2.3 Image Generation Controller

Principal Author: John Eyles

Microelectronic Systems Laboratory

The University of North Carolina at Chapel Hill
Department of Computer Science
New West Hall 035 A
Chapel Hill, N.C. 27514

These changes made in IGC-2 doc

2.3. Image Generation Controller

Introduction

Graphics Engine.

The ~~image-generation~~ controller (IGC) ~~is the module which~~ supplies all the signals necessary for image generation to the Pixel-planes ~~memory chips~~; this includes all the signals supplied to the memory chips, with the exception of the signals which are used for video scan-out and for loading the configuration and alive registers. These signals are supplied by the Video Controller; operation of the Video Controller is completely independent of the operation of the IGC, with two exceptions: first, each Pixel-Planes memory address representing video intensity data must be "visited" by the IGC at least once per video scan line; second, the Video Controller must supply a signal to the IGC marking the beginning of video retrace ~~(VRBHLXF)~~, so that the copy operation for double-buffering occurs during retrace.

The function of the IGC is to accept input from the translator in the form of basic commands (such as "setenabs", "edge", "zload") and corresponding values of the planar coefficients (A, B, and C), and produce as output the signals necessary to generate images in the Pixel-planes frame buffer.

The image-generation controller will be implemented in 2 forms, one used in a basic Pixel-planes graphics system and the other used in a higher performance buffered Pixel-planes system. The IGC for the basic system will be implemented using discrete memory and logic components and one custom IC (the Coefficient Serialization Chip). In the first version of the basic system, the IGC will accept input from a parallel interface of a host computer, ~~as in the Version 3 prototype~~. In later systems, the IGC will accept input from ~~a hardware~~ the translator ~~unit~~, which receives an image description in terms of screen-space vertices (x,y,z,R,G,B) and outputs basic commands to the IGC. In the buffered system, the IGC will be implemented as a single custom ~~IC~~ chip, and each IGC will control one or more Pixel-Planes memory chips, with the complete system containing many modules, each consisting of one IGC chip and its associated memory chips. In the buffered system each IGC is paired with a FIFO/arbitration module, probably implemented on the same chip, which in turn receives input from a translator similar to that in the basic system; the FIFO/arbitration module passes to its IGC only those basic commands which affect the pixels represented by the Pixel-Planes chips under the control of that IGC. This provides the potential for a great increase in performance, as detailed elsewhere. In the ultimate buffered system there is one IGC for each memory chip, and one large IC incorporates a ~~Pixel-Planes~~ memory chip, an IGC, and its associated FIFO/arbitration module.

2.3.1. Input Interface

The several versions of the IGC described above will essentially be identical in their external interfaces. Inputs to the IGC are: a 32 bit data word (D0 - D31), 3 type bits (T0, T1, and T2), a write enable signal (WrEnLLCR), and a clock CLK. The data bits, type bits, and write enable signal should be of signal type -LCR; that is, they should change on the rising edge of CLK. A write to the IGC occurs if and only if WrEnLLCR is low on a given CLK cycle; D<0:31>HLCR and T<0:2>HLCR are ignored if WrEnLLCR = 1. The interface also includes 2 signals which are sent from the IGC back to the translator: BusyHLPR indicates that the IGC is "busy" and that no further writes should be attempted; DoneHLPR, if it remains hi for 3 or more consecutive clock cycles, indicates that the IGC has finished processing all instructions which it has received.

Err indicates that an illegal write to the IGC has occured

The data bits D<0:31>HLCR are interpreted in different ways, according to the values of the type bits T<0:2>HLCR, as shown in Table 2.3.1. The D and L word types are used only when initializing the IGC or reloading the microcode store; their use is described more completely in the section "Loading the Microcode Store".

Type bits may be thought of as selecting 1 of 8 32bit registers.

An instruction sent to the IGC consists of 1-5 words. The I word is required, and it must be the *last* word sent since it "triggers" the instruction. The P, A, B, and C words are optional and may be sent in any order. The A, B, and C words are sent only if the instruction is one which requires the planar coefficients. The P word is sent only if the instruction requires more than one

T2	T1	T0	WORD	FUNCTION
0	0	0	A	defines the planar coefficient A in IEEE standard floating-point format: D22-D0 represent the fractional part of the unsigned mantissa, which is understood to be between 1 and 1.111... D30-D23 represent the exponent in excess 127 form D31 is the sign bit (1 is negative)
0	0	1	B	defines the B coefficient in IEEE standard floating-point format
0	1	0	C	defines the C coefficient in IEEE standard floating-point format
0	1	1	I	defines the basic instruction to be executed, as follows: D7-D0 (McodeAddr) represent the starting address of the microcode sequence which executes the instruction D14-D8 (StDstAddr) represent the low address of a segment of Pixel-Planes memory to be accessed D21-D15 (LpCnt1) represent a count to be used in executing some loop in the instruction D22 (CVI) when hi, indicates that the instruction is one which uses the A, B, and C coefficients D31 (MBI) when hi, indicates that the tree result (Ax+By+C) must be fully formed (to the sign-bit) for this instruction D30-D23 (FNI) are encoded to represent a fixed number of bits of the tree result which must be formed for the instruction (see below)
1	0	0	P	supply supplementary parameters for more complex instructions, or to reset IGC parameters, as follows: D14-D8 (StSrcAddr) represent the low address of a second segment of Pixel-Planes memory to be accessed D21-D15 (LpCnt2) represent a second loop count D26-D23 (FBITS) reset the number of fractional bits of precision in the fixed-point representations of the coefficients (0 to 15, with D23 as LSB) D30-D27 (REFCNT) reset the range of Pixel-Planes memory addresses which are automatically refreshed by the IGC (addresses 0 to 8*REFCNT+7 are refreshed)
1	0	1	X	NOT USED
1	1	0	D	a word of microcode to be held in the Load Latch, for subsequent writing into the microcode memory store of the IGC
1	1	1	L	controls loading of the microcode memory store, as follows: D8=0 indicates that a write to the microcode store is to occur on this clock cycle D9=1 enables writing to the microcode store in fast mode, until another L word is sent with D9=0 D10=1 enables operation of the Microcode Output Latch, until another L word is sent with D10=0 D7-D0 represent the address in the microcode memory store at which a microcode word (held in the Microcode Load Latch) is to be loaded

Table 2.3.1: Types of Input Words for IGC

These changes made in IGC-2

loop count or more than one address sequence. If the instruction is one for which the P, A, B, or C words normally would be sent and any of them is omitted, then the most recently sent word of that type is used. (This means, for example, that if two successive instructions used the same value for the B coefficient, then a B word would not have to be sent for the second instruction).

The P word also serves two other purposes: it is used to specify the number of fractional bits of precision of the coefficients (FBITS), and to specify the address range for automatic Pixel-planes memory refresh (REFCNT). This may be done by sending a separate P word, or the information may be included in a P word sent as part of an instruction. Note that whenever a P word is sent, the bits representing FBITS and REFCNT must be set to the currently selected value, else these parameters will be changed when the P word is loaded. Two precautions must be observed whenever the number of fractional bits (FBITS) is changed: A, B, and C must be sent again, even if they are the same as for the previous instruction, and the P word used to change FBITS must precede the A, B, and C words.

After an I word has been written to the IGC, concluding the transmission of an instruction, BusyHLPR goes hi, indicating that the IGC is "busy", and no further writes to the IGC may occur until BusyHLPR is lo. That is, to write the I word which concludes an instruction, the translator asserts WrEnLLCR for one CLK cycle, as it does when writing any type word to the IGC; BusyHLPR will go hi immediately, and the translator must not assert WrEnLLCR again until BusyHLPR has gone lo. The translator may then send all the words of the next instruction (ending with an I word) without repolling BusyHLPR.

WrEnLLCR may remain low for consec. cycles when writing several words, but must go hi after I word.

The DoneHLPR signal indicates, when it stays hi for 3 or more consecutive clock cycles, that the IGC has finished processing all instructions which have been transmitted. If the IGC microcode is to be reloaded on the fly, the translator should poll DoneHLPR before sending the D and L words to reload the microcode.

Err indicates . . .

2.3.2. Output Interface

The output interface of the IGC consists of 24 signals which control image generation in the Pixel-planes enhanced memory chips (known as the image generation vector IGV) along with the clock PhH. The 24 IGV signals change on the rising edge of PhH (signal name suffix -HLPR).

2.3.3. Theory of Operation

The IGC is based on a conventional microcode engine, consisting of a microcode memory store and logic for sequencing the microcode based on the status of various condition flags; no subroutine linkage is implemented. A basic instruction is implemented by executing a series of words from the microcode memory. The microcode engine controls a set of shift registers which bit serialize the A, B, and C coefficients, handle numerical exceptions, and return signals which mark the LSB and MSB of the tree result Ax+By+C to the microcode sequence logic. A interface/control structure governs the flow of instructions within the IGC.

The 24 signals of the image generation vector are generated by 3 separate structures in the IGC. The 10 ALU controls (controls for the a, b, and c ports of the ALU and the Carry and Enable registers) and the 2 pixel memory write controls are provided directly by the microcode memory; ShiftHLPR (TrStHLPR) is provided by the microcode memory but may be inhibited under certain circumstances (in the Shift Generation Logic). The Pixel-Planes memory address bits (Add6 - Add0) are provided by the Pixel Memory Address Sub-Module; it contains a set of counters and a multiplexer, which are controlled by 3 bits of the microcode word. The bits of the A, B, and C coefficients (ADat, BDat, and CDat) and LSB are provided by the coefficient shift registers, which convert the coefficients into fixed-point numbers with FBITS fractional bits.

In order to achieve optimum performance, it is necessary to introduce two complications into this microcode control scheme: first, the magnitude of the coefficients, and hence the number of microcycles (Ph clock cycles) necessary to fully form the tree result, will vary from instruction to instruction, so the length of an instruction is not known a priori; second, it is desirable to overlap tree operations, shifting the coefficients for an instruction into the Pixel-Planes memory chips' multiplier/tree structures while the previous instruction is still being executed at the Pixel-Planes chips' ALU and memory. The solution to these problems is to provide, in addition to shift registers to bit serialize the coefficients, a mechanism which provides signals which mark the location of the LSB and MSB of the coefficients. These LSB and MSB signals can be used as condition flags for the microcode sequencer; this allows bit serialization and shifting into the Pixel-planes chips' multipliers of the coefficients of an instruction to begin as soon as the correct number of bits of the coefficients of the previous instruction have been shifted in. Thus coefficients from the next instruction may be shifted in even while execution of the current instruction is being completed, thereby allowing overlapping of tree operations.

Instructions pass through what effectively is a 3-stage pipeline within the IGC. In the first stage of the pipe, the words comprising an instruction are loaded into various latches within the IGC. In the next stage of the pipe, the P and I words are transferred to another set of latches and bit-serialization of the A, B, and C words begins (so that the coefficient bit-streams can begin being shifted into the mutliplier trees of the memory chips); at this point, the instruction is said to be "pending". The instruction enters the final stage of the pipe when execution of microcode for the previous instruction has completed. Serialization of the coefficients continues if necessary, the loop count and pixel-memory address fields from the P and I words are loaded into their respective counters, the starting microcode address from the I word is loaded into the Microcode Address Latch (program counter), and execution of the microcode begins.

