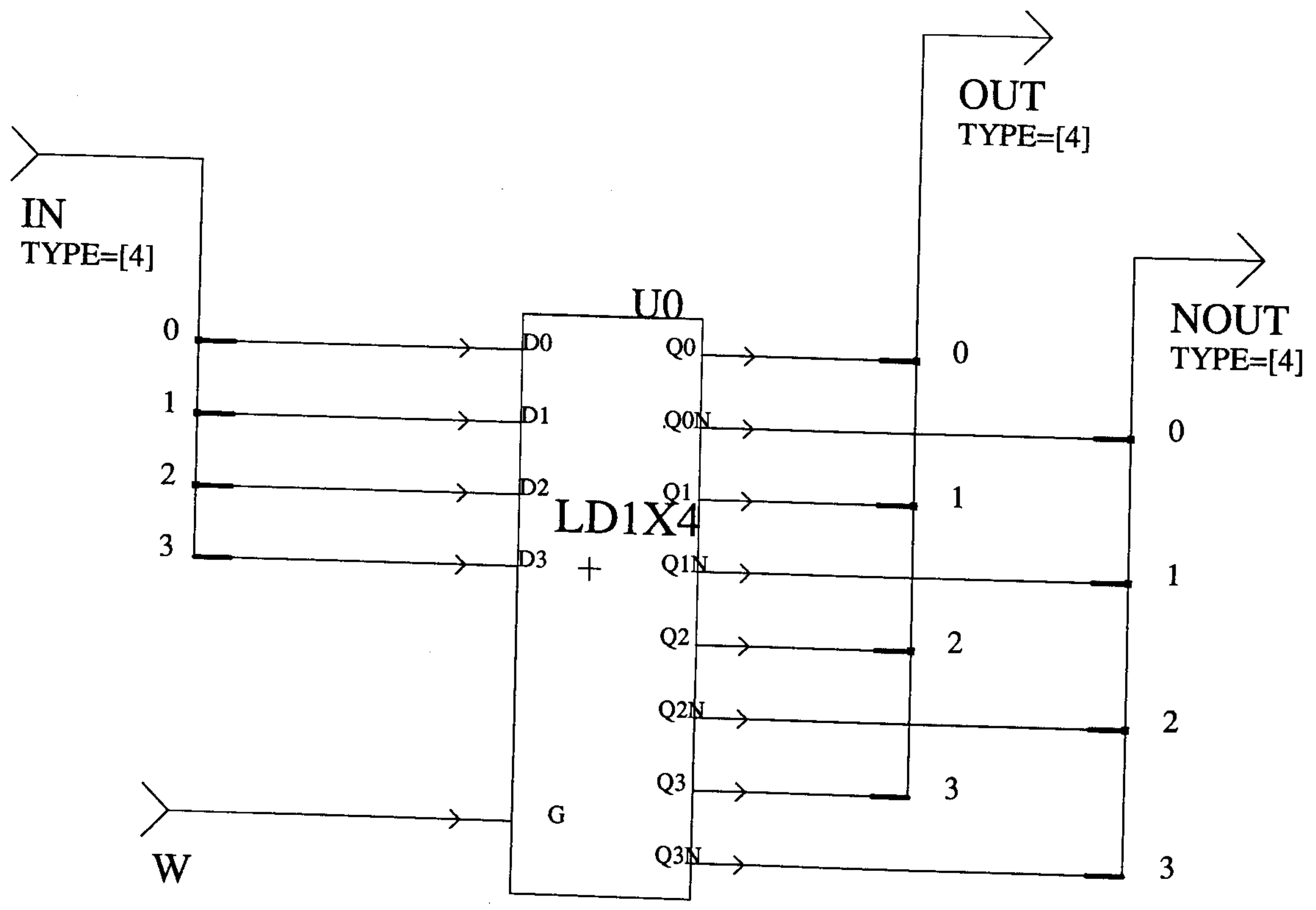
ADRESS		WRITE LOCATION
00	XX 0000	STATUS REG LOW
01	XX 0001	STATUS REG HIGH
02	XX 0010	ADDRESS LATCH FOR RAM
03	XX 0011	RAM
04	XX 0100	PS REFERENCE REG LOW
05	XX 0101	MASK REG LOW
06	XX 0110	PS REG H [illegible]; MASK REG H [illegible]
07	XX 0111	GLOBAL RESET FOR NIU (NRES)
08	XX 1000	RESET CNT
09	XX 1001	RESET ERRCNT
0A	XX 1010	INCREMENT ALL PARTIAL COUNTERS

ADRESS		READ LOCATION
00	00 000X	STATUS REG LOW
02	00 001X	STATUS REG HIGH
04	00 010X	RAM OUTPUT
06	00 0110	ERROR CNT LOW
07	00 0111	ERROR CNT HIGH
08	00 100X	COUNT_NO(7:0)
0A	00 101X	COUNT_NO(15:8)
0C	00 110X	COUNT_NO(23:16)
0E	00 111X	COUNT_CS(7:0)
10	01 000X	COUNT_CS(15:8)
12	01 001X	COUNT_CS(23:16)
14	01 010X	OCYC_BLOCKED_SEND(7:0)
16	01 011X	OCYC_BLOCKED_SEND(15:8)
18	01 100X	OCYC_BLOCKED_SEND(23:16)
1A	01 101X	LCYC_INBUF_FULL(7:0)
1C	01 110X	LCYC_INBUF_FULL(15:8)
1E	01 111X	LCYC_INBUF_FULL(23:16)
20	10 000X	LCYC_INBUF_FULL(27:24) ; OCYC_BLOCKED_SEND(27:24)
22	10 001X	OCYC_WFACK(7:0)
24	10 010X	OCYC_WFACK(15:8)
26	10 011X	OCYC_WFACK(23:16)
28	10 100X	OCLK(7:0)
2A	10 101X	OCLK(15:8)
2C	10 110X	OCLK(23:16)
2E	10 111X	OCLK(27:24) ; OCYC_WFACK(27:24)
30	11 X000	PS REFERENCE REG LOW
31	11 X001	MASK REG LOW
32	11 X01X	PS REG H [illegible]; MASK REG H [illegible]

