

SLAC-TN-75-12
August 1975

CONVERSION OF AN XDS SIGMA-5 TO A PSEUDO SIGMA-7 OR SIGMA-9

by

Warren L. Graves, Anthony E. Gromme, Robert E. Melen

INTRODUCTION.

In order to run a Xerox Data Systems (XDS) Sigma 5 computer with the newest CP-5 software releases designed for Sigma 7 and-up machines, there must be either conversion of the software down to a Sigma 5 level or of the hardware up to a Sigma 7 level or a combination of both. The major software problem is the "memory mapping" and the major hardware difficulty is in the "byte string" instructions. Therefore implementation of Mapping in the hardware and Byte-string, Decimal and Convert Simulators in the software was undertaken. This report will cover the details involved in implementing the Hardware and Software modifications required to allow the CP-5 system to be used on the Sigma 5 computer.

INTRODUCTION TO THE HARDWARE MODIFICATIONS.

Mapping requires that a 256 word, 8 bit memory be placed between the central processing unit (CPU) address lines and the memory address lines. The theory of mapping or paging is well known and will not be dealt with here. As in most installations, only part of the address is mapped and is usually the high order bits of the address. Since there are certain restrictions as to when the CPU is allowed to map, the ability to map is controlled by the CPU itself.

The instruction that loads the Map memory register (the Map is a changeable memory) in the Sigma 7/9 is a variation of the same instruction in the Sigma 5 which is used to load the Write Lock (WL) memory which is used to protect memory from unauthorized write operations. The Sigma 7/9 also has additional memory protection called Access Protect (AC) which is loaded with another variation of Move to Memory Control (MMC) instruction. Therefore two additional versions of the MMC must be implemented in the Sigma 5.

The WL control on the Sigma 5 causes the main memory to abort the cycle if attempting an unauthorized write cycle. The CPU then traps (aborts the instruction and forces an error routine) to the proper location in memory for supervisory action. The AC control on the Sigma 7, 9 does essentially the same except the conditions of the abort are extended to read and instruction fetches also.

There are then four areas of hardware modification that will be described: Mapping the address lines; Move to Memory Control Instruction; Access protect; Write Locks; and Abort memory and trap.

The actual modification was done by replacing the four Write Lock memory cards in location 1N, 2N, 3N and 4N in the CPU with S.L.A.C. designed logic cards. The Write Lock memory was then condensed and reimplemented on part of the replacement cards. One additional card was used in 2B to collect terms in the second bay and transmit them to the 1N area. The initial design contains plugs which switch the configuration from the Sigma 5 to Sigma 7, 9 modes if desired during initial checkout. The modification also attempted to disturb the wiring of the CPU as little as possible. The wiring on 1N, 2N, 3N, and 4N contained many of the signals necessary and therefore using these locations was desirable.

MOVE TO MEMORY MONITOR (MMC).

There are four versions of the MMC instruction: Bits 12, 13 and 14 of the instruction distinguish between them. The Sigma 5 does not store any of these bits but simply traps if bits 12 or 13 are present at the time the instruction is fetched. The conversion prevents the trap by simply shorting the decoded trap signal to ground. Two register bits are required to save bits 12 and 14 of each instruction as they are fetched. If the instruction is eventually determined to be a MMC then these bits are decoded to load the proper memory block: Map; Extended address Map (Sigma 9); AC; or WL.

Flip-flops F1 and F2 on drawing #2 are used to save MMC12 and MMC14. OXC is true each time an instruction is brought from memory. C12 and C14 are data or instruction bits just received from memory. CL is the CPU general clocking term. Logic gate A1, A2, A3, A4, & A5, decode the MMC12 and MMC14 signals and select the proper memory enable signals for the four systems: NEWLM for the WL; NEACM for the AC; NEMM for the Map and NEGEM for the extended Map.

The memories are always addressed by the program counter register, P. Therefore, when loading, the address must be biased properly. The AC and WL is biased at P20 while the Maps are biased at P22. The exact details are described in the Sigma 5, 7, 9 reference manuals. Signal NPUC20 is the normal signal used by the Sigma 5 to increment the P register during an MMC instruction. This signal is altered in the

modification as shown on drawing #2 and #6. NPUC20 operates normally for AC and WL only. When the Maps are loaded, NPUC20 is disabled and incrementing P takes place at P22 with the modified NPUC22/1 term. Further, if the Extended Map is loaded, the incrementing at P22 takes place every other cycle since two memories must be loaded with the same address reference. Likewise the two memories must alternate being enabled to store the information. The term from the CPU, BC1 is a byte counter that is used by the MMC instruction to count the four bytes of each data word from memory that the WL memory will store. This alternating byte directs the incrementing of P and the enabling of NEGEM or NEMM only when the Extended Map is loaded in a Sigma 9 configuration.

ACCESS CONTROL AND WRITE LOCK CARD.

The AC and WL memory is located in slot 04N. See drawing #4. Data bits to this card are from the sum buss S0 to S7. Addressing for the AC memory is from the P register address lines P15 to P20. Address line P21 and P22 are added. Normally P21 and P22 would be used later (as in the existing WL system) to decode the AC bits, however the 82S16B memory chips are capable of being tied together and enabled individually as needed by address bits P21 and P22. OR gate O1 decodes these bits to select only two memory IC's at once during normal reading. However, during writing, all eight memories must be enabled. The term LOCKW enables all of the memory registers at the same time for this operation overriding the P21, P22 decoding logic.

The enabling term for the memories, NEACM from card 02N, combined with a clocking term NK/LK2, generated by the MMC instruction logic, causes the S data to be written into the memory.

The WL memory works similarly except that the bit decoding is done in existing logic with address bits P21 and P22 and therefore is not required on this card. The LOCKx terms are already wired to the decoding logic in the CPU.

The address bits for the WL memory are not the same as the AC memory. The WL memory is for the final, actual location (effective address) being used and therefore gets its address from the mapping memory output described later. If the Sigma 5 is not mapping then the AC and WL addressing will be the same.

MEMORY MAPPING MEMORY.

The map memory is located in slot 03N. See drawing #1. The P lines and S lines are already wired to this location. The P line output bits P15/1 to P22/1 are the modified or mapped address bits. These bits are from two sources; the unmodified P bits or the contents of the memory that the P lines address. Tri-state drivers (8T09) combine with the tri-state output of the 82S16B memories to form the resultant P address to the main memory and the WL memory that always works with the effective address. This card also contains the extended group address bits (2) that are not implemented. Also P15/1 can be interpreted differently in the Sigma 9 mode and therefore gate G1 is available for future use.

The write clocking terms NK/LKx for all the memories are generated on this card.

The enabling term for the map memory, NEMM, originates on card 02N. This term is used during the time the memory is being loaded and when the Sigma 5 is operating in the map mode.