2.3.4. Structural and Functional Description

The first version of the Image Generation Controller will be implemented using a combination of 2 technologies: discrete TTL logic and MOS memory timed by the 1-phase PhH clock (derived by inverting the CLK supplied by the translator), and a large VLSI chip, the Coefficient Serialization Chip, timed by non-overlapping 2-phase clocks (Ph1 and Ph2) generated on-chip from PhH. Functionally, the IGC may be divided into the following modules:

(1) the Interface and Control Module, incorporating: logic for generating signals which control the loading of input words into the IGC, and which control the flow of instructions through the 3-stage "pipeline" within the IGC

(2) the Microcode Memory Module, incorporating: the microcode memory store; logic for controlling the sequencing of microcode; a sub-module for computing the Pixel-Planes memory addresses; a sub-module containing loop counters; and a sub-module for controlling loading of the microcode store

(3) the Coefficient Serialization Module (Coefficient Serialization Chip), incorporating: latches for latching the A, B, and C words and the exponent field of the I word from the Input Buffer; a set of structures for bit serialization, 2's complement conversion, sign-extension, and staggering of the A, B, and C coefficients and for providing the LSB signal which marks the beginning of a new set of coefficients; a shift-register for generating the TRRH signal which marks the LSB of the tree result Ax+By+C; a mechanism for generating SRBH (shift registers busy) which goes lo when the coefficients have been sign-extended or truncated to sufficient length, and the coefficient shifters can therefore be loaded with the next set of coefficients; a shift-register which generates the MBRH signal which marks the MSB (sign-bit) of the tree result. Figure 2.3.1 shows a block diagram of the IGC.

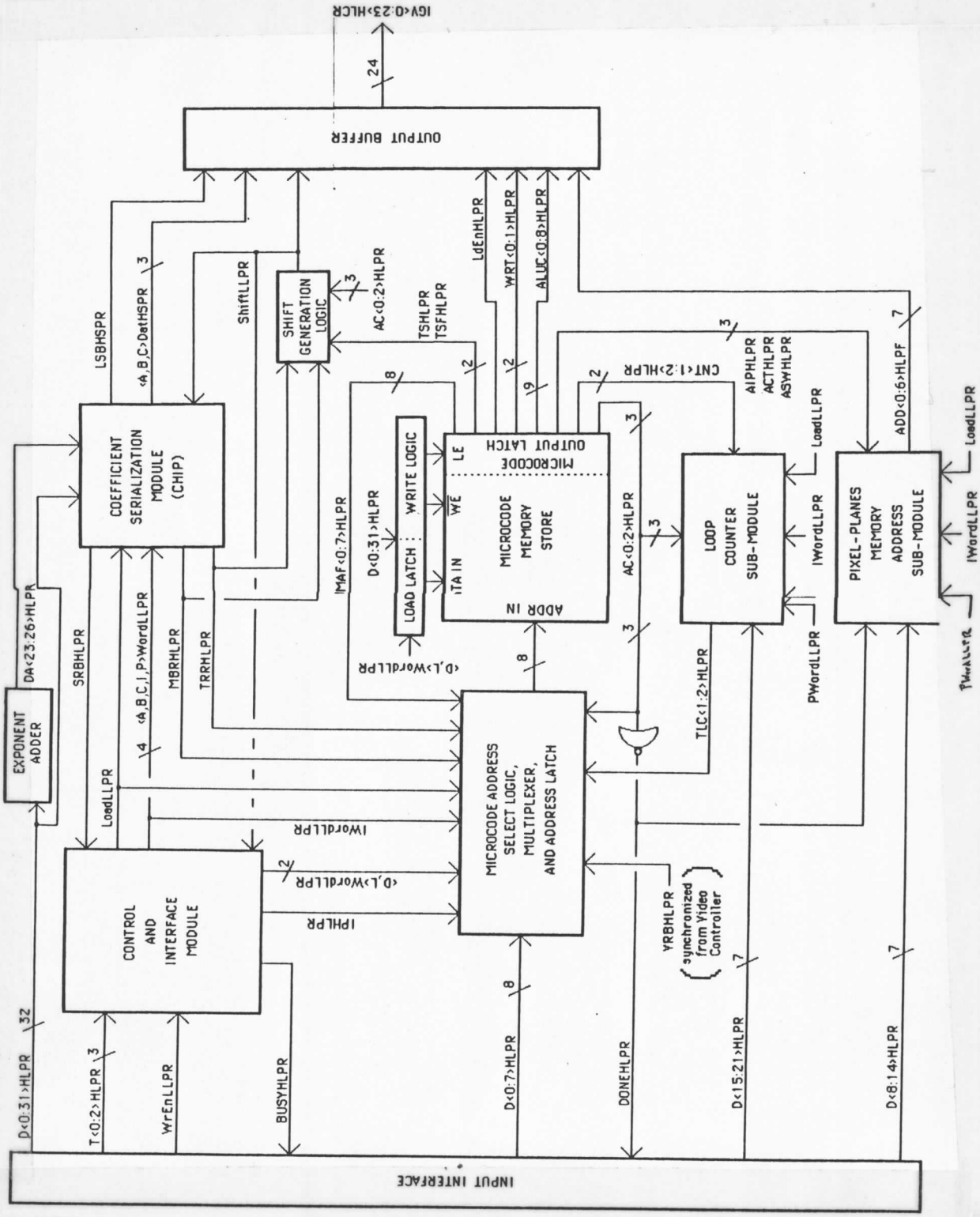


Figure 2.3.1: IGC Block Diagram

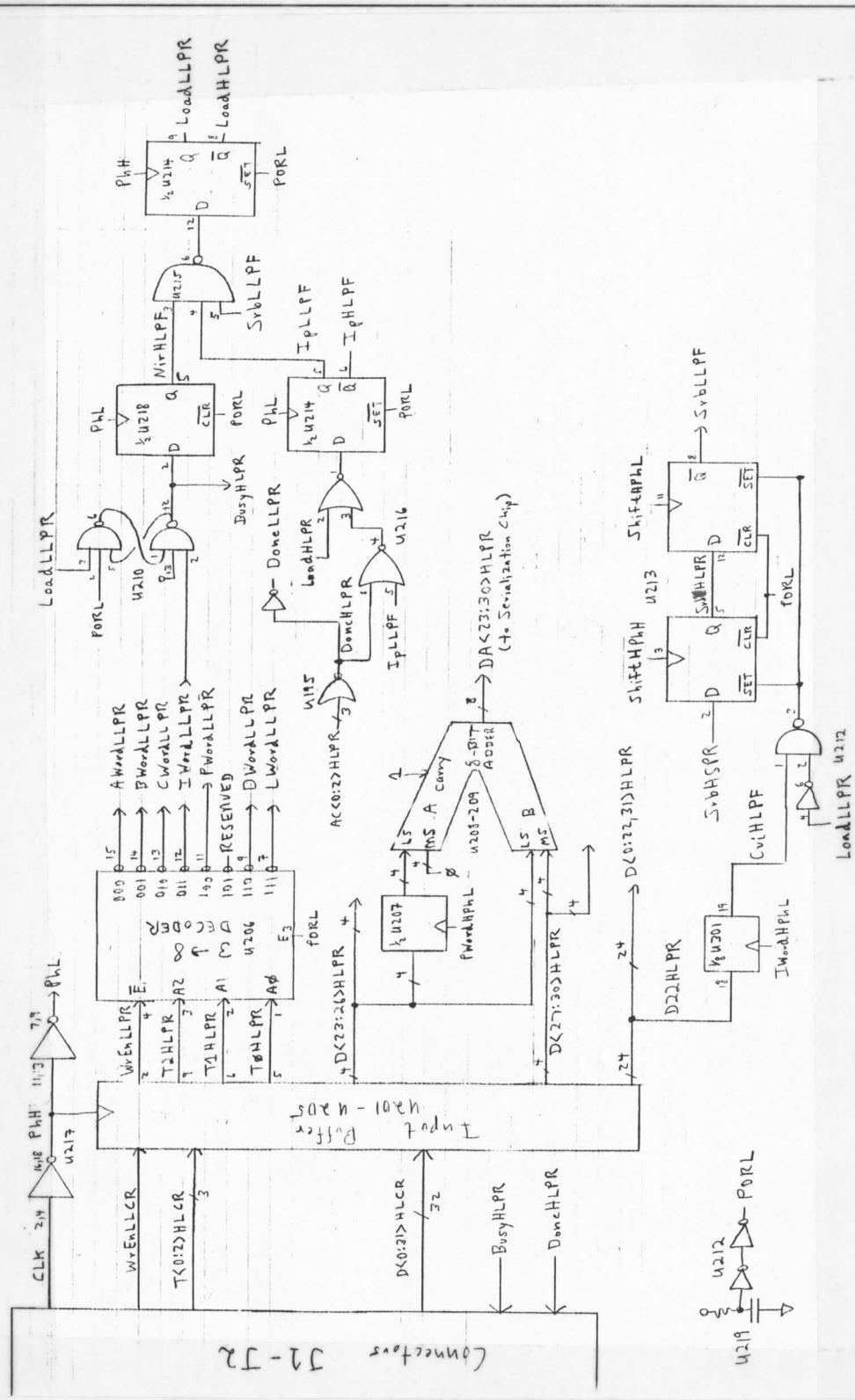


Figure 2.3.2: Interface and Control Module

2.3.4.1. Interface and Control Module

Figure 2.3.2 shows the Interface and Control Module. The input clock CLK is inverted and becomes PhH, the clock for the IGC. (This clock PhH becomes Ph, the image-generation clock for the Pixel-Planes memory chips, after being inverted at each stage of the pipelined tree which distributes the IGC signals to the memory chips). The input signals D<0:31>HLCR, T<0:2>HLCR, and WrEnLLCR are latched into the Input Buffer on the rising edge of PhH, and become of type -LPR. Thus the input signals have half of a CLK cycle to stabilize.

On any clock cycle for which WrEnLLPR (the input WrEnLLCR latched on PhH) is active, the type bits T0, T1, and T2 (represented by the signals T<0:2>HLPR) are decoded ~~by a 3-to-7 decoder~~ to produce PWordLLPR, AWordLLPR, BWordLLPR, CWordLLPR, IWordLLPR, DWordLLPR, and LWordLLPR. Each of these signals indicates that the type word represented by the first letter of the signal name is being written to the IGC ~~by the input device~~, i.e.: IWordLLPR indicates that an I word is being written to the IGC. For an A, B, or C word, AWordLLPR, BWordLLPR, or CWordLLPR causes the mantissa, sign-bit, and exponent of the coefficient to be latched into the Coefficient Serialization Module (chip). For a P word, PWordLLPR causes the StSrcAddr and LpCnt2 fields to be latched into the Pixel Memory Address Sub-Module and Loop Counter Sub-Module, respectively. For an I word, IWordLLPR causes the McodeAddr, StDstAddr, and LpCnt1 fields to be latched, and bits 22 and 31 (which define the handling of the coefficients) to be latched into the Serialization Module. IWordLLPR also causes NirHLPF to be asserted, indicating that a new instruction has been latched into the IGC, and causes BusyHLPR to be asserted, signalling the translator that no further words should be written until this instruction has moved to the next stage of the pipe.

Passing of the instruction from this first "input latching" stage to the second stage of the "pipeline" is controlled by LoadLLPR. LoadLLPR is asserted when (1) NirHLPF = 1, indicating that a new instruction has been latched into the IGC (the first stage of the pipeline is full), (2) SRBLLPF = 1, indicating that the coefficients of the previous instruction have been fully bit-serialized, and (3) IpLLPF = 1, indicating that there is no instruction pending (the second stage of the pipeline is empty). When LoadLLPR is asserted on a clock cycle, the instruction is "loaded", meaning that the following operations occur:

(1) the McodeAddr, StDstAddr, and LpCnt1 fields of the I word and the StSrcAddr and LpCnt2 fields of the P word are passed to another set of latches

(2) serialization of the A, B, and C coefficients begins in the Serialization Module.