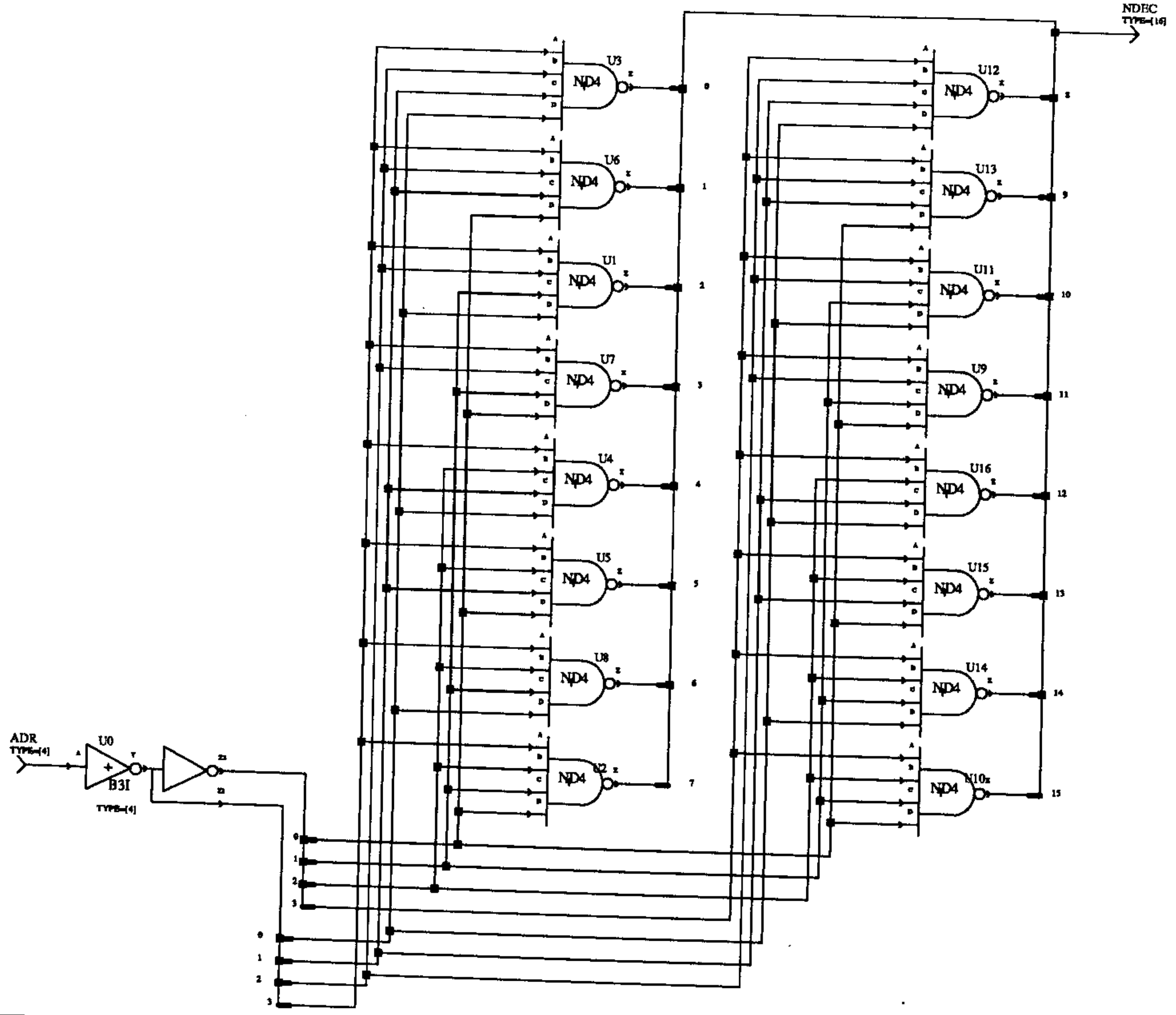
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: INSTRUME
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 29 OF 45



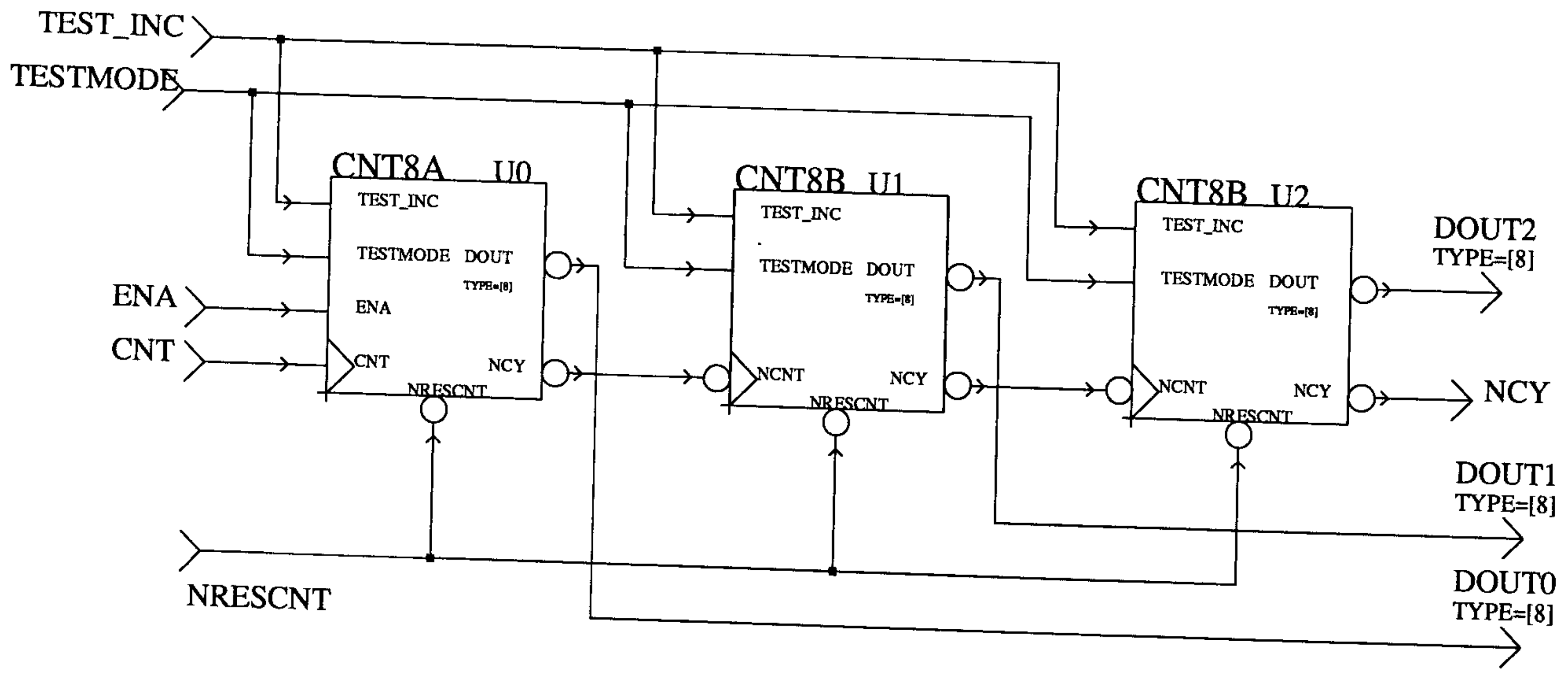
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: CNT4A
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 30 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: LATCH4
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 31 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: ADRDEC
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 32 OF 45



M.I.T.