ABORT.

The access control and the write lock (AC & WL) memories determine whether the CPU will abort the current main memory cycle and trap. The exact use of these signals is described in the technical manuals of the Sigma 7.

The Sigma 5 only has capabilities for WL abort and traps. Refer to diagram #7; Abort Timing Modification. The original ABO/2 signal was generated by the WL decoding and the Write Key in the PSD. The ABO/2 signal is broken and Ored with the abort signals from the AC control to form a new singel ABO/3 on card 01N.

It was also necessary to modify the timing signal ABO/1 as this signal was ANDed with ABO/2 to form the final signals, /ABOC/ and ABO. Also timing latch signal ABO/1 is therefore broken and replaced with ABO/1/1. The signal which set the abort timing latch is broken to eliminate the write signal NMBXS etc., (OR gate 01 on diagram #7), from the remaining signals (PCPACT, IOACT, etc.) since the AC aborts can occur independent of a write cycle. The new signal added is S/ABO/1/1.

ABO/3 LOGIC.

Refer to drawing #1 of the card in location 01N and diagram #7. The ABO/3 signal is a combination of four signals at OR gate 04. The signal from A4 is essentially the original ABO/2 from the WL logic. The signal on A20 disables all abort signals from the AC logic if the CPU is:

Not in map mode. A22.
Not in slave or master protected mode.
(MA&NMASTER), A21.
Addressing first sixteen locations.
(NCROSSADD), A20.

The other three aborts are controlled by NAC0 and NAC1 from the AC memory. These bits are read out using the high order eight bits of the P register address to main memory before any mapping takes place. The combinations of these bits for access protect are in the reference manual for the Sigma 7.

MRQ is equal to Memory request, S/ABO/1/1 is equal to a Write into memory. Term ACCTIM is equivalent to an instruction access and will be described later along with signal NINABO.

REGISTER CHANGE INHIBIT.

During certain instructions when an abort is about to take place, one of the general registers may be modified when the instruction format specifies that it should contain specific information. To prevent this A23 on 01N and G1 on 03N were added to the RW function in the CPU. RW is set when a register reference is being made. The Abort signal now will prevent the setting of RW and, if it is already set, inhibit the gating of the register control terms NRW-1 and NRW-2.

MAP DISCONNECT.

During certain instructions the map mode must be suspended temporarily. F-F MAPDIS inhibits the MAP signal for these periods of time. If an interrupt or trap occurs the CPU must refer to a specific location in main memory and not be mapped. When INTRAP F-F is set MAPDIS is also set. MAPDIS will remain set until INTRAP is reset - (R/INTRAP)-, at (FAPSD.PH4) or earlier if the instruction is an XPSD and the combinations of bit 10 of the XPSD and bit 9 of the current PSD so designate. See the Sigma 7 reference manual for details.

One exception to the above is the interval timer located in memory location 55. It is specified that this interrupt location will be mapped if Map mode is active. The only instruction that should be in this location is a Modify and Test. MAPDIS is therefore reset by gate A24 during the prep phase 1 of instruction (FAMT) if location 55 is being accessed by the P register.

MODE ALTERED.

The Sigma 9 has an added feature in that the PSD can contain a bit to force an access protect trap even when the CPU is in the master mode. This bit is stored in the second half of the PSD and labeled MA. F3 was added on card 02N(drawing #1) for the PSD, MA storage. MA then ORs with NMASTER -slave- to enable the AC abort logic. The output of MA also is gated with AXPSW2 to set A8 -S/A8- when the PSD is stored back in memory. MA also lights a lamp on the processor control panel (PCP).

MAP CONTROL.

The CPU is set in the mapping mode by bit 9 of the PSD (PSD 9), labeled MM on the drawing. Signal PSW1XS on A25 gates the first PSD from the S buss at the proper time (existing signal). Clock CL loads F2. This will put the CPU into the map mode except if map disconnect F-F (MAPDIS) is set or during MMC phase 6 when MAP, WL or AC memories are being loaded, (LOCKW) on A22. The NOT output of the MM F-F is used to light a light on the PCP and set F-F A9 at the proper time when the PSD is stored(LPSD or XPSD).

ACCESS CONTROL TIMING.

ACCTIM signal at gate A6 (see drawing #5), enables the ABO/3 signal for the cases in which an instruction access is inhibited. PH10 -phase 10- is the normal time when an instruction is fetched for the next operation. During a branch instruction however the branch-to address is real during prep phase 3 but the branch condition is not determined until phase 1. Therefore the abort at Prep 3 is inhibited and the abort at phase 1 is dependent upon the branch and the address that the CPU will branch to.

The branch conditional timing was implemented on card 02B. The exact branch conditions will not be detailed since they are described in the technical manuals.

The modify and test instruction in location 55 does not set the map disconnect F-F as mentioned above. However the access protect must be disabled. To inhibit an abort the S/ABO1 signal is grounded when this instruction occurs during an interrupt, (FAMT) (INTRAP), at A34.

PH10 MODIFICATION.

Another problem with the abort timing was caused by excessive delay through the logic that prevented the setting of PH10 from PH1 of a branch. See diagram #7. The ABORT signal was routed around the (S/TRAP) logic and ORed with the NCLEAR signal just prior to the S/PH10 input to PH10 F-F. The gate for this is located on card 02B, A36.

NBRP.

If a trap occurs during certain phases when the program counter is in the P register the return location will be lost. The F-F BRP indicates where the program counter is (B or P), as opposed to the operand address. If the program counter is in P the trap is inhibited until the address has been saved in B. The inhibit is generated at OR gate 033.

P LINE MODIFICATION.

The P lines to the main memory are replaced with the P/1 lines from the map memory. See block diagram #3. These lines are switched at the linedrivers from the CPU to the main memory door. Therefore all logic in the CPU receives the true P register bits except the Write Lock memory which receives the P/1 address from the mapping card.

DIAGNOSTIC PROGRAM FOR MAP AND ACCESS CONTROL

To facilitate testing the added hardware for the memory map and access control, an extensive diagnostic program was written. The diagnostic program can only run stand-alone, and loads itself with a bootstrap operation from tape. To create the tape, the program is assembled, loaded, and run under the normal operating system; when this is done, the only portion of the program that is actually executed is a small routine which copies the diagnostic program in core-image onto tape, preceded by a bootstrap record.

The diagnostic program is divided into two parts. The first part tests the memory map, and the second part tests the access control. Both parts are table-driven as much as possible. All error information is printed on the line printer.

The memory map test verifies that each page (above X'1200') can be mapped onto every other page (above the same limit). Instructions tested are: STW, BAL, LPSD, XW, and indirectly-addressed XW. Clocks 3 and 4 are also tested, to verify 1) that the "equal-zero" interrupt XPSD will store through the map if bit 10 in the XPSD is set and the interrupted PSD is mapped, and 2) that the clock 4 "count-pulse" MTW will map its operand if the running PSD is mapped, but the clock 3 "count-pulse" MTW will never map its operand.