(3) IpHLPF is raised, signalling the microcode sequencer that an instruction is pending, so that execution of microocode for the instruction will begin as soon as the previous instruction has finished executing.

An instruction enters the third and final stage of the pipe when execution of the microcode sequence for the instruction begins. This stage begins when the Microcode Module executes a "done" word, a microcode word for which the advance control bits (AC0 - AC2) are set for the "done" state; this "done" word may be either the last word of the microcode sequence for the previous instruction, or the microcode word at address 0, which represents the "idle" state. When the "done" word is executed DoneHLPR goes hi, the Microcode Address Select Logic causes the starting microcode address for the instruction to be loaded into the Microcode Address Latch, and execution of the microcode sequence for the instruction begins; IpHLPF is lowered at this point, indicating that the instruction no longer is "pending", so that the next instruction may be "loaded" (transferred to the second stage of the pipe, provided also that SRBHLPR=0, indicating all necessary bits of the coefficients have been generated, and that NirHLPF = 1, indicating that a new instruction is ready). IpHLPF is implemented using a finite state machine with one bit of state, which is IpHLPF itself. IpHLPF is 1 if on the preceding clock cycle EITHER (1) LoadLLPR was active, meaning that an instruction was transferred to the second stage of the pipe (loaded) OR (2) IpHLPF was already hi and DoneHLPR was lo, meaning that an instruction

already was pending and the previous instruction had not completed; conceptually, IpHLPF is generated by an RS flip-flop which is set by LoadLLPR and cleared by DoneHLPR, with set overriding clear.

2.3.4.2. Coefficient Serialization Module

The function of the Coefficient Serialization Module (CSM) is as follows:

(1) convert the floating point A, B, and C coefficients supplied by the translator into bit-serial 2's-complement fixed-point numbers with FBITS fractional bits

(2) introduce the proper relative delays into the bit-serialized coefficients

(3) produce the signal LSB, which controls multiplier pipelining in the Pixel-planes memory chips

(4) produce the signals TRRH and MBRH, which mark the LSB and MSB of the tree result Ax+By+C

(5) produce the signal SRBH, which indicates when a new set of coefficients may be loaded.

The CSM will be realized as one large intergrated circuit, controlled by non-overlapping 2-phase clocks (Ph1 and Ph2) generated on-chip from PhH. A block diagram of the Serialization Module is shown in Figure 2.3.3.

2.3.4.2.1. Exponent Decoder

The heart of the Coefficient Serialization Chip is the Exponent Decoder, which produces the 49 outputs, E<0:48>HEP2. All 49 outputs are precharged on Ph1, and then all but one of these is pulled low on Ph2; thus the output of the Exponent Decoder is a set of "one-hot" signals. Inputs to the Exponent Decoder are DA<23:30>HSP1, D<23:26>HSP1, PWordLSP1, and the clocks Ph1 and Ph2. DA<23:30>HLPR are inputs to the Serialization chip and are generated off-chip in the IGC by adding the IEEE standard floating-point exponent field of the coefficient (D<23:30>HLPR) to FBITS + 1. FBITS is stored off-chip in a register which is loaded by writing a P word to the IGC. The inputs D<23:26>HSP1 are derived directly from D<23:26>HLPR, bypassing the exponent adder. The Exponent Decoder is programmed so that for values of DA<23:30>HLPR representing decimal values from 128 through 175 (with DA23HLPR as LSB), the 1-hot output is seen on outputs E47HEP2 through E0HEP2; that is, EiHEP2 will be high for a decimal value of DA<23:30>HLPR of (175-i). If the value represented by DA<23:30>HLPR lies outside this range, the 1-hot output is on E48HEP2; thus E48HEP2 represents an "out-of-range" number. Since the exponent field of an IEEE floating-point number is represented in excess-127 form, and since it is added to FBITS+1 to produce the signals DA<23:30>HLPR, this means that the valid range for coefficients is with exponents in the range -FBITS to 47-FBITS.

2.3.4.2.2. Coefficient Serializers

A block diagram of a Coefficient Serializer is shown in Figure 2.3.x. The components of a coefficient word pass through a two-stage pipeline within the Serialization Chip, corresponding to the first two stages of the instruction pipeline. When a C word is written to the IGC, CWordLLPR is asserted by the Control and Interface Module, causing the CWordHPh2 qualified-clock to be asserted on-chip. CWordHPh2 causes the mantissa of the C coefficient to be latched into the C Mantissa Latch and the sign-bit of C to be loaded into a latch in module SepCom. Simultaneously, the exponent is decoded by the Exponent Decoder and the 1-hot output is loaded into the C Token Latch; the single 1 which is loaded into the C Token Latch is known as the

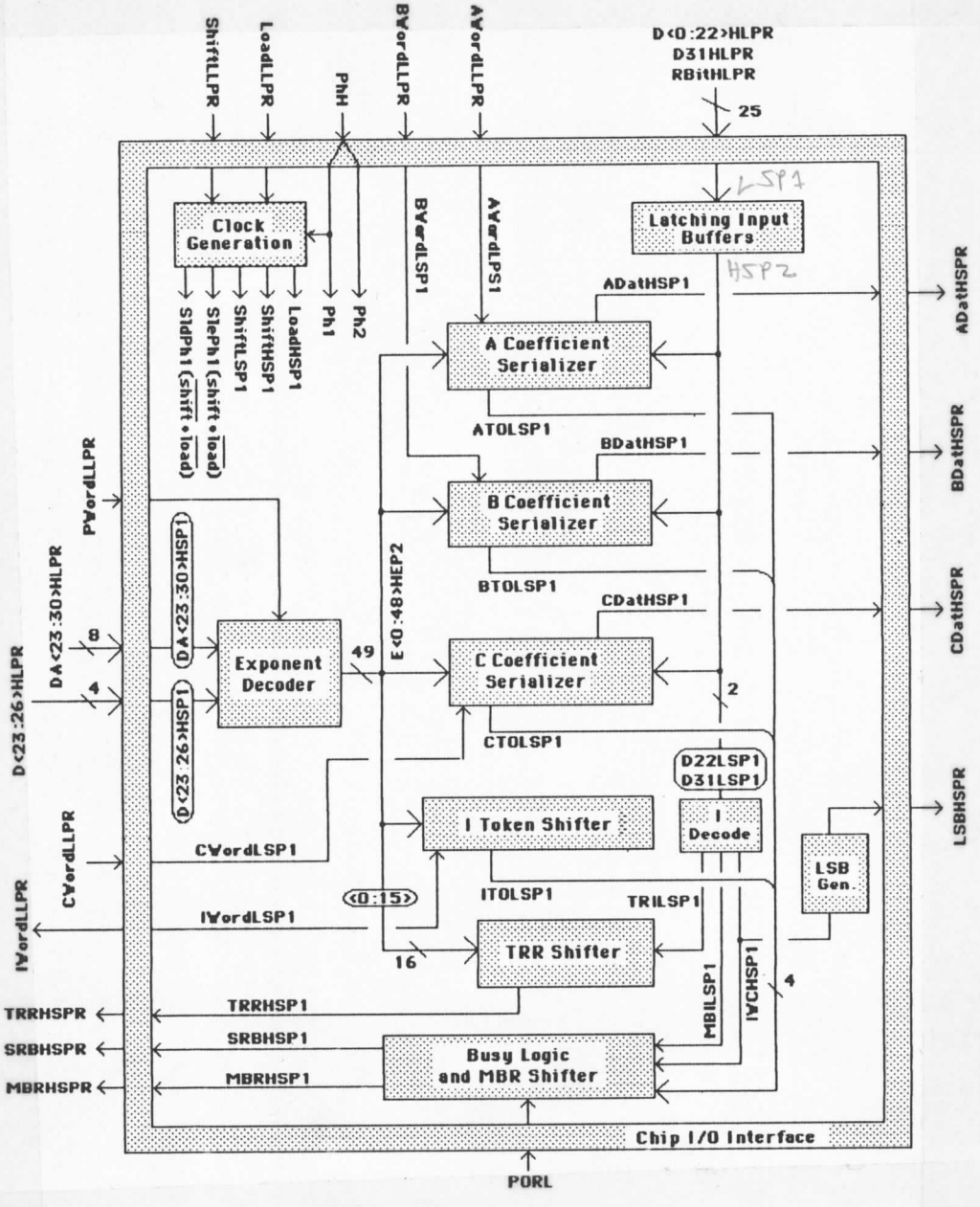


Figure 2.3.3: Block Diagram of Coefficient Serialization Chip
Routing for Ph1, Ph2, and the -Ph1 qualified clocks
is not shown since these signals go to virtually every circuit block.

figs: I Tok Sh
TRR Shifter
MBR Shifter
LSB Gn
I Decode

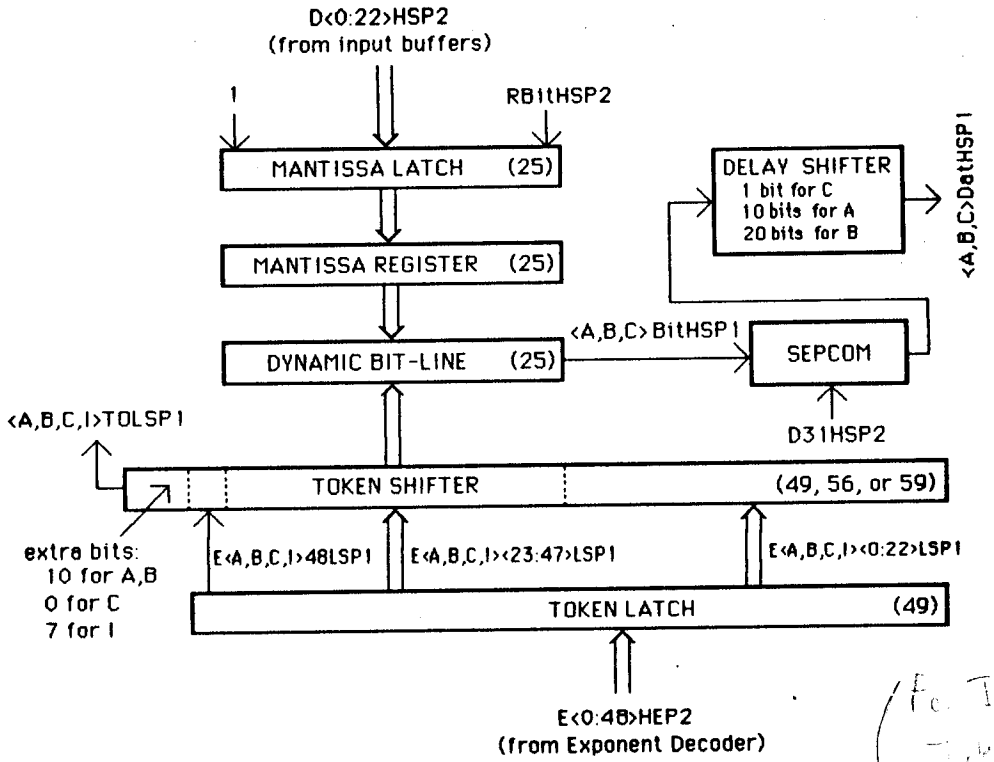


Figure 2.3.x: Coefficient Serializer

"token". When the instruction passes to the second stage of the pipe on LoadLLPR, LoadHPh1 is asserted on-chip and the mantissa is transferred from the C Mantissa Latch to the C Mantissa Register; simultaneously, the contents of the C Token Latch are transferred to the C Token Shifter (a parallel-loadable shift register). Bit-serialization of the coefficient then begins. A precharge/evaluate bit-line is used to produce the coefficient bit. This line is precharged hi on Ph1, and is then pulled lo on Ph2 provided that the token in the Token Shifter is positioned opposite a stage of the Mantissa Register which contains a 1.