LL10075

DRAWN BY: FRANZ HUTNER

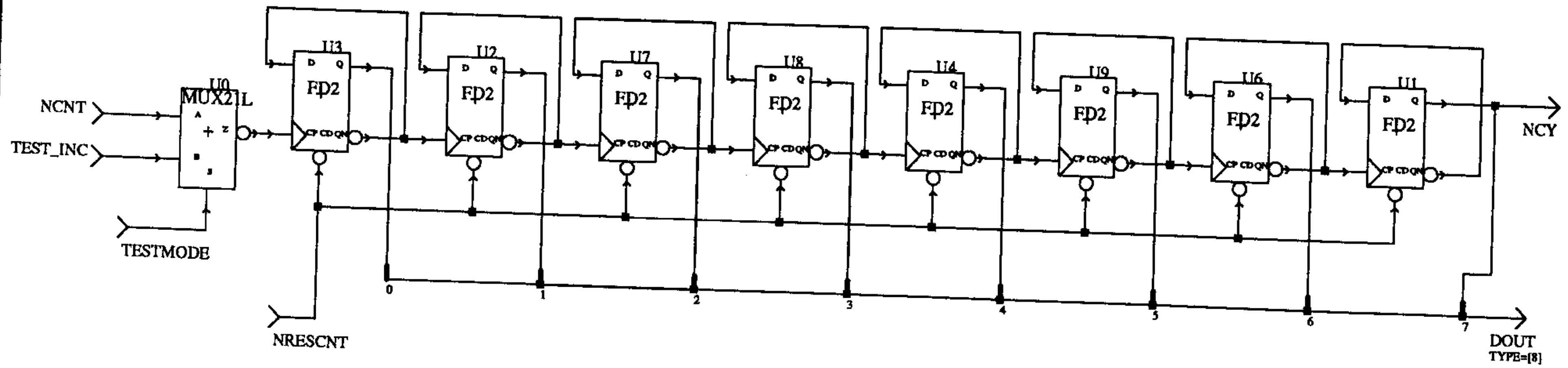
CHK'D:

TITLE: NIU

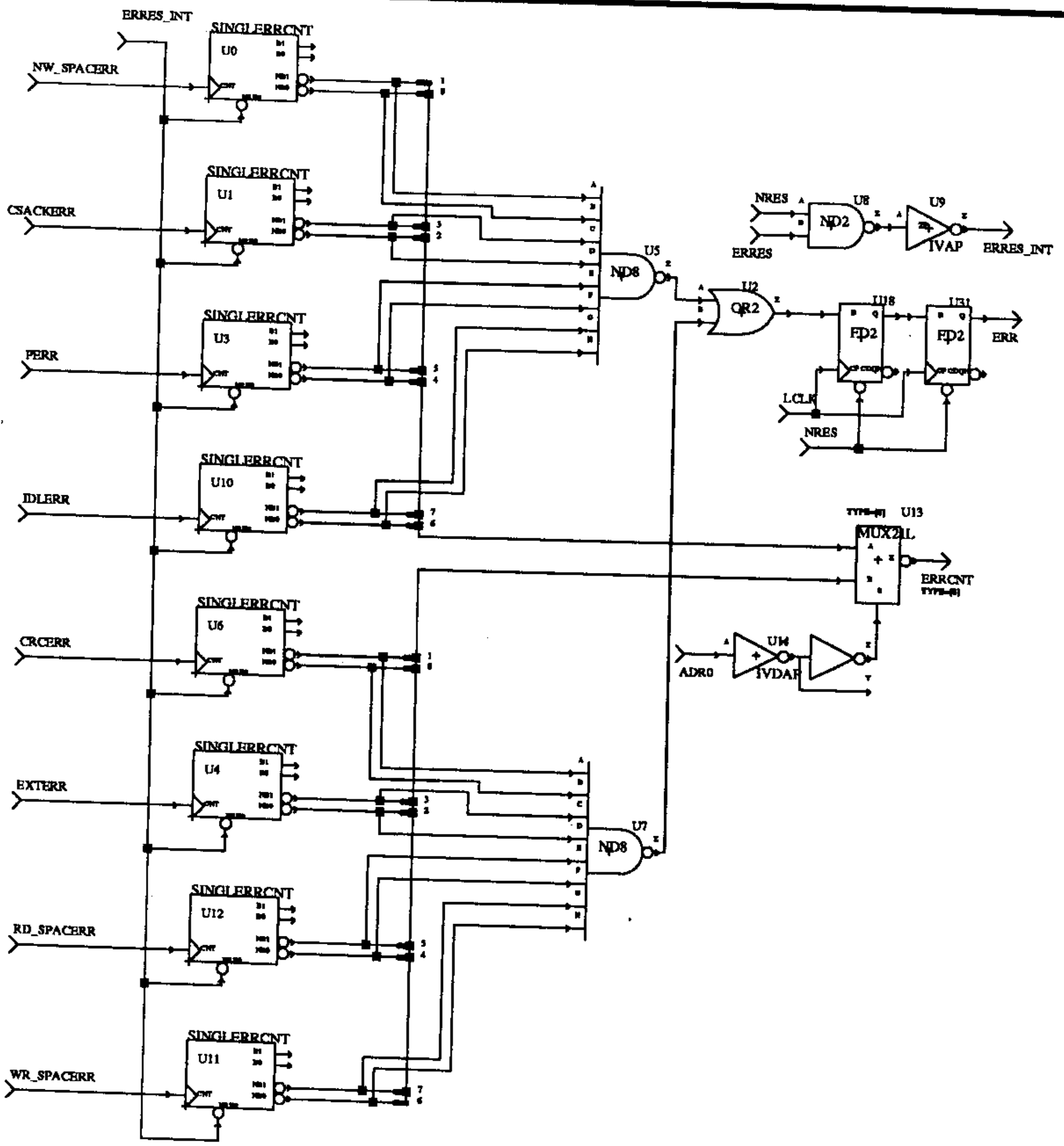
MODULE: CNT24

DATE: AUG 29 1988

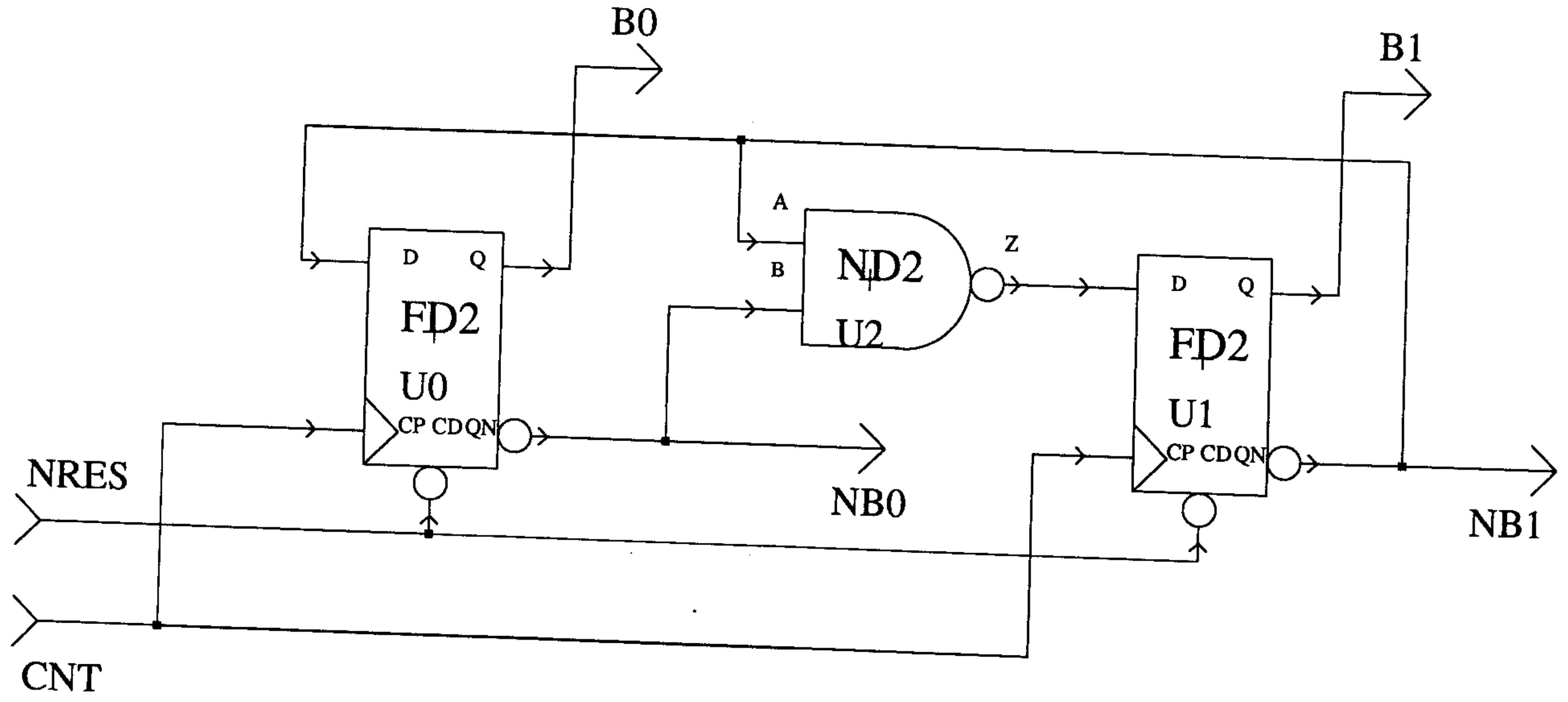
PAGE: 33 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: CNT8B
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 34 OF 45

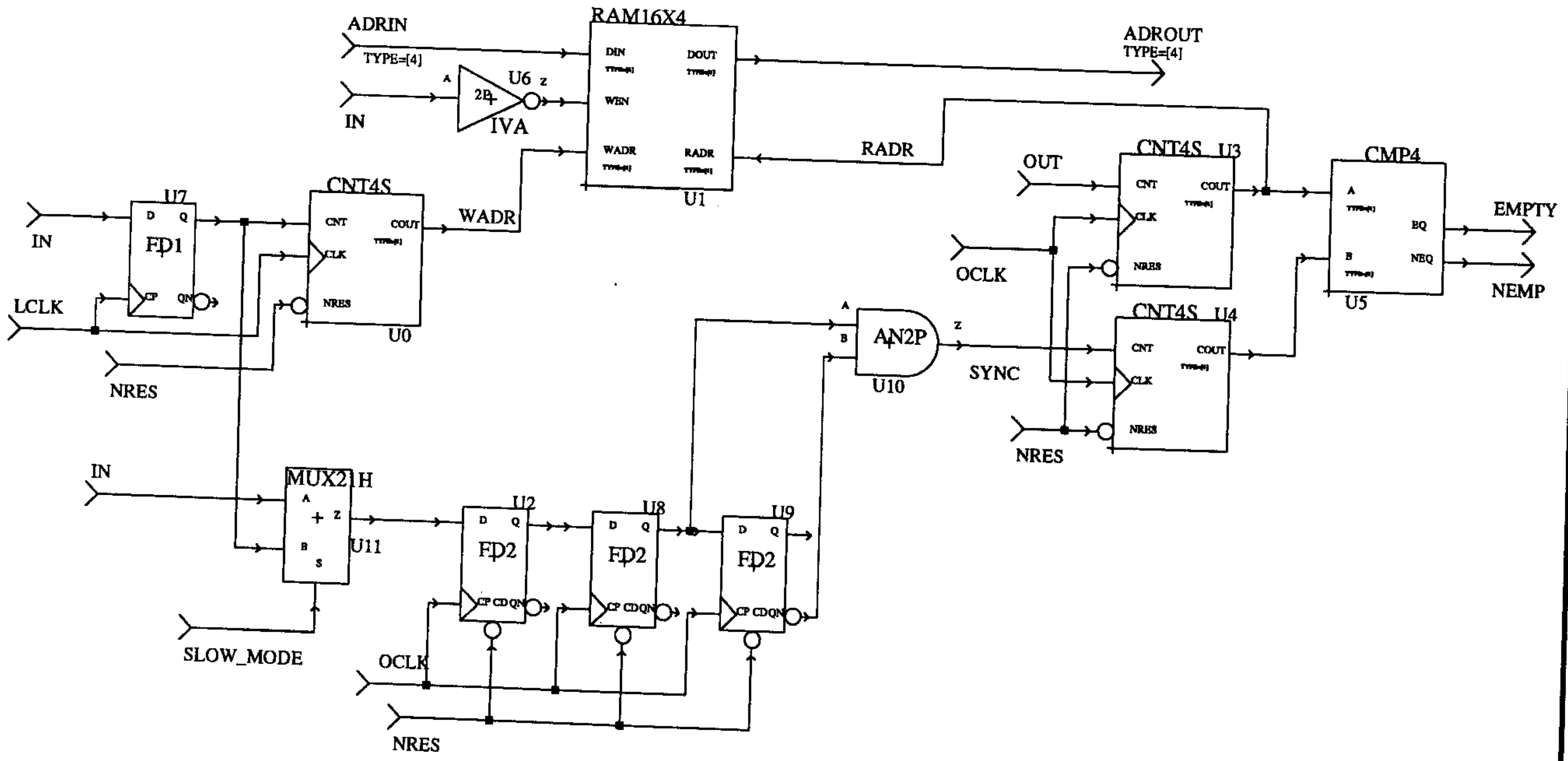


M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: ERRCNT
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 36 OF 45