The access control test verifies (for each page above X'1200') that traps occur as expected when protected memory is accessed. The following CPU running modes are tested: master unmapped, master mapped, slave unmapped, slave mapped, and master protected. The test verifies that the PSD bits defining these modes are correctly set. All four access control codes are tested. In both trap and non-trap cases, the test verifies that register

contents are altered as expected. Various instructions are tested with a total of 49 tests. Loading, arithmetic, storing, and stacking instructions (LW, STW, LD, STD, XW, AWM, MTW, LM, STM, PLW, PSW, PLM, PSM, MSP, ANLZ) are tested for operand access. Of these, the multiple-register instructions are also tested for the case when the operand word string crosses a page boundary. Access of indirect addresses is tested. Branching and state-switching instructions (BCR, BIR, EXU, LPSD, XPSD) are tested for access of the instruction branched to, as is instruction "fall-through" across a page boundary. Also, a test is made to verify that the clock 3 and clock 4 "count-pulse" MTW's are exempt from access control.

REQUIRED CHANGES TO CP-V

The map, the access control, the MMC instruction (op code X'6F'), and the PSD bits controlling the map and access control are programmed identically as on Sigma 7 hardware, with the addition of the PSD "mode altered" bit (PSD bit 40), which provides "master-protected" mode as on a Sigma 9, though "real extended" addressing is not implemented. Therefore, no software changes are necessary relating to the map or access control. The only requirement for software changes arises from the need to simulate the instructions existing on a Sigma 7 but nonexistent on the (modified) Sigma 5, namely, the byte string, decimal, and convert instructions.

At boot time, routine INITIAL in module INITIAL executes a test instruction to determine whether a system generated for extended memory (>128K) is being run on a machine that cannot support extended memory. This test was changed to avoid the missing instructions, though it will still fail on the Sigma 5 if the system being run was generated for extended memory, since neither the modified Sigma 5 nor the simulator supports extended memory. Routine NSIG7TRP in module BOOTSUBR makes a further test to determine whether the system is being run on a Sigma 7, Sigma 9, or 560, and sets a flag at CS\$MACHINE (in module M:CPU) accordingly. This test was left unchanged, and will give the same result on the Sigma 5 as on a Sigma 7.

Since the traps and interrupts serviced by PFSR and CS\$7 (namely: X'50', X'51', X'56', X'46') behave the same way on Sigma 5 hardware as on Sigma 7 hardware, these modules need not be changed.

Since the missing instructions trap to location X'40' as nonexistent operation codes, the "new" PSD for trap 40 (NOPPSD in module TCATCH) was changed to point to NOPPGMX in the simulator (which is an added module, S5SIM). Thus,

the simulator intercepts all occurrences of trap 40, unless the XPSD instruction in location X'40' is replaced, effectively disconnecting the simulator. Code in module ENTRY was also changed to facilitate trap interception. Since executive delta and crash recovery both overwrite location X'40', all instances of the missing instructions in executive delta and crash recovery were recoded using Sigma 5 instructions. This affected modules XDLT, RCVCTL, CYCUSR, TSTHGP, SYMFILS, and RCVRIO. This method was used in order to keep executive delta and crash recovery each as self-contained as possible.

The simulator (more precisely, the portion of the simulator beginning at NOPPM2 entered only as a result of a nonexistent op' code trap) must be reentrant for unmapped (monitor) usage, though it need not be reentrant for strictly mapped usage (since routine T:MASTER in module SCHDSUB prevents a task switch from occurring as a result of an interrupt if the "interrupted" PSD was master unmapped or master mapped with instruction address below X'8C00'). Reentrancy is accomplished by pushing all temporary storage into the TSTACK (defined in module JIT). To simplify the code, this is done in the same way in both the mapped and unmapped cases.

If the "trapped" PSD has any interrupt inhibits set, than all three interrupt inhibit bits will remain constantly set during execution of the simulator.

Since the simulator uses the TSTACK, all instances of the missing instructions that could occur before the TSTACK is initialized were recoded using Sigma 5 instructions. This affected modules INITIAL, BOOTSUBR, RECONFIG, and XDLT.

Merely to facilitate finding instances of the missing instructions, SYSTEM UTS was changed so that if the variable S5PROC is set to 1 before SYSTEM UTS is called, and the variable UTSPROC is not explicitly set to 0, then SYSTEM UTS will call SYSTEM SIG5P instead of SIG7FDP. This will cause the missing instructions to show up as errors in the assembly listing.

The simulator employs the Sigma 9 "master-protected" mode (in which the access control is in effect though the CPU remains in master mode with map) to enforce protection of operands of simulated slave-mode instructions, except when the destination address of a byte string instruction capable of changing core memory lies in the general registers. This method is faster than testing the image held in core of the access control bits before simulating the instruction.

Since the simulator must be capable of running in "master-protected" mode, the memory pages containing the simulator must not be "branch-protected"; that is, they must have access control code 01, giving write-protection only. The

routine LMA in module INITIAL was changed to make this exception. Since the simulator (and neighboring code) is pure (invariant) procedure, this change cannot affect system or user security.

Since a trap 40 because of nonexistent or protected memory can occur in the simulator, these cases must be distinguished from cases of these traps occurring when the simulator is not truly being executed (for example, when a stray branch lands somewhere in the middle of the simulator). To accomplish this, a flag is stored in the JIT when the simulator is entered. Treatment is identical in both the mapped and unmapped cases. Location J:ALB in module JIT was used for this purpose, and the definition of J:ALB (which was used only to hold the contents of the 560 "last branch" register), and all existing references to J:ALB, were removed. The location was renamed J:SIMSP. When the simulator is running, its contents are as follows:

- bit 0: bit 8 (slave) or'ed with bit 40 (mode altered)
from trapped PSD
- bit 1: bit 9 (map) from trapped PSD
- bits 2-7: bits 10-15 (dec mask, etc.) from trapped PSD
- bit 8: "or" of inhibit bits from trapped PSD
- bit 9-14: all zero
- bits 15-31: pointer to stacked register 0 in TSTACK

J:SIMSP is initialized to zero in the "skeleton" JIT.

When the simulator is entered at NOPPM2, the old contents of J:SIMSP are saved, along with the trapped PSD and general registers, in the TSTACK. The new value of the flag is stored in J:SIMSP, and then decoding of the instruction to be simulated begins. When the simulator exits (due to successful completion or due to nonexistent or protected memory trap), the old value of J:SIMSP is restored. When a trap due to nonexistent or protected memory occurs, J:SIMSP (bits 15-31) is compared with the contents minus 15 of location TSTACK, to determine whether the simulator was truly executing. If the simulator was truly executing, the relevant registers and PSD are, of course, already stacked.