The C Mantissa Register is 25 bits in length. The leftmost bit of the Mantissa Register contains a 1, the understood leading 1 of the mantissa for an IEEE floating-point standard number. The next 23 bits contain the actual mantissa bits from the IEEE format, and the rightmost bit contains RBit (which is 0 or 1 depending on which is appropriate to the rounding algorithm implemented by the floating-point engine which computes the coefficients). The C Token Shifter is 49 bits long. One bit of the Token Shifter lies to the left of the Mantissa Register and corresponds to the E48HEP2 output of the Exponent Decoder, 25 bits are aligned with the Mantissa Register and correspond to E<23:47>HEP2 (with E47HEP2 corresponding to the understood leading 1 and E23HEP2 corresponding to Rbit), and 23 bits of the Token Shifter lie to the right of the Mantissa Register and correspond to E<0:22>HEP2. If the coefficient is out of range, the 1-hot output of the Exponent Decoder is E48HEP2, and the token is placed in the Token Shifter at the bit that lies to the left of the Mantissa Register; the bit-line is never evaluated low and the coefficient is all 0's (since the bit-line is terminated by a latch and an inverter). If the coefficient is in range, 0's and 1's are produced as the token is shifted to the left

in the Token Shifter. When the token is shifted beyond the left-most bit of the Mantissa Register, only 0's are produced; the reflects the fact that sign-extension in a sign-magnitude representation is realized simply by adding trailing 0's. If the coefficient is in range, but is large enough that the exponent is greater than 24-FBITS, then the 1-hot output lies in the range E<0:22>HEP2 and the token is placed in the C Token Shifter to the right of the Mantissa Register; thus the bit-line is not evaluated lo on Ph2 and the first few bits of the coefficient will be 0's. This reflects the fact that precision is lost with very large numbers.

The precharge-evaluate bit-line is evaluated and latched on Ph2 and level restored by an inverter to produce the signal CBitHSP1. The stream of bits produced on CBitHSP1 is shifted through the module SepCom. A 1-bit parallel-loadable shift register introduces one clock cycle of delay and introduces a 0 separator bit at the beginning of the coefficient bit-stream; this 0 separator bit is required for correct operation of the multipliers on the smart memory chips. The bit stream then goes through a latch; this latch is cleared on LoadHPh1, and its output subsequently goes hi when the first 1 in the new coefficient's bit-stream appears. The output of this latch (CsoneHSP1) is NAND'ed with the sign-bit to produce the signal CinvLSP1, which is exclusive-OR'ed with the bit-stream going into the latch to produce CbdHSP1. This signal goes into a 1-bit Delay Shifter (a shift-register with no parallel input) which introduces an additional clock cycle of delay an produces CDatHSP1. This cycle of delay, plus the cycle introduced by the separator-bit injector, insure that the LSB signal preceeds the LSB of the C coefficient by 2 clock cycles, as required by the multipliers of the Pixel-planes memory chips.

The serialization of the A and B coefficients is performed by identical structures, except that the length of the Delay Shifter is different for the 3 coefficients; the number of bits of extension of the Token Shifter to the left of the Mantissa Registers also is different (as will be discussed in the next section). Nine additional clock cycles of delay must be applied to the A coefficient because the LSB of A must be introduced into the "side"inputs X serial multiplier on the same clock cycle as the LSB of the C coefficient is at the last bit of the X serial multiplier (assuming a 10 level multiplier). The B coefficient is delayed by an additional 10 clock cycles because the sequence of partial products reach the y-multiplier/tree, whose "side" inputs are B, 10 clock cycles after they enter the x-multiplier (whose "side" inputs are A). Thus, the Delay Shifters for C, A, and B are 1, 10, and 20 bits in length, respectively.

2.3.4.2.3. SRBH Logic and MBR Shifter

The Coefficient Serialization Module must generate the signal SRBHSPR, which indicates that the Coefficient Shifters are in the process of bit serializing a coefficient and must not be reloaded, and a related signal MRBHSPR, which indicates that the MSB or sign-bit of the tree result (Ax+By+C) has reached the ALU's of the Pixel-planes memory chips.

Theoretical Considerations

If the lengths of the A, B, and C coefficients (not including sign-bit) are defined as LenA, LenB, and LenC, respectively, then the length of the 2's complement form tree result Ax+By+C (including sign-bit) is given by

(1) Max (LenA + 10, LenB + 10, LenC) + 3.

This allows for A and B being multiplied by 10 bit numbers (x and y), two bits for overflow since 3 quantities are being summed, and an additional bit for the sign bit. The coefficients must therefore be sign-extended to this length, in order to form the full tree result.

However, for other instructions it may not be necessary to fully form the tree result; it may only be necessary to form a certain minimum number of bits of the tree result. (For example, an instruction which loads the tree result into some segment of the pixel memory representing R, G, or B intensity information would only need for the first 8 bits of the tree result to be formed, if the intensity buffer were 8 bits in length; clock cycles would be wasted if the tree result were computed in its entirety.) For other instructions it may be necessary to compute the full tree result, but it may also be necessary to sign extend the tree result to some minimum length, regardless of the magnitude of A, B, and C. Thus, it is in fact necessary to sign-extend the coefficients to the following length:

(2) FNI + MBI * [Max (LenA + 10, LenB + 10, LenC) + 3].

In this equation, FNI is a number and MBI is a flag, both of which are supplied with the basic command being executed, in the I word. FNI encodes the number of bits of the tree result which must be computed and is independent of the magnitude of the coefficients. MBI, when set, indicates that the tree result must be fully formed to its sign-bit. No conditional flag is associated with FNI, because FNI may simply be set to 0 for an instruction for which no fixed number of bits of the tree result need be formed.

Implementation

Associated with each Token Shifter is a latch, which is conditionally set on LoadHPh1 when the Token Shifters are loaded and coefficient serialization begins, and is cleared when the token for the coefficient reaches the left end of its Token Shifter (ATOLSP1, BTOLSP1, or CTOLSP1 goes low for one clock cycle). The outputs of the latches associated with each of the 3 Coefficient Serializers are ABsyHSP2, BBsyHSP2, and CBsyHSP2. To handle the FNI factor in the above equation, an additional I Token Latch and I Token Shifter are provided. The I Token Latch is loaded with the set of 1-hot Exponent Decoder outputs on IWordHPh1 (when an I word is sent to the IGC), in the same fashion as for the coefficient words; the token is transferred into the I Token Shifter on LoadHPh1. The placement of the token in the I Token Shifter is defined by the FNI field (bits 23-30) of the I word, just as for the A, B, and C words. Associated with the I Token Shifter is the I Busy Latch, with output IBsyHSP2. The outputs of the 4 Busy Latches are connected by a large NOR gate; it produces the signal SRBHSP1, which, when asserted, indicates that bits of the coefficients are being generated and new coefficients must not be loaded. The Busy Latches and the SRBHSP1 NOR gate are shown in Figure 2.3.y.

The A, B, and C Busy Latches are loaded with the value of the MBI flag of the I word, which indicates whether the tree result need not be fully formed (to the sign-bit); if MBI = 0, the signals ABsyHSP2, BBsyHSP2, and CBsyHSP2 never are asserted, and the number of bits of the coefficients produced is determined only by the I Busy Latch, and hence by the FNI field of the I word. Similarly, the I Busy Latch is loaded with the value of the CVI flag of the I word, which indicates whether the the instruction uses the A, B, and C coefficients. If CVI = 0, IBsyHSP2 is never asserted (nor are ABsyHSP2, BBsyHSP2, and CBsyHSP2) and thus the signal SRBHSP1 is never asserted.

The output SRBHSP1 is latched off-chip in the Interface and Control Module as SRBHLPR, and qualifies LoadLLPR which controls loading of the next set of coefficients. The timing of SRBHSP1 must be such that the coefficients are sign-extended to the correct length before bit-serialization of the next set of coefficients is initiated by LoadLLPR. To accomplish correct timing of SRBHSP1, the C Token Shifter is extended 1 bit to the left of the C Mantissa Register, and the Token Shifters for the A and B coefficients are extended 11 bits to the left of the leftmost bit of the Mantissa Registers (the Mantissa Registers for the 3 coefficients are all aligned). Thus, CTOLSP1 goes low one clock cycle after the MSB of the C coefficient (not including sign-bit)

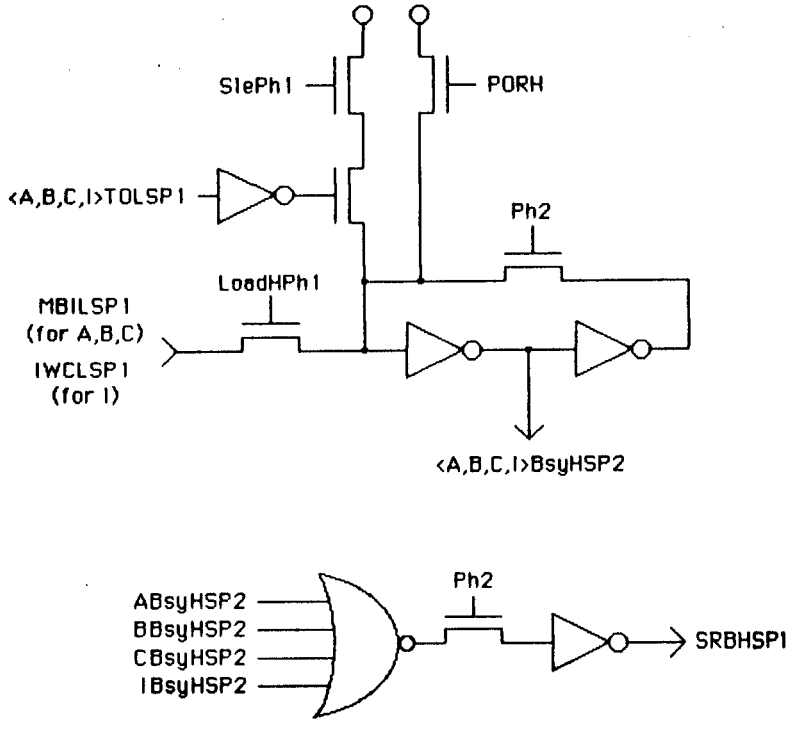


Figure 2.3.y: ~~Busy Latches and SRBHSP1~~ NOR Gate

appears on CBitHSP1 (the output of the inverter which terminates the dynamic bit-line); CTOLSP1 stays low for exactly one clock cycle, and this clears the CBusy Latch and CBsyHSP2 goes low. Similarly, ATOLSP1 and BTOLSP1 go low 11 clock cycles after the MSB of A and B are produced. After the last of the tokens produced on ATOLSP1, BTOLSP1, and CTOLSP1 "pops out", SRBHSP1 goes lo (ignoring the I Busy Latch for the moment); thus SRBHSP1 goes low 12 clock cycles after the MSB of A and B, or 2 clock cycles after the MSB of C, appear on ABitHSP1, BBitHSP1, and CBitHSP1. Assuming NirHLPF = 1 and IpLLPF = 1 off-chip in the Interface and Control Module, 2 additional clock cycles elapse before LoadHSP1 goes high on-chip again, and then on the third clock cycle after SRBHSP1 goes low the LSB's of the next set of coefficients appear on ABitHSP1, BBitHSP1, and CBitHSP1. Thus the A and B coefficients are sign-extended for at least 14 bits beyond the larger of the two, and C is sign-extended at least 4 bits. This accounts for the terms in Equation 1 as well as an additional clock cycle for the required separator bit. The timing for SRBH is shown in Figure 2.3.f.