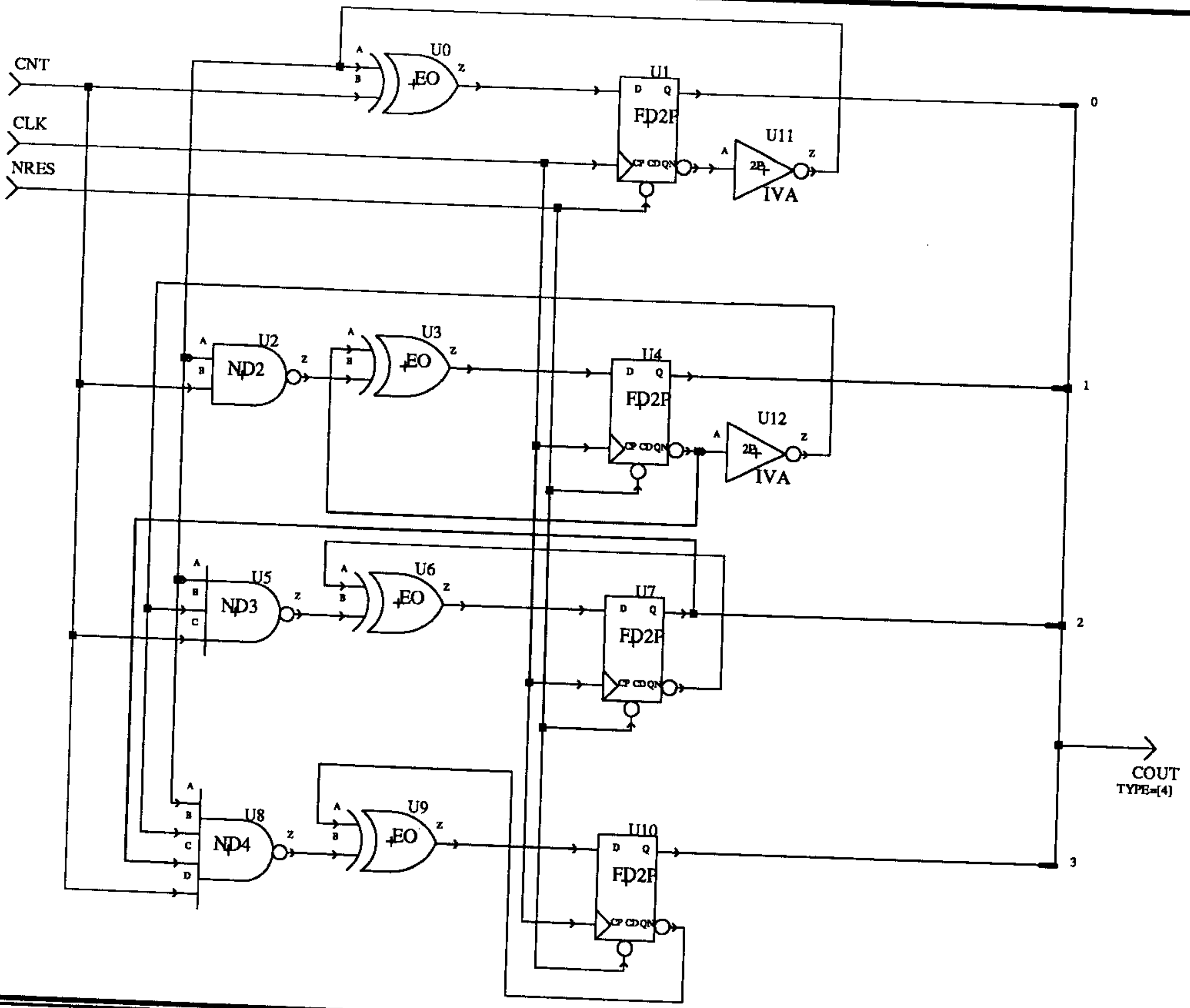


GRAY - CNT: 00-01-11-10-10-10-10...

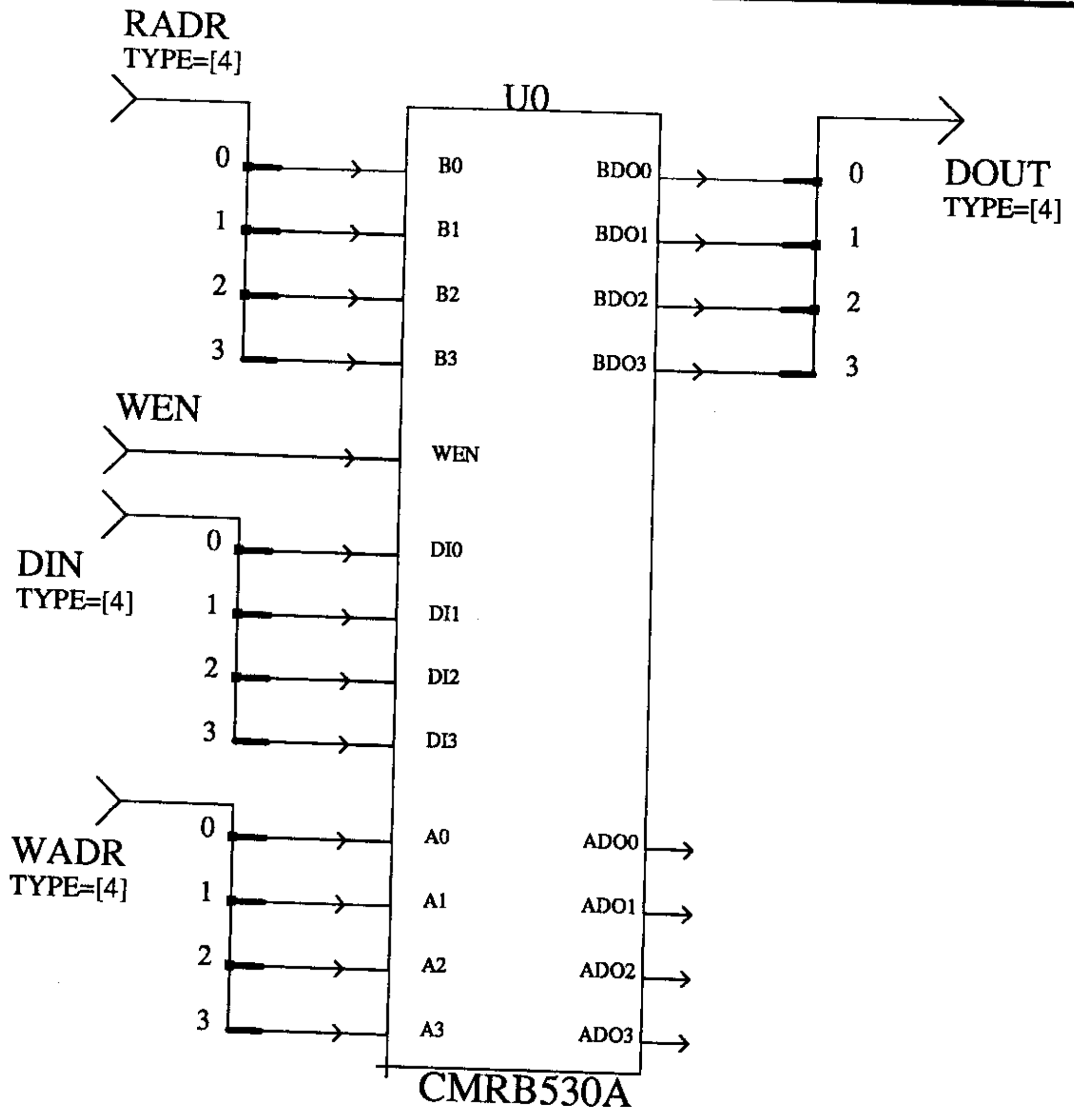
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: SINGLERR
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 37 OF 45