All occurrences of trap 40 not requiring instruction simulation (namely: privileged instruction in slave mode, nonexistent op code other than byte string, decimal, or convert instructions, and nonexistent or protected memory) are made to look just as though the trap were never intercepted.

If simulation of a decimal instruction encounters overflow or an illegal digit or sign code, and the trapped PSD has bit 10 (decimal mask) set, then entry into trap 45 will be simulated just as though the decimal hardware existed.

To avoid undesirable task switching and to reduce overhead, the simulator never exits to T:SSE or T:PULLE (in module SCHED), but rather contains its own routine to pull the stacked registers and load the return PSD. If the return PSD is slave mode, and the clock 4 counter has run out, then the clock 4 "equal-zero" interrupt is triggered so that it will occur just after the return PSD is loaded.

During instruction simulation, the clock 4 "count-pulse" MTW instruction is left unchanged from what it was just before the trap. Thus, time spent in the simulator is generally accounted for as execution time, not overhead time.

The simulator has the following peculiarities (most of which could be corrected at the expense of slowing down the simulator):

- a. If any operand "wraps around" from location X'1FFFF' into register 0, or overlaps from register 15 into location X'10', the results will be meaningless.
- b. Simulation will generally proceed until nonexistent memory or a memory protection violation is detected. When such a trap occurs, an unprotected portion of a destination operand in memory may already have been altered, though the general registers (as seen in a dump) will remain at the values they had just before the instruction being simulated was encountered.
- c. If the destination operand of a byte string instruction capable of changing core memory is below X'10', i.e., lies in the general registers, then "read-protection" for the source operand will not be enforced.
- d. Byte string instruction simulation is done strictly byte-by-byte, not necessarily the fastest way.
- e. If the instruction to be simulated is EXU'ed, and the trapped PSD was master mode, then inhibits remain constantly set while the EXU chain is being decoded.
- f. For decimal instructions except EBS, the contents of registers 12 through 15 (the "decimal accumulator") are assumed to have been established by the previous execution of a decimal instruction and are not tested for validity, except for the sign code. All operands

- fetched from memory or general registers in response to the instruction's effective address are, however, completely validated.
- g. The simulation of the EBS instruction (edit byte string) ignores PSD bit 12 (ANSCII/EBCDIC mask bit), and always produces EBCDIC codes. The source digit string is not checked for validity.
 - h. If Sigma 9 "real extended" addressing were implemented on the Sigma 5, the simulator would fail for instructions executed in "real extended" addressing mode.

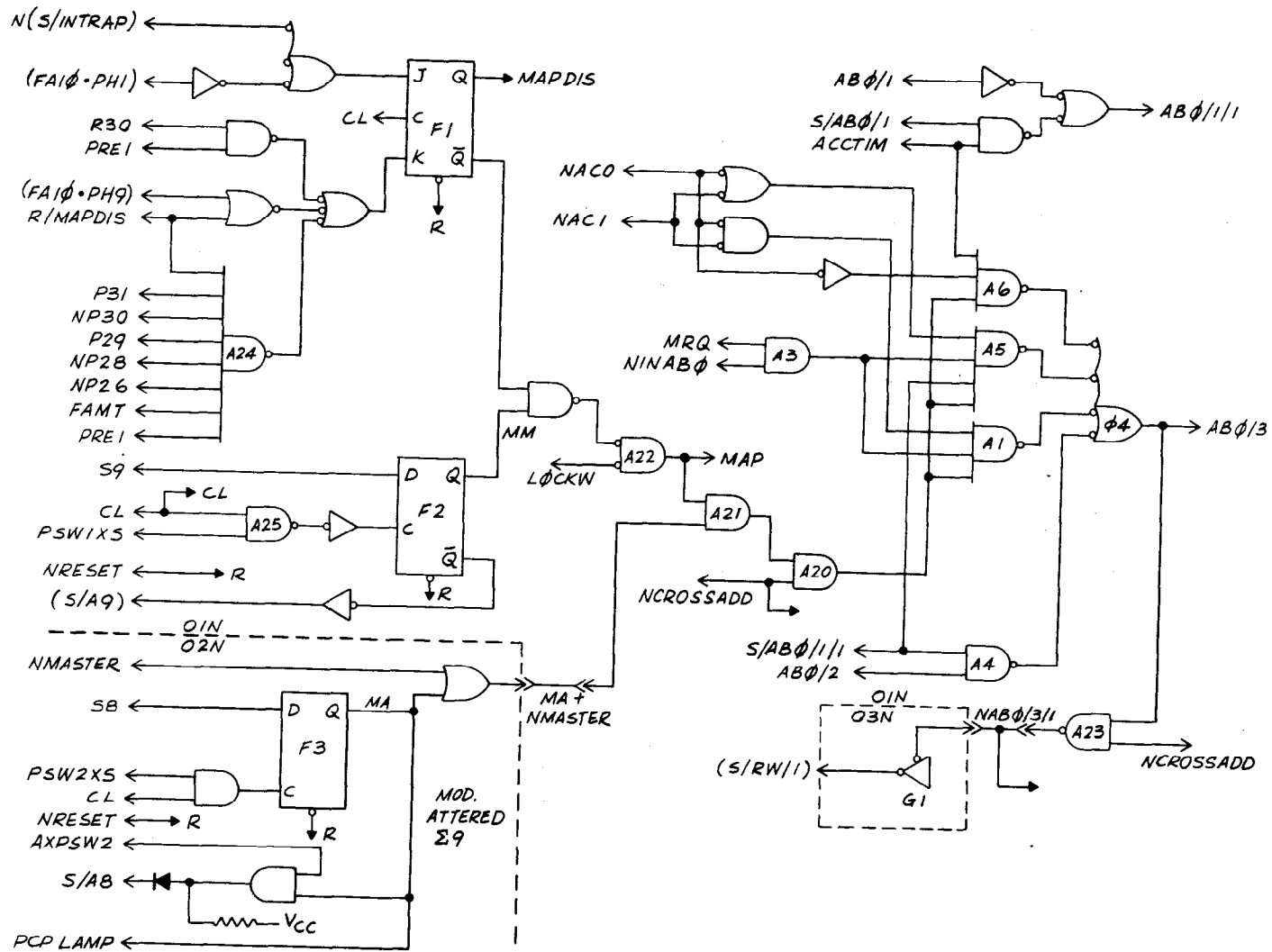
Since the methods used for simulation are very nearly the same as those used in the simulators provided by Xerox for BPM, timing should agree fairly closely with figures quoted in the Xerox BPM System Technical Manual (catalog number 90 15 28A, July 1971, page 18-2. Of all the missing instructions, MBS and CBS are undoubtedly the most frequently used, and are probably the only ones whose simulation may cause noticeable degradation. I estimate their timing as:

MBS: 340 + 15.9N microseconds

CBS: 340 + 16.5N microseconds

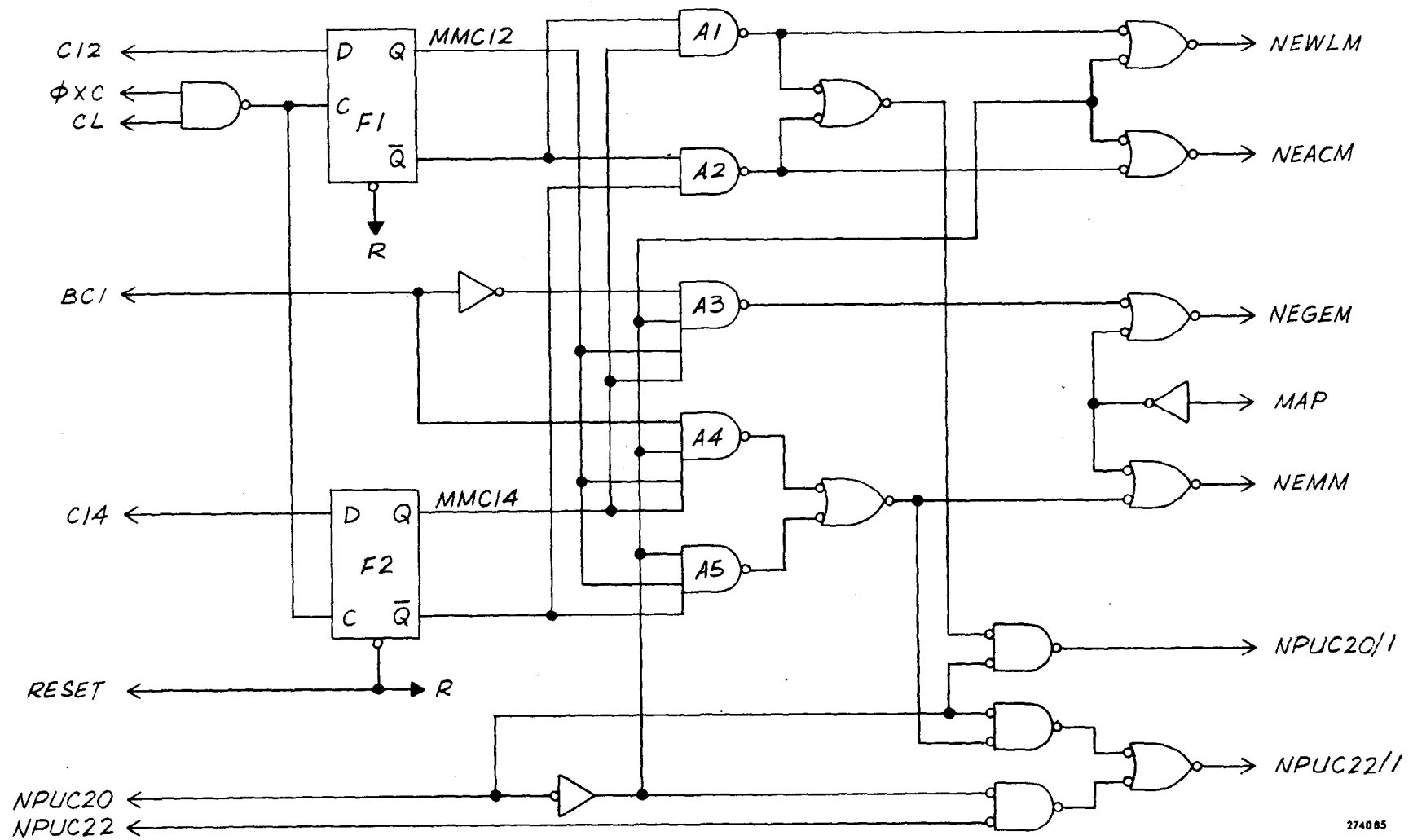
The best way to alleviate any degradation caused by simulation of missing instructions would be to recode them in-line wherever they occur (especially in the file system) using Sigma 5 instructions. Other instances of MBS instructions that should be recoded are: 4CHAR in module HANDLERS, and ERRLOG in module ERHNDLR.

Very few changes of the SYSGEN procedure are required. All altered or added modules should be assembled, and the LOCCT for M:MON should be changed to include the simulator, S5SIM (which is the only added module), in the root in such a way that at least one page of pure procedure precedes and follows S5SIM. The :MON command for PASS2 should specify SIG6, and should not specify BIG.