The I Token Shifter is extended 8 bits to the left of the Mantissa registers. This ensures that at least 10 bits of the coefficients are produced regardless of the magnitudes of the coefficients or the FNI field of the I word (providing of course that the CVI flag of the I word is set). This is required for correct functioning of the multipliers in the Pixel-planes chips. In order to provide for at least FN bits of the tree result to be formed to the left of the radix point, the FNI field of the I word should be set to 127 + FN - 11; FN must not be larger than 58-FBITS. (The exponent adder adds FBITS+1 to this field, and thus DA<23:30>HLPR represent 128 +

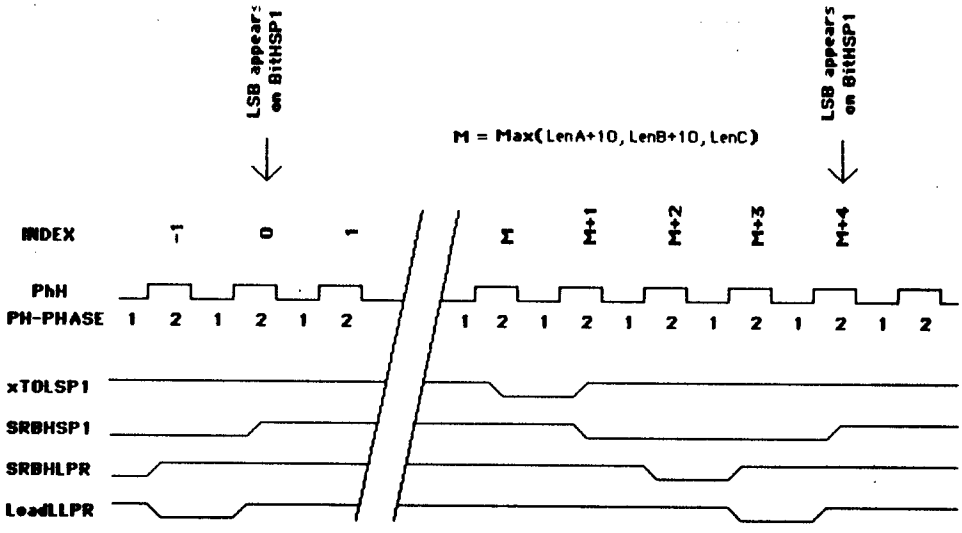


Figure 2.3.f: Timing for SRBH

"xTOLSP1"
refers to the last of the tokens to
"pop out".

FN' -11, where FN' = FN + FBITS is the number of bits of the tree result to be formed including the fractional bits. If FN' = 11, the token from decoding the FNI field of the I word will be placed in the bit of the I Token Shifter corresponding to the leftmost bit of the Mantissa Registers. The 8 bits of extension of the I Token Shifter, together with the 4 bits of delay between the token coming out the end of the shifter and the appearance of the separator bit, insure that 11 bits of the tree result are formed. If FN' is smaller than 11, DA<23:30>HLPR will be decoded as out-of-range, the token will be placed in the position of the I Token Shifter just to the left of the mantissa registers, and at least 10 bits of the tree result will be formed as required by the multipliers on the Pixel-planes chips).

MBR Shifter

The MBR Shifter is a 21-bit shift register, preceded by a special latch similar to the Busy Latches, as shown in Figure 2.3.zz. The MBR Input Latch is loaded with the value of the MBI flag when LoadHPh1 is asserted. If the MBI flag was set, MSInHSP1 is asserted for one clock cycle when SRBHSP1 goes lo, thus injecting a token into the left-hand end of the shift register. When this token appears at the right-hand end of the shifter, MBRHSP1 is asserted for 1 microcycle. The MBR Shifter is sized so that MBRHLPR (MBRHSPR latched off-chip) goes hi 1

microcycle prior to the MSB (sign-bit) of the tree result appearing at the "tree" input to the Pixel-planes ALU.

2.3.4.2.4. TRR Shifter

The TRR Shifter is a 35-bit shift register, the first 16 stages of which are parallel-loadable. When a new set of coefficients is transferred to the Mantissa Registers and Token Shifters on LoadHPh1, a token is loaded into the TRR Shifter. When this token appears at the right-hand end of the shifter, TRRHLPR is asserted for 1 microcycle. The placement of the token is such that TRRHLPR (TRRHSP1 latched off-chip) goes hi 2 microcycles prior to the LSB of the whole part of the tree result appearing at the "tree" input to the Pixel-planes ALU.

The correct position of the token is determined by the number of fractional bits in the bit-serialized coefficients (FBITS), which is specified using a P word. The Exponent Decoder has multiplexed inputs, so that when a P word is being loaded the inputs are taken from D<23:26>HSP1 rather than from DA<23:30>HSP1; the inputs corresponding to DA<27:30>HSP1 are set so that a 1-hot output is produced on Exponent Decoder outputs E<0:15>HEP2. This output is independent of the previous value of FBITS since D<23:26>HLPR bypass the off-chip exponent adder. This 1-hot mask is latched into the P Token Latch, where it is subsequently avaialble to be transferred into the TRR Shifter whenever LoadHPh1 is asserted.

TRRHLPR must preceed the arrival of the LSB of the whole part of the tree result at the ALU by 2 clock cycles for instructions for which contents of memory must be arithmetically combined with the tree result; this is because one-cycle advance warning is needed if the first bit of memory is to be fetched in time to arrive at the ALU on the same microcycle on which the LSB of the tree result arrives.

For instructions which have no coefficients, such as "set enables", the CVI bit of the I word would be set to 0. This CVI bit is latched on IWordHPh2 and used to qualify the signals going from the TRR Token Latch to the TRR Shifter, so that all 0's are transferred into the TRR Shifter if CVI=0. Thus the TRRHLPR signal is not generated for these instructions.

2.3.4.2.5. Specification of I Word Fields for Coefficients

Instructions may be divided into 4 types, according to the way in which the coefficients are handled. For each type of instruction the CVI and MBI flags, as well as the FNI field, of the I word must be properly specified, as follows:

(1) the instruction has no coefficients

CVI = 0
MBI = 0
FNI field = 0

thus none of the 4 Busy Latches is set, and SRBHSP1 is never asserted

(2) the tree result must be of a certain length (FN bits to the left of the radix point), but need not be fully formed (extended to its sign-bit)

CVI = 1
MBI = 0
FNI field = 127 + FN - 11

(3) only the sign-bit of the tree result is important

CVI = 1
MBI = 1
FNI field = 127 - 9

It is necessary to insure that MBRHLPR occurs no earlier than 2 microcycles after TRRHLPR, to avoid confusion in the microcode logic. Since TRRHLPR occurs 1 microcycle before the most significant fractional bit of the tree result appears at the ALU and MBRHLPR occurs 1 microcycle before the sign-bit of the tree result appears at the ALU, the tree result must be formed 2 bits to the left of the radix point.

(4) the sign-bit of the tree result is important, but the tree result must be of a certain minimum length in any case, specified as FN bits to the left of the radix point

CVI = 1
MBI = 1
FNI field = 127 + FN - 11

This generally applies when the tree result must be combined with a segment of memory of some fixed length, and so the tree result should be sign-extended to the length of the memory segment. A loop count is used until the computation has preceeded to the length of the memory segment, the "memory" input to the pixel-ALU is then put to 0, and the computation continues until the MSB of the tree result arrives. FN should be selected to insure that MBRHLPR occurs after the loop counter underflows so that the MBRHLPR signal is not missed by the microcode logic. For example, for the "disk define" instruction, this means that MBRHLPR must occur no earlier than QLEN + 2 cycles after TRRHLPR (where QLEN is the length of the Q-buffer). Setting FN = QLEN + 2 insures this.

2.3.4.2.6. Control of Coefficient Serialization Module

The signals AWordLLPR, BWordLLPR, CWordLLPR, and IWordLLPR control loading of the A, B, C, and I Token and Mantissa Latches, as described above. LoadLLPR controls loading of the coefficients in the Mantissa Registers and Token Shifters, placing of a token into the TRR Shifter, and setting of the Busy Latches, as described above. An additional control signal, ShiftHLPR, is provided indirectly by the current word of microcode memory, via the Shift Generation Logic. Shifting of the Token Shifters (and the other components of the coefficients' serial data path), the TRR and MBR Shifters, and the LSB Generator, are controlled by ShiftHLPR; these shift structures are shifted on a given microcycle only if ShiftHLPR = 1. ShiftHLPR also is an input to the Pixel-planes chips, controlling the operation of the serial multipliers and multiplier tree. For any microcycle for which ShiftHLPR = 1, the multiplier/tree structures of the Pixel-planes chips are "stepped", and the various shifters within the CSM are shifted. Because image-generation signals are latched into the Pixel-planes chips on Ph1, and because the shifters of the CSM utilize full 2-phase clocking, the inputs to the Pixel-planes chips effectively change at the END of the microcycle for which ShiftHLPR = 1.

ShiftHLPR is produced by the Shift Generation Logic, under control of the TS and TSF bits of the current microcode word. If TSFHLPR is hi, ShiftHLPR is forced hi unconditionally. If TS is hi but TSF is lo, ShiftHLPR is hi provided that if either of TRRHLPR or MBRHLPR is hi on the given clock cycle, that the advance control bits of the microcode word (AC0-AC2) are set to test for that condition. Thus, microcode words for an instruction which does not use coefficients might have TS=1 and TSF=0, thereby allowing the coefficient serializers and multipliers to perform computations using the coefficients for the next instruction while the current instruction is being executed; shifting of the coefficient serializers and multipliers would automatically halt when the LSB of the tree result appears.

2.3.4.3. Microcode Memory Module

The microcode memory is 32 bits wide and will have a maximum length of 256 words. (Presently only 30 bits of the microcode word are used). The 8 address inputs to the microcode memory are latched on the falling edge of PhH (or, equivalently, the rising edge of PhL) and the outputs are latched on the rising edge of PhH.

The bits of the microcode word are specified as shown in Table 2.3.2.

Since the microcode address generally remains fixed when waiting for the least or most significant bit of the tree result to reach the ALU (indicated by TRRHLPR or MBRHLPR), the number of words in the microcode memory required for a given instruction usually is much less than the actual number of clock cycles in an instruction.

2.3.4.3.1. Microcode Address Sequencing

Flow control is achieved by means of a 4-to-1 mux from which the Microcode Address Latch (program counter for the microcode engine) is loaded; this mux is controlled by the Microcode Address Select Logic according to the state of certain signals within the IGC as defined by the "advance control" bits (AC2-AC0) of the microcode word. The four selectable inputs are:

Table 2.3.2: IGC Microcode Word Configuration

BITS	FUNCTION
0-9	the 10 ALU control bits, direct inputs to the smart memory chips
10-11	MWrt0 and MWrt1: direct inputs to the smart memory chips
12-14	ASW, AIP, and ACT: control operation of the counters and multiplexer within the Pixel Memory Address Sub-Module, which supply the signals Add6 - Add0 for the smart memory chips
15-16	Cnt1 and Cnt2: enable counting in the two loop counters
17-18	TS and TSF: utilized by the Shift Generation Logic to produce ShiftHLPR; ShiftHLPR becomes the TrStH input to the smart memory chips, and also controls operation of the shifters within the Coefficient Serialization Module
19-21	AC0 - AC2, the advance control bits: the settting of these bits for the current microcode word, along with the values of the signals MBRHLPR, TRRHLPR, VRBHLPR, IpHLPF, TLC1HLPF, and TLC2HLPF, define the next microcode word to be executed; this control is implemented by a logic block (the Microcode Address Select Logic) that defines which input to the 4-to-1 Microcode Address MUX is loaded into the Microcode Address Latch
22-23	reserved for future use
24-31	MAF0 - MAF7: microcode branch address, bit 24 is LSB

(1) the previous address plus 1, which causes an advance to the next microcode word

(2) the address contained in the branch address field (MAF) of the current microcode word (MAF usually is set to point to the current word, causing the word to repeatedly be executed until a given condition is met; MAF might also point to a lower address to cause looping until some condition is met)

(3) the starting address for the next instruction to be executed

(4) an address which defines the location at which a word of microcode is to be loaded; this input is used only during the loading of the microcode store, not during normal operation.