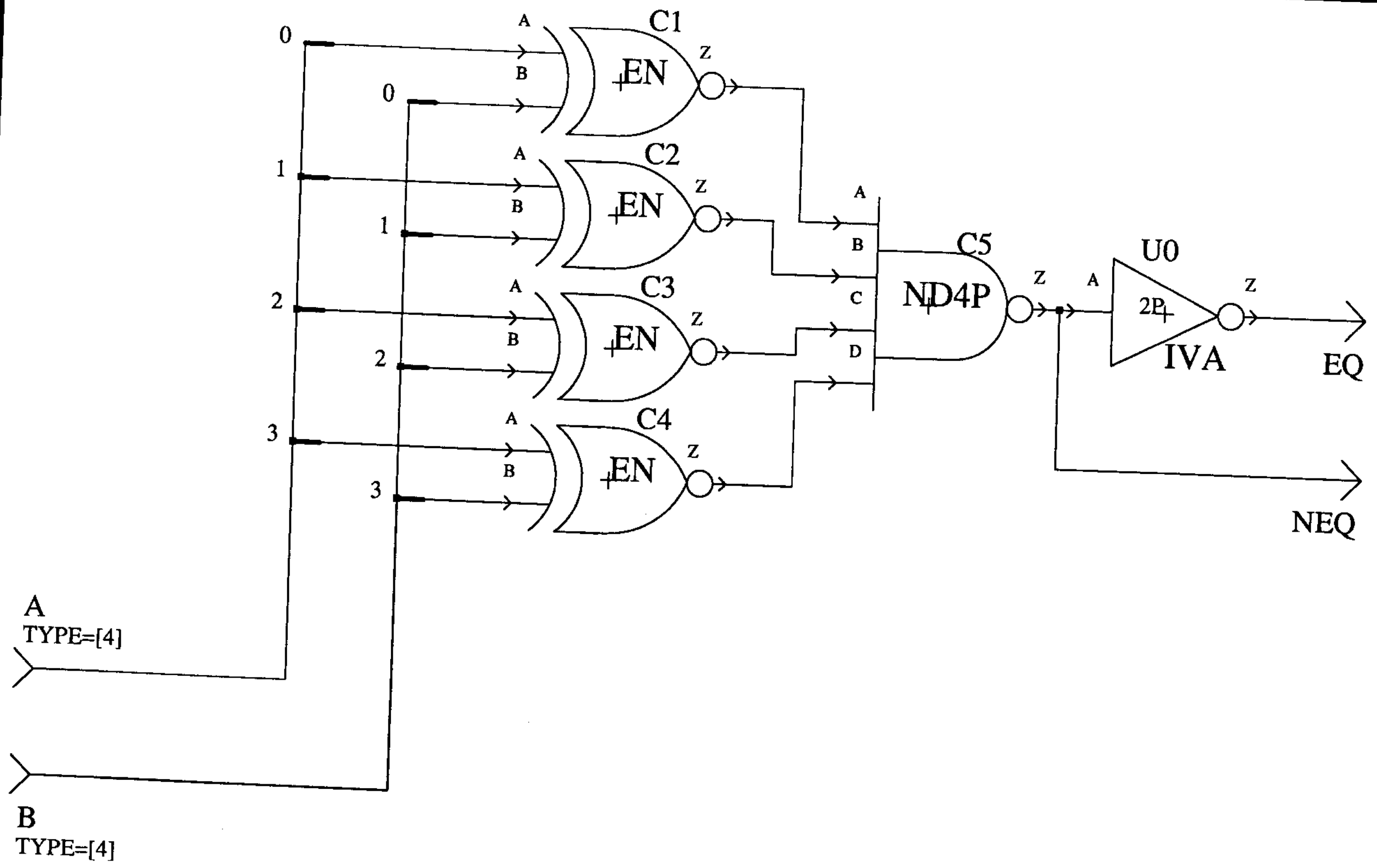
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: ADR_FIFO
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 38 OF 45



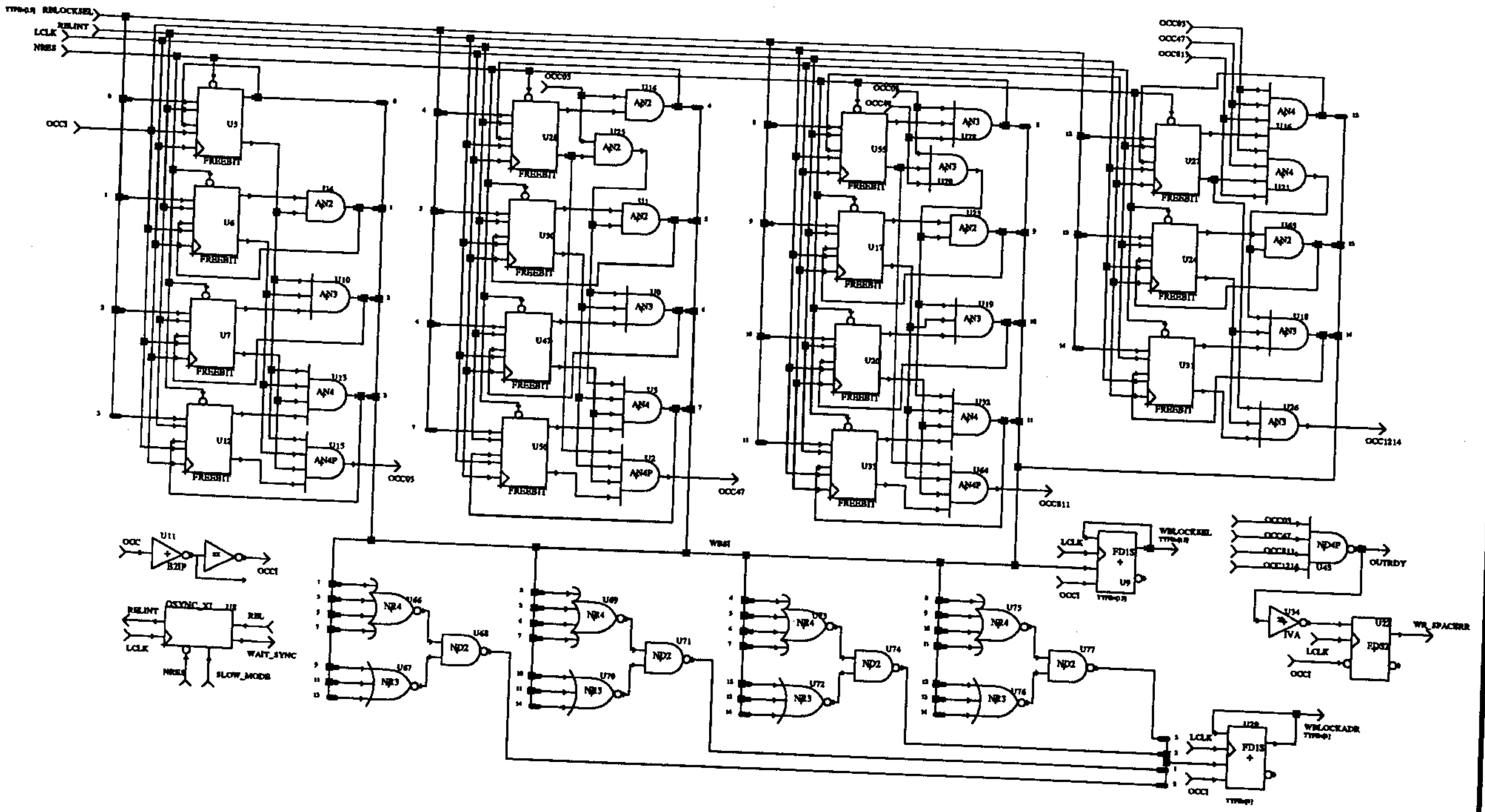
M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: CNT4S
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 39 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: RAM16X4
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 40 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: CMP4
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 41 OF 45



M.I.T.