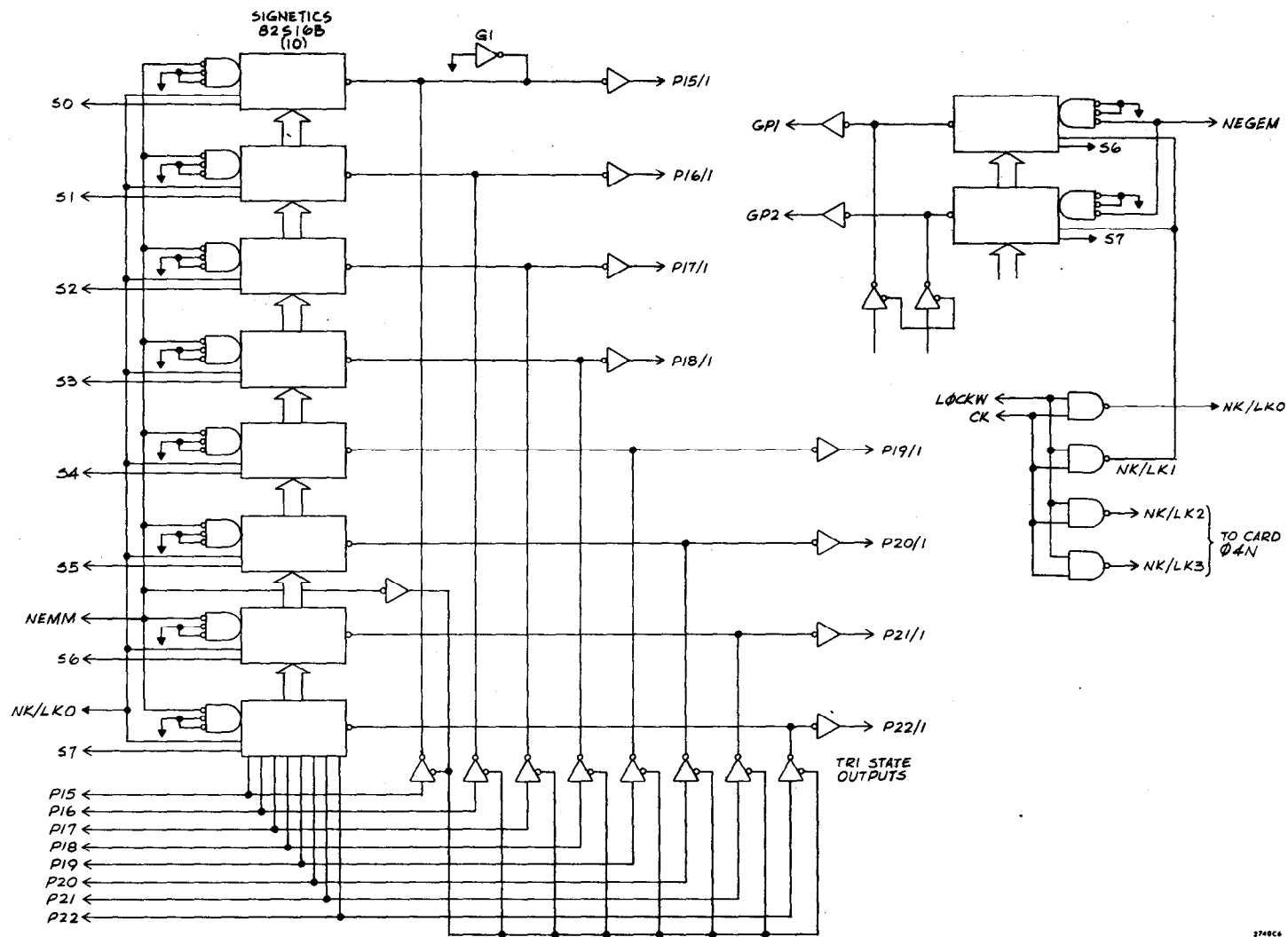
2740C4

Fig. 1



274085

Fig. 2



2749C6