The Microcode Address Select Logic is an unclocked combinational logic network. Its inputs are the advance control bits of the microcode word, AC0 - AC2, the control signals DWordLLPR, LWordLLPR, and IpHLPF from the Interface Module, the signals TRRHLPR and MBRHLPR latched off the Coefficient Serialization Module, the underflow signals TLC1HLPF and TLC2HLPF from the Loop Counters, and the signal VRBHLPR (begin of video retrace) which is produced by synchronizing VRBHLXF from the Video Controller. The advance control bits of the microcode word are decoded as shown inTable 2.3.3.

The last word in the microcode sequence which executes some instruction, as well as the microcode word at address 0 (the "idle" state), should have the AC bits set for the special "done" state (AC2 = AC1 = AC0 = 0) and the branch field set to address 0. Thus when an instruction is completed, if a new instruction is pending (IpHLPF = 1), control jumps directly to the first microcode word of the new (pending) instruction; otherwise, control jumps to the "idle"state, where it remains until IpHLPF goes hi.

Table 2.3.3: Advance Control Codes for Microcode Sequencing

AC2	AC1	AC0	ACTION
0	0	0	the "done" state: load starting microcode address if IpHLPF = 1, otherwise branch to MAF
0	0	1	increment address unconditionally
0	1	0	increment address if TLC1HLPF = 1 (loop counter 1 has underflowed), otherwise branch to the address specified by the microcode branch address (MAF) of the microcode word
0	1	1	increment address if TLC2HLPF = 1 (loop counter 2 has underflowed), otherwise branch to MAF
1	0	0	increment address if VRBHLPR = 1 (at begin of video retrace), otherwise branch to MAF
1	0	1	branch to address represented by MAF field of microcode word
1	1	0	increment address if TRRHLPR = 1, otherwise branch
1	1	1	increment address if MBRHLPR = 1, otherwise branch

Table 2.3.4 - T... Table f... Manual A. S.L

2.3.4.3.2. Pixel-Memory Address Control

The pixel-memory address inputs to the Pixel-planes memory chips are supplied by the Pixel-Memory Address Sub-Module of the IGC, which is controlled by the AIP, ASW, and ACT bits of the microcode word. The outputs of this sub-module are selected from a 3-to-1 mux, which has as inputs the Refresh Counter, the Source Address Counter, and the Destination Address Counter. If AIP = 0, indicating that the address inputs may be arbitrary for the next microcycle, then Add6 - Add0 are selected from the Refresh Address Counter, and it is automatically decremented on the next Ph1. When AIP = 1, indicating that a specific address must be applied to the address inputs of the Pixel-Planes chips on the next clock cycle in order for the instruction to be properly executed (as with portions of instructions which write to Pixel-Planes memory or read the Q-buffer or z-buffer), then Add<6:0> are selected from either the Source Address Counter (if ASW = 1) or the Destination Address Counter (if ASW = 0). If ACT = 1, then the selected counter is incremented on the next ph1; ACT has effect only if AIP = 1, since the refresh counter always is incremented when AIP = 0.

The Source and Destination Address Counters are loaded from their respective fields of the I word at the beginning of execution of the microcode sequence for the instruction. Thus two sequences of Pixel-planes memory addresses, defined by these fields, may be specified for a given instruction. Loading of the Source and Destination Address Counters is enabled by DoneHLPR, since the microcode word preceding the beginning of any instruction always is either the microcode word at address 0 (the "idle state") or the final microcode word of some other instruction, and thus always has bits AC2 - AC0 set for the "done"

Pixel-planes Memory and Shadow Register Refresh

The purpose of the Refresh Counter is to insure that each address is asserted at least once per scan line, so that the "shadow register" will be properly loaded and the memory adequately refreshed. The Refresh Counter is loaded from a REFCNT Register, whenever it has counted down to address 0. The REFCNT Register is accessed by transmitting a P word. The contents of bits 26-30 of the P word are loaded into the REFCNT register. The Refesh Counter cycles through addresses 0 to 8*REFCNT + 7, where REFCNT is represented by bits 26-30 of the P word with bit 26 as LSB. state.

Assuming a scanline time of 30 us and a phi clock rate of 10 MHz, 300 microcycles are executed per scanline; if REFCNT is specified for automatic refresh on bits 0-71, and if AIP = 0 for at least 72 of the 300 microcycles, then filling of the shadow register on every scan-line is assured. However, only those pixel-memory addresses used to store color intensity values for display need be visited on each and every scanline. Other memory locations, such as those used for building new images in double-buffering, computations in anti-aliasing and tecturing, the z-buffer, and the Q-buffer, need be visited only often enough to refresh their contents, perhaps of the order of once every 10 scan lines (conservatively). Thus REFCNT might be specified to provide automatic refresh for just the "visible" bits, provided they lie at the low end of the pixel-memory address space; the other pixel-memory addresses (used for z-buffer, Q-buffer, double buffering) would be refreshed "manually", just in the process of executing algorithms.

2.3.4.4. Loading the Microcode Store

The microcode store must be loaded at system initialization time, and may be altered or reloaded "on the fly" while the system is running. When loading microcode on the fly, the input device may either wait for DoneHLPR to remain hi for 3 consecutive clock cycles, indicating that the IGC has entered the "idle"state, or interrupt the IGC and risk possibly not executing or

erroneously executing instructions which are "in the pipe". Reloading of the microcode could also be used to reset a "hung" IGC.

Loading of the microcode store is performed by writing D and L type words to the IGC. The microcode store may be loaded in two modes: normal mode, in which one L word and one D word is required per memory location to be loaded, and fast mode, in which just one D word is required per memory location. For fast mode, the memory locations to be loaded must be a contiguous sequence of addresses, and the D words must be written on successive clock cycles. The pre-processor program "prep" gives examples of how the microcode store is loaded, in both modes.

2.3.5. Programming

Microcoding of the IGC is done using the Pixel-Planes Assembler "asmpp"; "asmpp" reads standard input and creates output in 2 files, "code" and "coms". The file "code" contains the 32-bit microcode words to be loaded into the Microcode Memory Store of the IGC at initialization time, and the file "coms" identifies instruction mnemonics with specific P and I words and microcode scraps.

The input to "asmpp" specifies microcode words for the Microcode Memory Store, on a one-to-one basis between input lines and microcode words, and also describes basic commands for the IGC as sequences of these microcode words along with specific P and I words which should be sent to implement these commands. Lines in the input file may be of 2 types: lines beginning with "#" are used to define basic commands for the IGC in terms of a scrap of microcode, while all other lines define a word of microcode (with a 1-1 correspondence between input lines and microcode words). Any portion of a line lying to the right of a semi-colon (";") is treated as a comment. Lines used to define microcode words consist of one or more of the following tokens:

done - used to indicate the end of an instruction, causes the following action: jump to the first word of the next instruction if IpHLPF = 1 (a new instruction is pending), else jump to address 0 (the "idle" state)

MBR:+/-N - advance to the next microcode memory address if MBRHLPR = 1, else jump to the specified offset from the current address

TRR:+/-N - advance to the next microcode memory address if TRRHLPR = 1, else jump to the specified offset from the current address

VR:+/-N - advance to the next microcode memory address if VRBHLPR = 1, else jump to the specified offset from the current address

TC1:+/-N - advance to the next microcode memory address if TLC1HLPF = 1 (loop counter #1 has underflowed), else jump to the specified offset from the current address

TC2:+/-N - advance to the next microcode memory address if TLC2HLPF = 1 (loop counter #2 has underflowed), else jump to the specified offset from the current address

src - apply the current contents of the Source Address Counter to the Pixel-planes memory address inputs Add6 - Add0, but do not increment the counter

src+ - apply the current contents of the Source Address Counter to the Pixel-planes memory address inputs Add6 - Add0, and then increment the counter

dst - apply the current contents of the Destination Address Counter to the Pixel-planes memory address inputs Add6 - Add0, but do not increment the counter

dst+ - apply the current contents of the Destination Address Counter to the Pixel-planes memory address inputs Add6 - Add0, and then increment the counter

cnt1 - decrement Loop Counter 1

cnt2 - decrement Loop Counter 2

WRT - set the MWrt0 and MWrt1 inputs to the Pixel-planes chips to cause memory writes for all pixels for which the Enable register is 1

FWRT - set the MWrt0 and MWrt1 inputs to the Pixel-planes chips to cause memory writes for all pixels, regardless of the settings of the Enable registers

LdEn - set the LdEn input to Pixel-planes chips to 1 (this causes the Enable register for each memory row within the array of Pixel-Planes chips to be loaded with the Carry output of the corresponding ALU bit)

ain:set - set the ALU control inputs to the Pixel-planes chips to provide the input specified by "set"to "a"input where "set"is "0", "1", "tree", "treebar", "sum", "sumbar", "sumtree"(sum "sumtreebar"(!(sum "sum"

bin:set - set the ALU control inputs to the Pixel-planes chips to provide the input specified by "set"to "b"input where "set"is "0", "1", "enable", "enablebar", "rddat", "rddatbar", "ear"(enable "earbar"(!(enable "0"

cin:set - set the ALU control inputs to the Pixel-planes chips to provide the input specified by "set"to "c"input where "set"is "0", "1", "cry", "crybar"; the default setting is "0"

CRY - set the ALU control inputs to the Pixel-planes chips to update the Carry registers of the Pixel-Planes chips' ALU's with the carry outputs of the ALU adder; otherwise the Carry register is saved

TS - step the Pixel-planes multiplier/tree structures and shift the Coefficient Serialization Module, unconditionally; that is, force ShiftHLPR = 1

TSC - step the Pixel-planes multiplier/tree structures and shift the Coefficient Serialization Module, conditionally; that is, ShiftHLPR = 1 provided neither TRRHLPR or MBRHLPR has occurred and is not being tested for by a "TRR:x"or "MBR:x"field

Lines used to define specific IGC commands (starting with "#") are placed in the input immediately before the sequence of microcode words used to implement the command. For example, a sequence of microcode words may be defined which causes the tree result to be loaded into a sequence of Pixel-Planes memory; this same code sequence can be used to define specific commands to load the red, green, and blue intensity buffers when painting, or to load the z-buffer. These specific commands are specified by defining different values for "StDstAddr"and "LpCnt1" in the input file, according to the lengths and locations of the red, green, blue, and Z buffers. A specific command is identified with the mnemonic "str", in a line beginning with "#str"; the remainder of the line may specify the following:

StDstAddr:N - the value to be loaded into the Destination Address Counter of the IGC

StSrcAddr:N - the value to be loaded into the Source Address Counter of the IGC

LpCnt1:N - the value to be loaded into Loop Counter 1 of the IGC

LpCnt2:N - the value to be loaded into Loop Counter 2 of the IGC

MBI - indicating that the tree result must be fully computed for this instruction, hence the MBI bit of the I word should be set to 1

FNI:N - the number of bits of the coefficients (to the left of the radix point) that must be transmitted for this instruction; FNI must be specified, else the instruction is assumed to not use coefficients

All of the above fields are optional; if neither StSrcAddr or LpCnt2 is specified, then no P word is sent. If FNI is not specified, then the instruction is assumed to be one which does not use coefficients and no A, B or C words are sent.