LL10075

DRAWN BY: FRANZ HUTNER

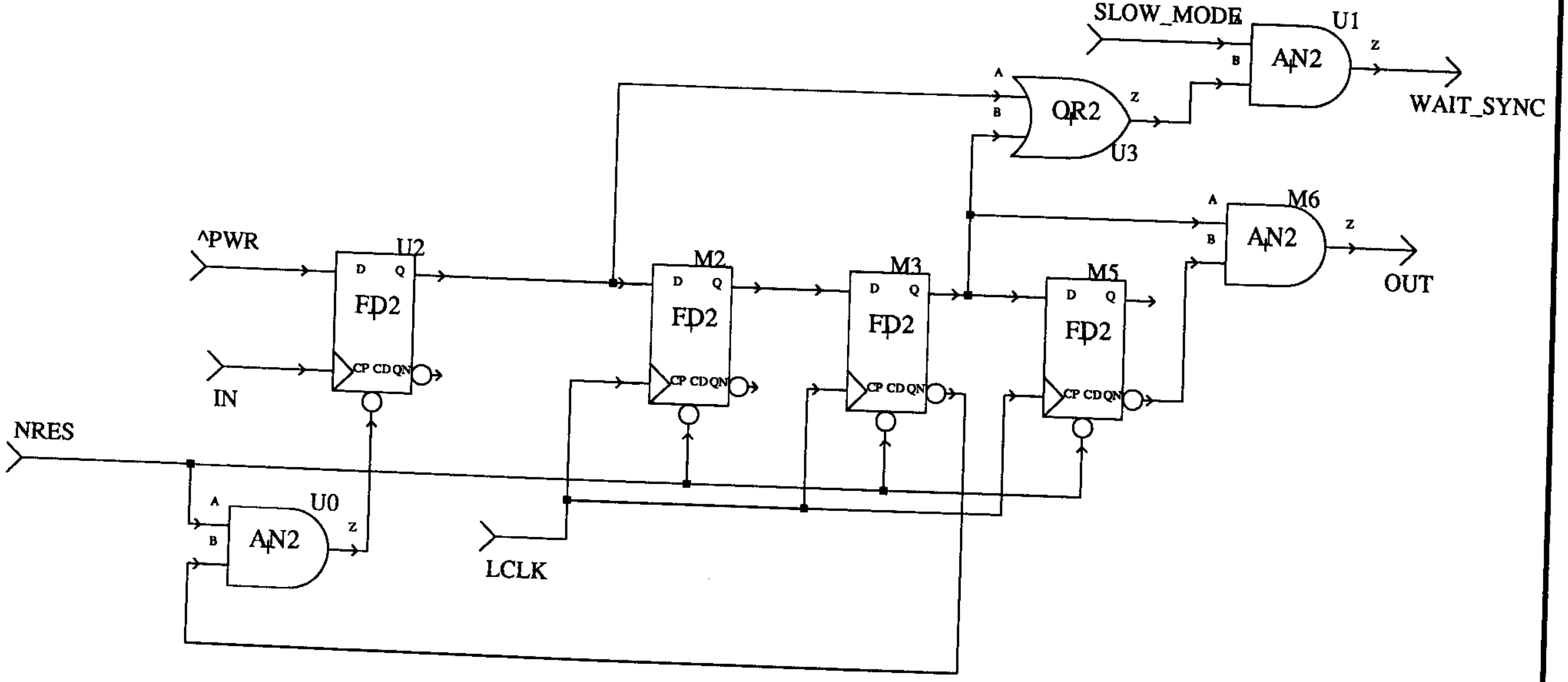
CHK'D:

TITLE: NIU

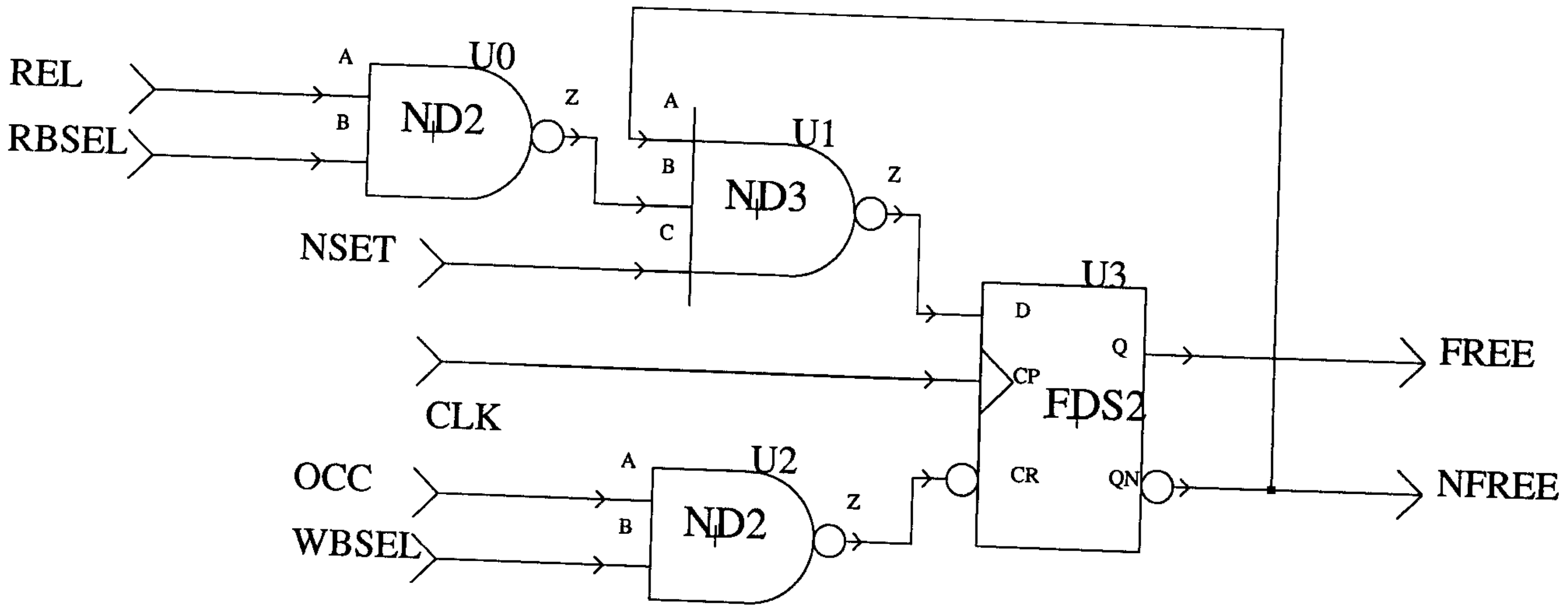
MODULE: FREE_BL

DATE: AUG 29 1988

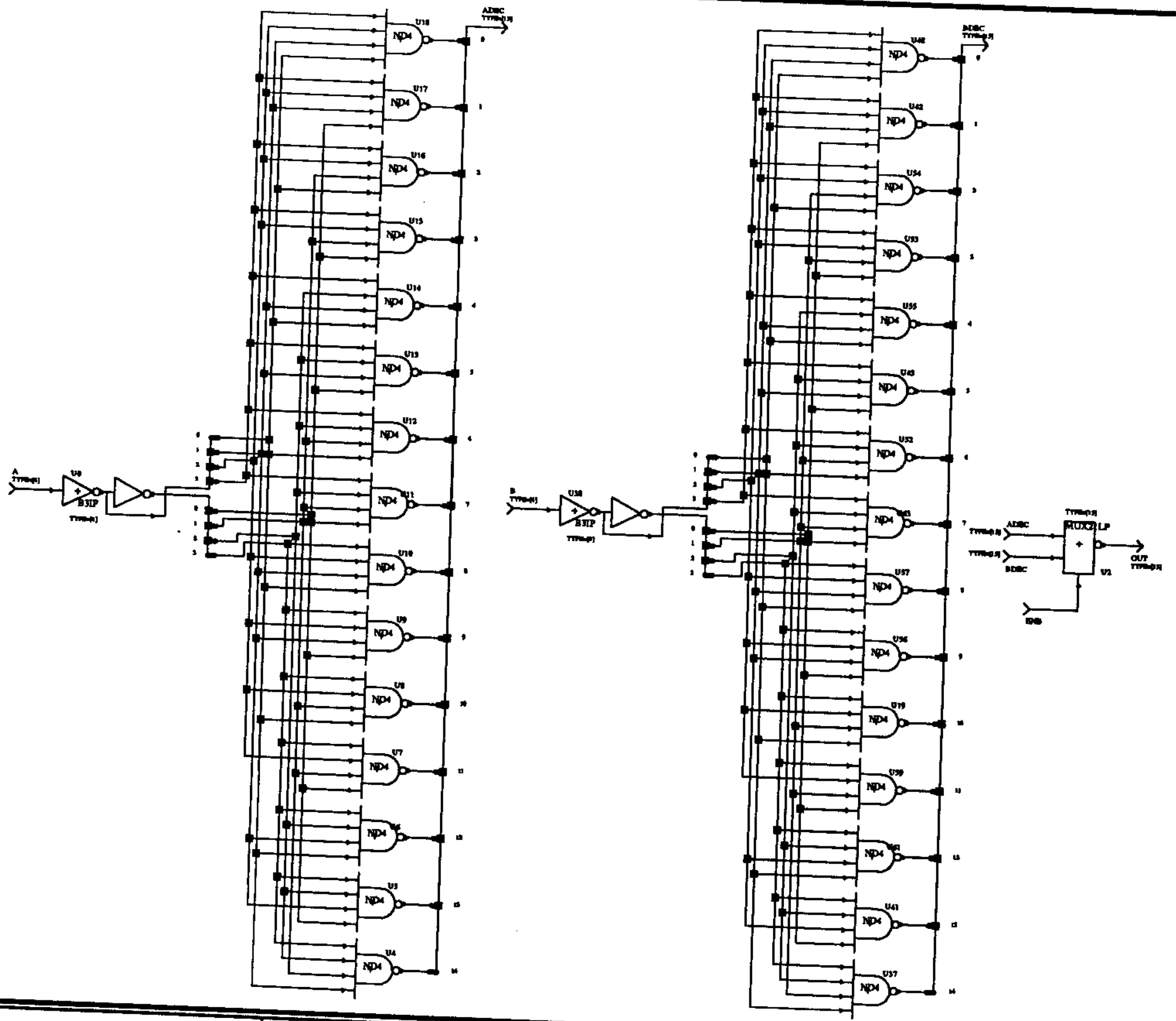
PAGE: 42 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: OSYNC_XI
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 43 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: FREEBIT
LL10075	CHK'D:		DATE: AUG 29 1988
			PAGE: 44 OF 45



M.I.T.

LL10075

DRAWN BY: FRANZ HUTNER

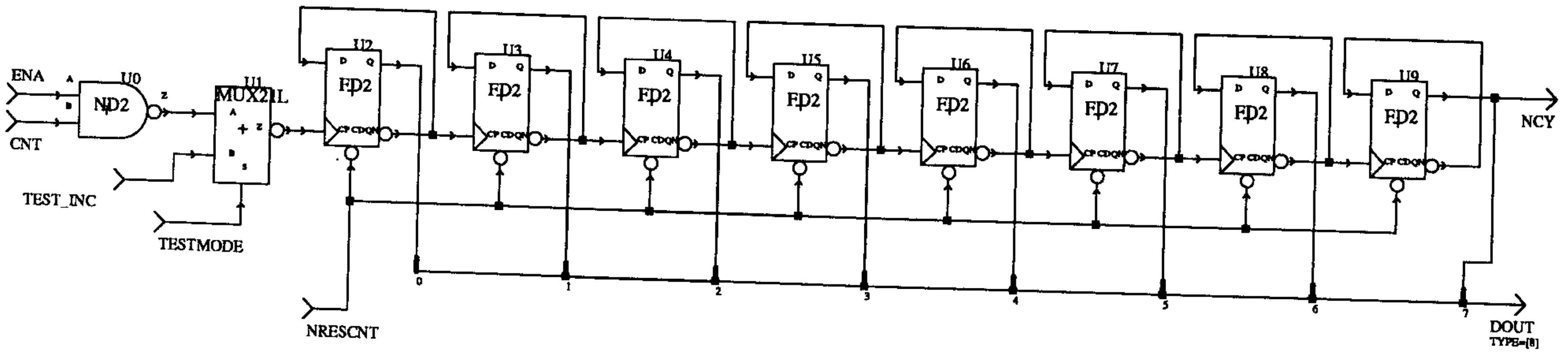
CHK'D:

TITLE: NIU

MODULE: MUXDEC

DATE: AUG 29 1988

PAGE: 45 OF 45



M.I.T.	DRAWN BY: FRANZ HUTNER	TITLE: NIU	MODULE: CNT8A
	CHK'D:		DATE: AUG 29 1988
LL10075			PAGE: 1 OF 1