Fig. 3

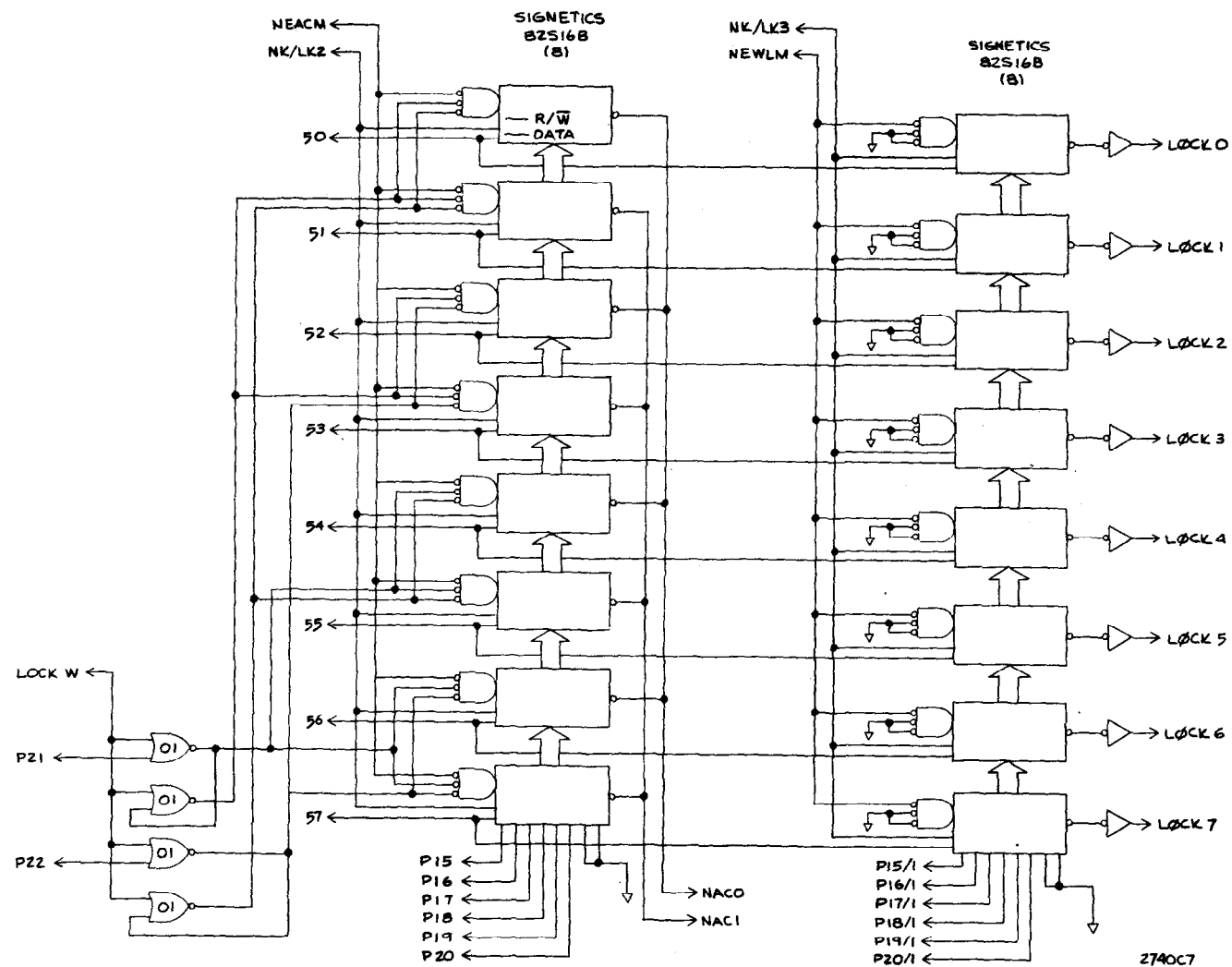
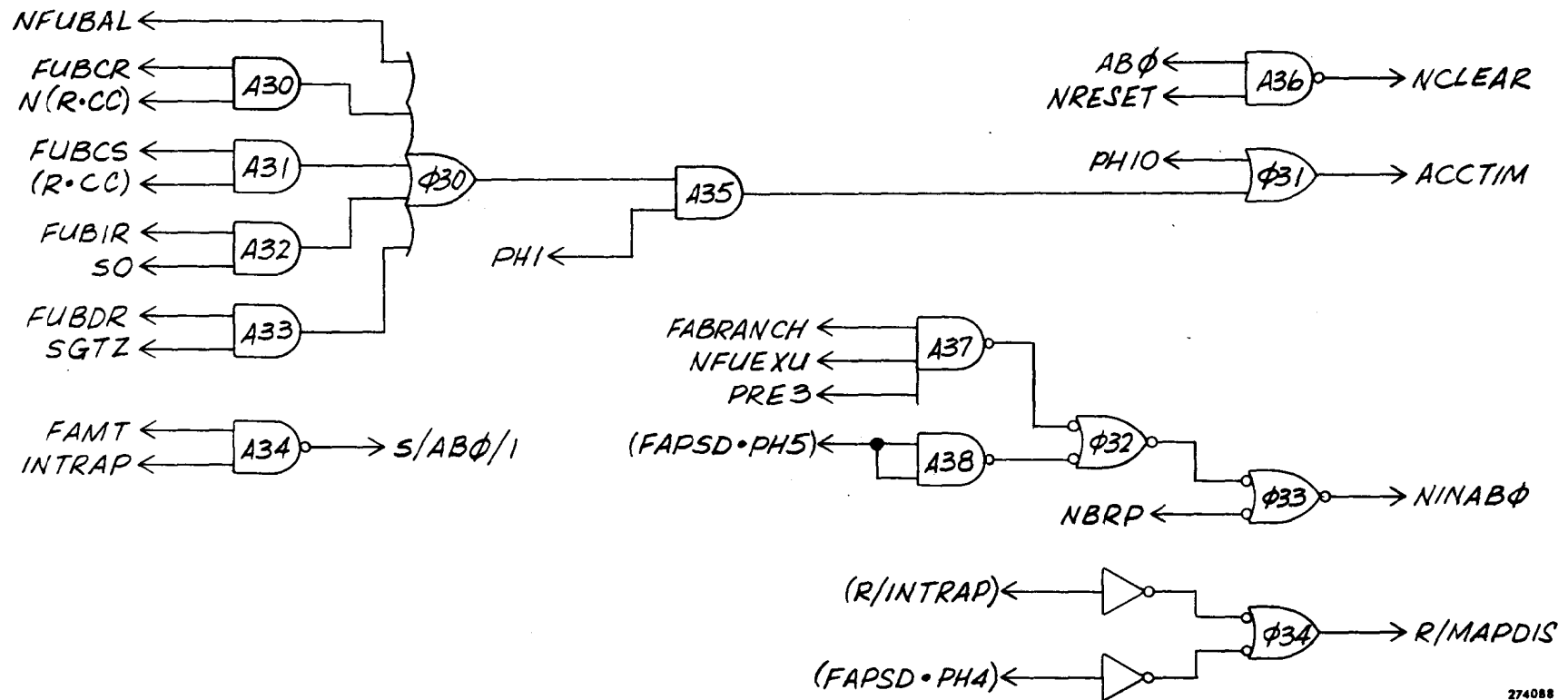