Some important considerations when writing microcode for the IGC:

(1) when none of ain, bin, cin, LdEn, or CRY is specified for a microcode word, the Pixel-planes chips' ALU's are effectively in a "no-op" state, since neither the Carry register or Enable register is loaded with a new value, and the Sum register is loaded with the "sum" input (since the default values for ain, bin, and cin are "sum", "0", and "0"respectively)

(2) microcode word 0 must be specified as follows: done ain:sum bin:0 cin:0 TS(or TSC)

(3) when implementing a loop using the tokens TRR:, MBR:, TC1:, or TC2:, with a negative offset, the loop must contain a microcode word with "TS" specified ("cnt1" or "cnt2" for TC1: or TC2:); otherwise a complete halt of the IGC probably would occur

(4) every instruction which uses coefficients must test for TRRHLPR at some point

(5) every instruction which has the MBI bit set in the I word must test for MBRHLPR at some point

(6) it is important to realize that whenever TS or TSC is set TRRHLPR or MBRHLPR may be asserted, but will only be asserted for 1 clock cycle; if TSC rather than TS is specified, the Shift Generation Logic will disable ShiftHLPR if either TRRHLPR or MBRHLPR occurs and is not being tested for; care must be taken to insure that TS is not set if TRRHLPR or MRBHLPR might occur without being detected; in particular, the microcode word which specifies ALU controls, etc for the clock cycle when the MSB of the tree result (or the last bit of the tree result to be used) is at the ALU, must not have TS set. TRRHLPR is asserted 2 clock cycles prior to the cycle at which the LSB of the integer part of the tree result appears at the "tree"inputs MBRHLPR is asserted 1 clock cycle prior to the cycle at which the MSB (sign-bit) of the tree result appears at the "tree"inputs If FBITS = 0, the LSB of the whole part of the tree result can be just 2 clock cycles behind the MSB of the previous tree result, not immediately behind, because of the separator bit. Thus, TRRHLPR can occur as soon as one clock cycle after MBRHLPR.

(This is seen in the microcode for the "red"instruction, in which the next to the last as well as the last word both have TSC specified; if the next to the last word had TS specified (rather than TSC). If FBITS=0, then TRRHLPR could be asserted on the clock cycle on which the most significant bit of the tree result to be used is at the input to the ALU, which is under the control of the next to last microcode word, ShiftLLPR would not be inhibited, and the system would be hung.)

(7) any instruction which generates two or more of TLCxHLPF, TRRHLPR, or MBRHLPR, must insure that they occur in the following order: TRRHLPR, TLCHxLPF, MBRHLPR

2.3.6. Simulator

The simulator for the IGC is "igc". each line of input represents one input word and contains 3 fields specified as follows:

(1) the number of IGC microcycles to wait before signaling that the input word is ready (this would normally be set to 0)

(2) an integer representing the type bits T2, T1, T0 (T2 is MSB, T0 is LSB)

(3) a 32 bit integer representing the data bits D31- D0 (D31 is MSB)

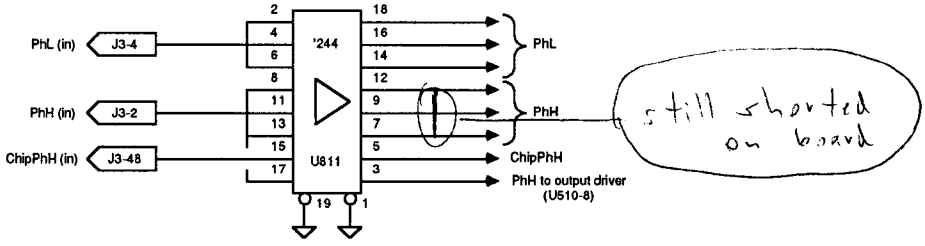
Output is placed on standard output and consists of lines of 0's and 1's representing the image generation vector (IGV, the signals to the Pixel-Planes chips), plus some additional bits representing the state of the IGC. The form of this output is compatible with the input format of the simulator for the array of Pixel-Planes memory chips ("chips" or "chipq").

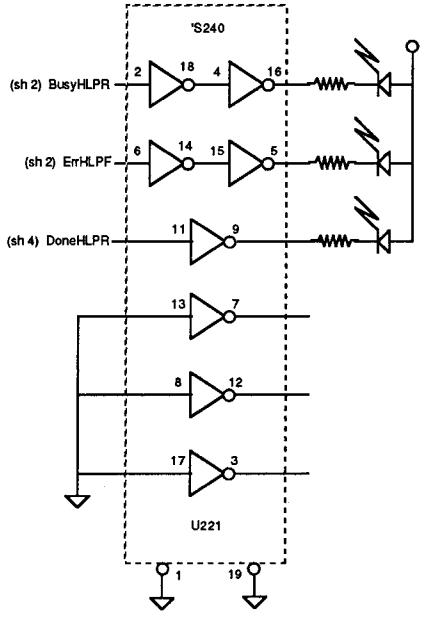
The "igc" simulator has 2 command line arguments. The first argument is specified as 1 to output the IGV for all microcycles, including those produced during the initial microocode loading sequence; if 0, the default, is specified, IGV's are not output until loading of the microcode store has completed, thus saving clock cycles in the chip array simulator. The second argument specifies how many clock cycles after the beginning of the simulation VRBHLPR should go hi; this is used to test the "VR:x" logic.

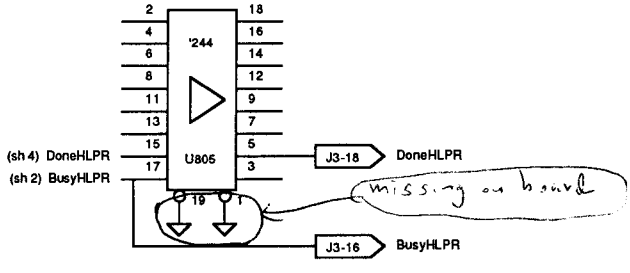
The simulator "igc" normally is used with the preprocessor "prep". The program "prep" uses the 2 files "code"and "coms" from the assembler "asmpp". "Prep" sends the correct D and L words to "igc" to load the Microcode Memory Store with the microcode words in "code". Then "prep" reads instructions and coefficients from standard input and sends the correct I, P, A, B, and C words to "igc". Input lines to "prep" consist of either 1 or 4 fields: the first field is an instruction mnemonic, and the remaining fields represent the A, B, and C coefficients, respectively, for instructions which have coefficients. The instruction mnemonic is looked up in "coms" (it should have been specified by a line beginning with "#" in the input file to "asmpp"), and the corresponding P and I words along with A, B, and C words representing the coefficients are output to the IGC. If any of the coefficients are not to be transmitted, a dash ('-') should be used for that field of the command. Input to "prep" may also contain lines of the form "FBITS N". This causes the number of fractional bits in the fixed-point numbers output from the Coefficient Serialization Chip to be changed to N (0 to 15). Note that if FBITS is changed, all coefficients must be transmitted on the next instruction, even if they are the same as for the previous instruction. FBITS is automatically set to 12 when "prep" starts up.

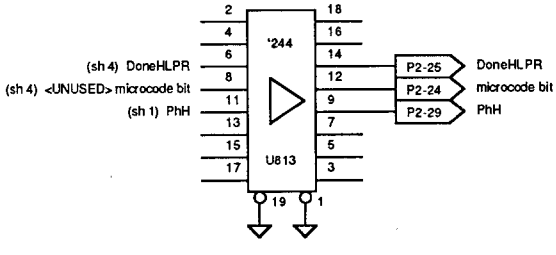
A typical simulation using the IGC and chip simulators would be run as follows:

```
prep < test.in | igc | chipq > test.out
```



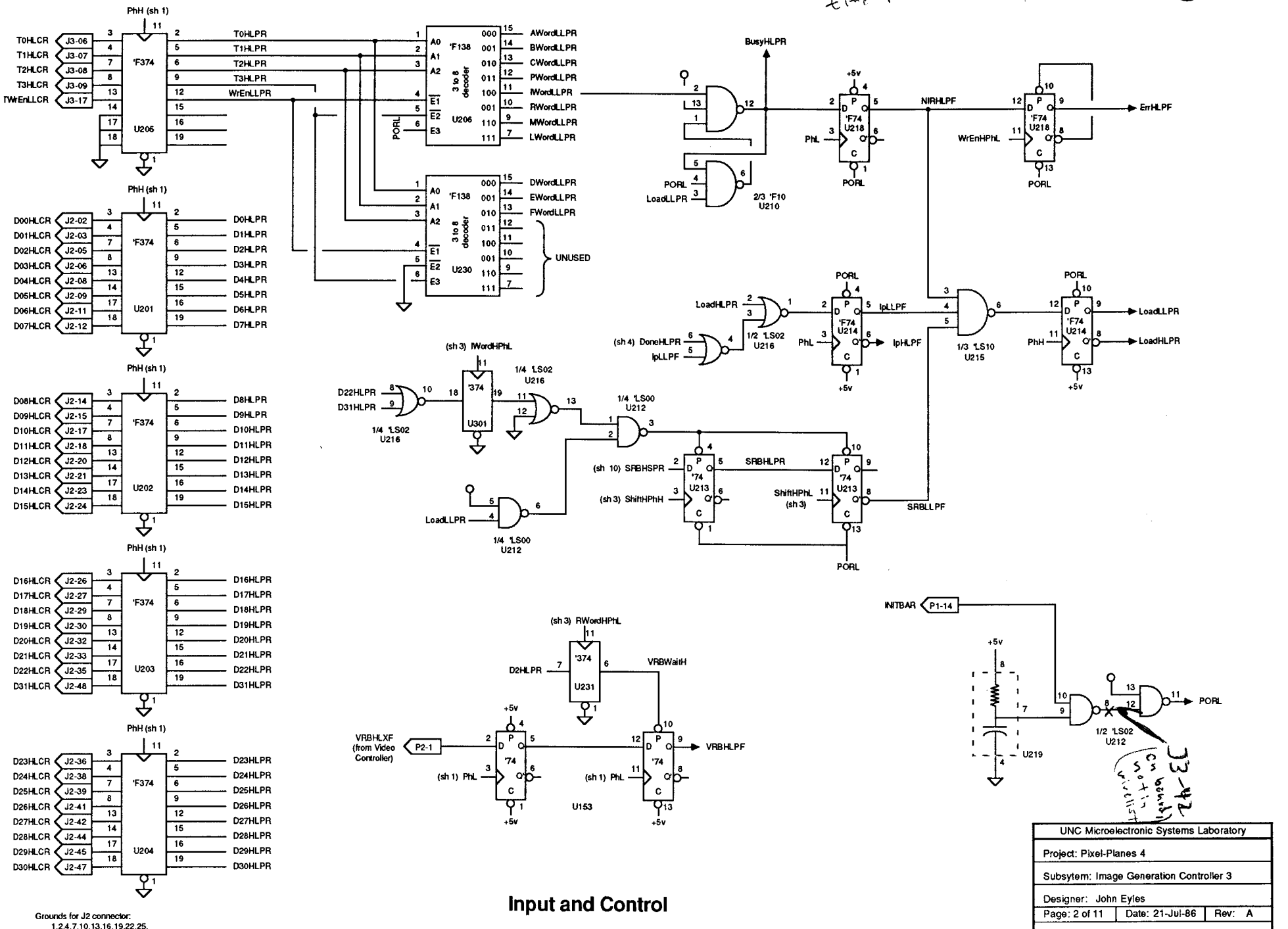






ZIP-BUS Ground Connections:
J2-1,4,7,10,13,16,19,22,25,
28,31,34,37,40,43,46,49
J3-1,3,5,10,15,20,24,
31,36,41,44,47,50