Fig. 4



274088

Fig. 5

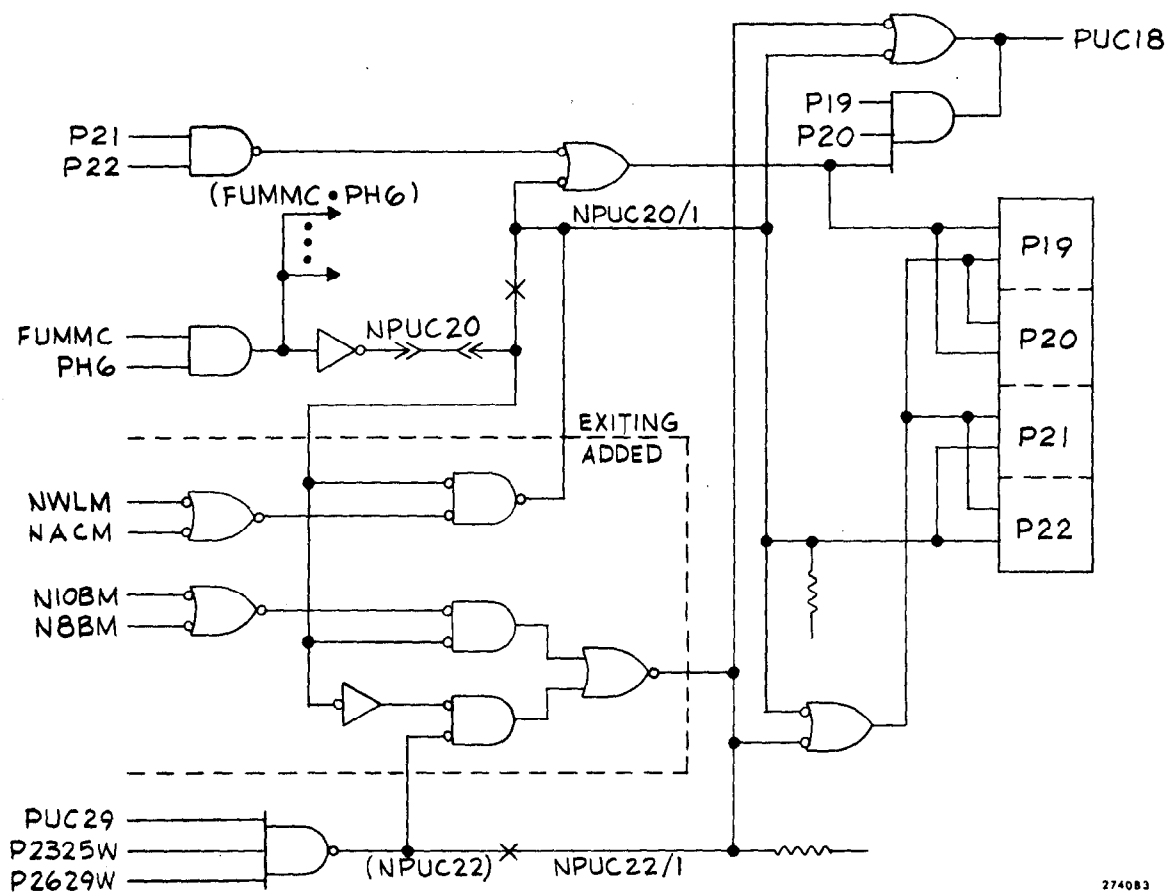


Fig. 6

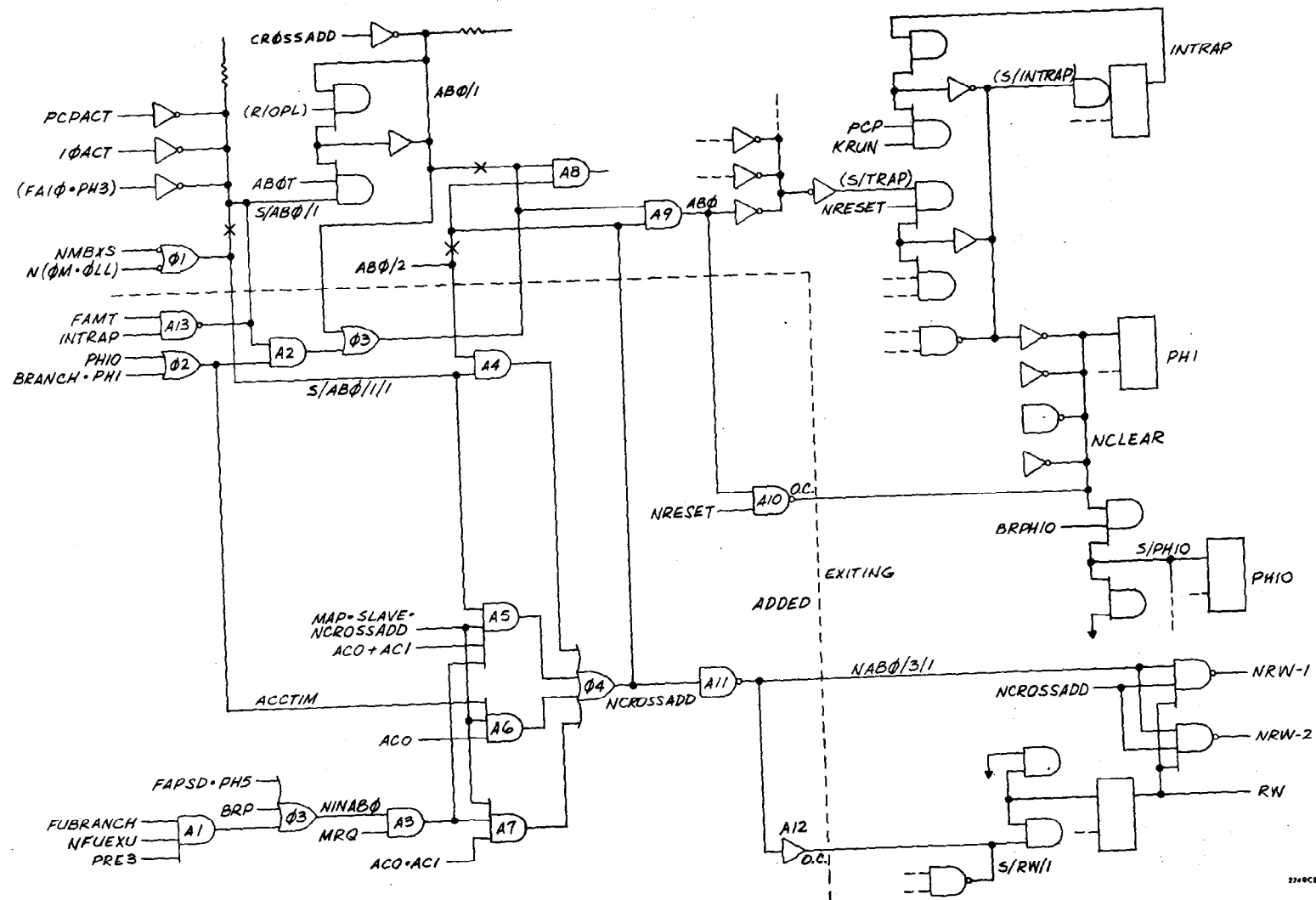
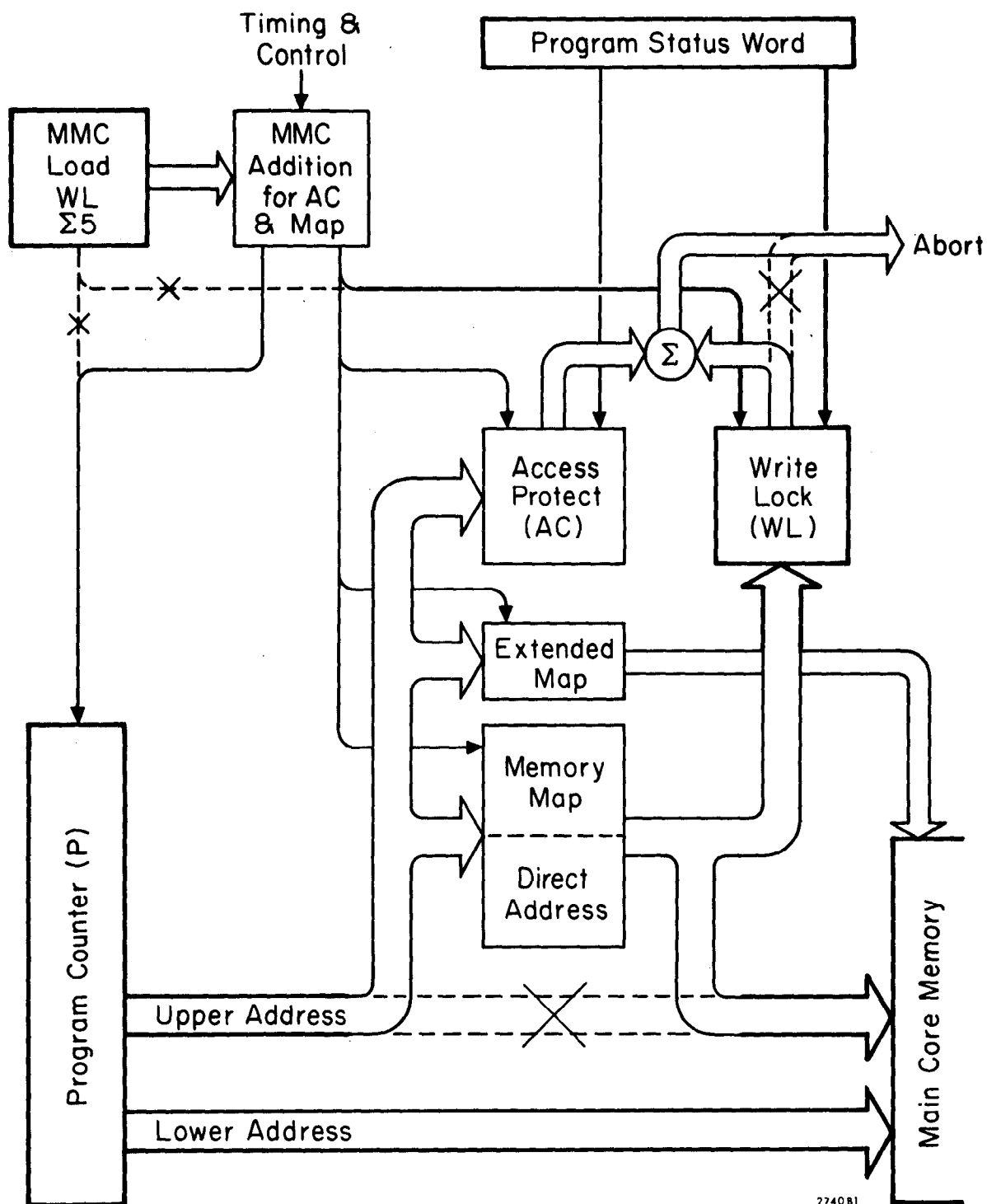


Fig. 7



274081

Fig. 8