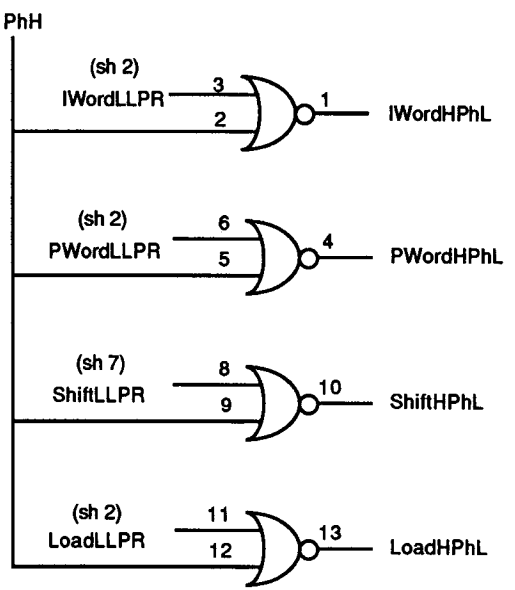
UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 1 of 11	Date: 11-Jul-86	Rev: A
Comments:		



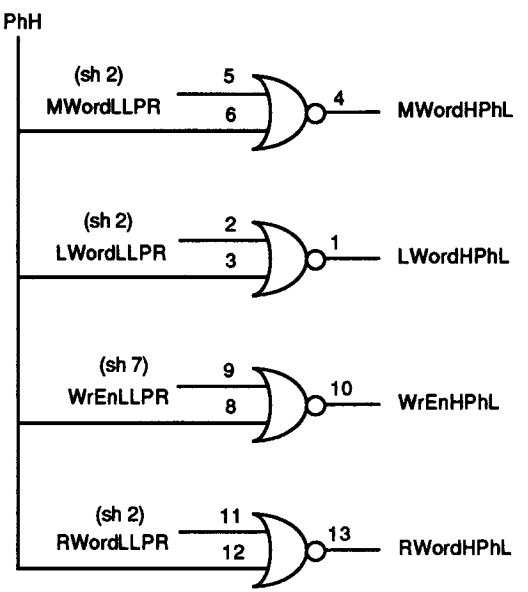
time PhH ↑ to BusyHLPH ↓ < 2 ns

Input and Control

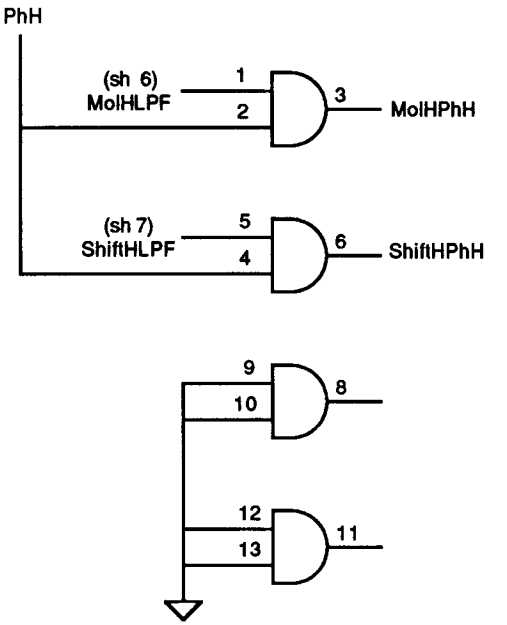
UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 2 of 11	Date: 21-Jul-86	Rev: A
Comments:		



'LS02
U196

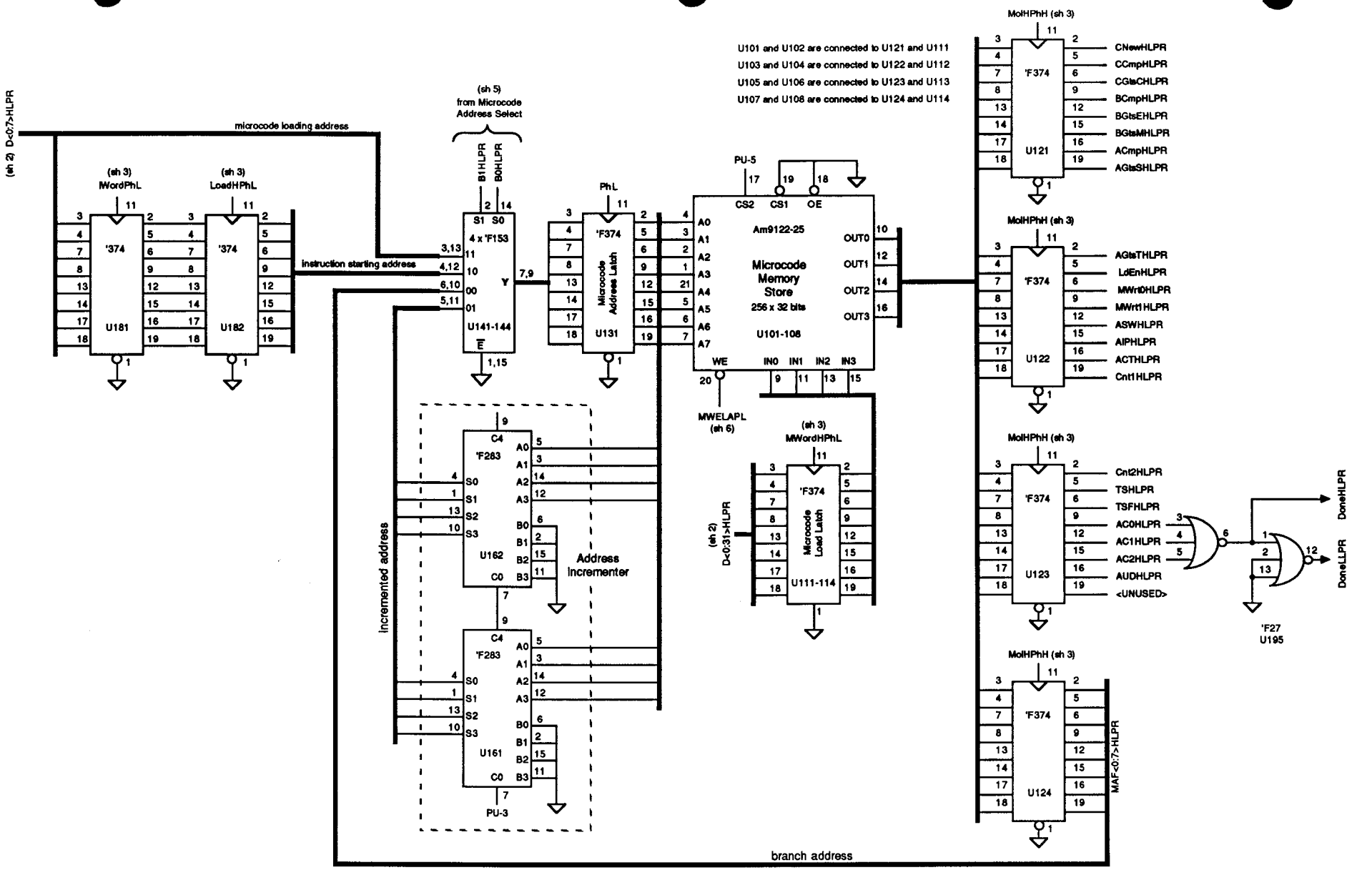


'F02
U194



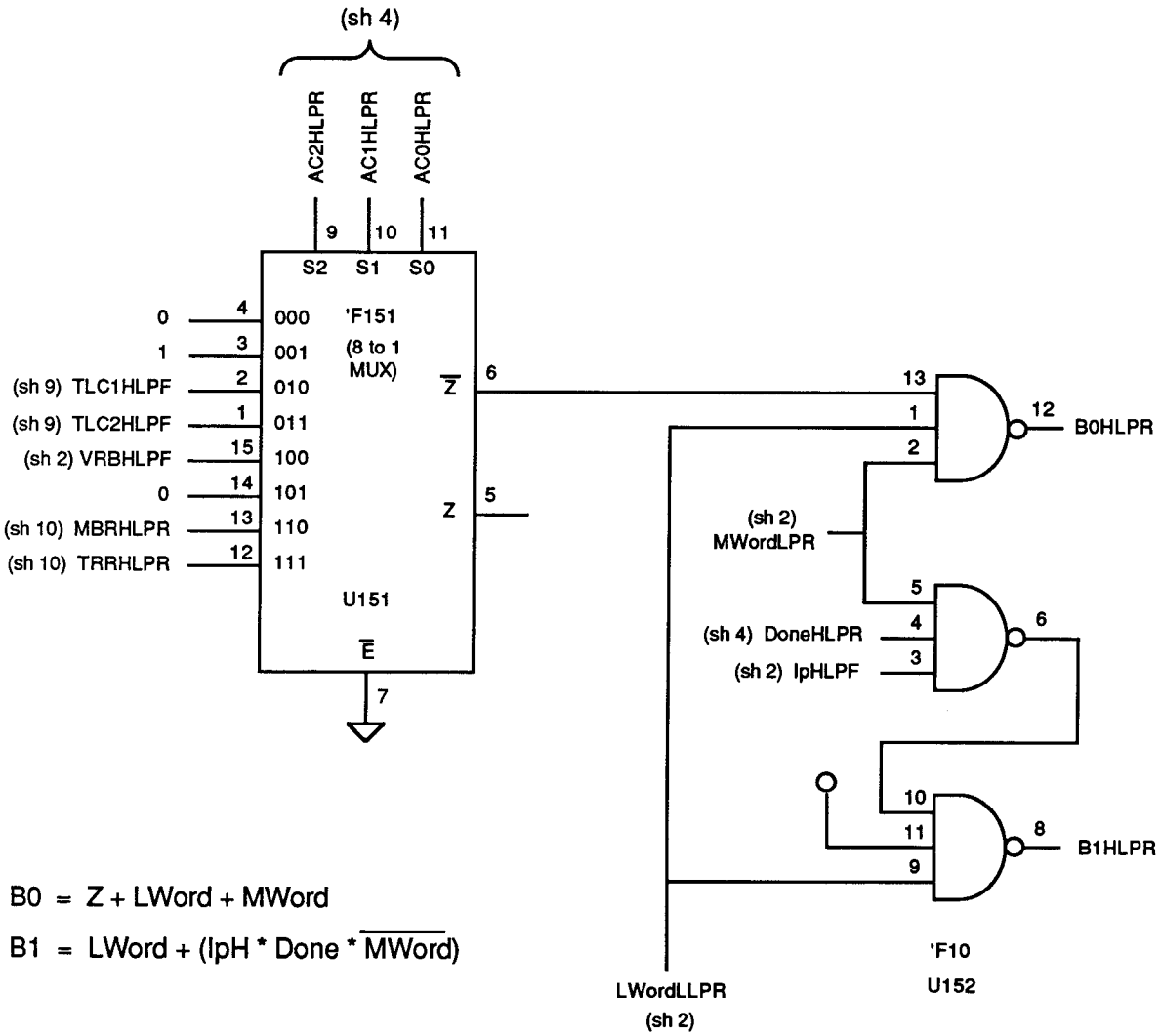
'F08
U193

UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 3 of 11	Date: 21-Jul-86	Rev: A
Comments:		



Microcode Engine

UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 4 of 11	Date: 21-Jul-86	Rev: A
Comments:		



B0 = Z + LWord + MWord

B1 = LWord + (IpH * Done * $\overline{\text{MWord}}$)

B1HLPR	B0HLPR	selected input to address mux
1	1	microcode loading address
1	0	instruction starting address
0	1	incremented address
0	0	branch address

LWord

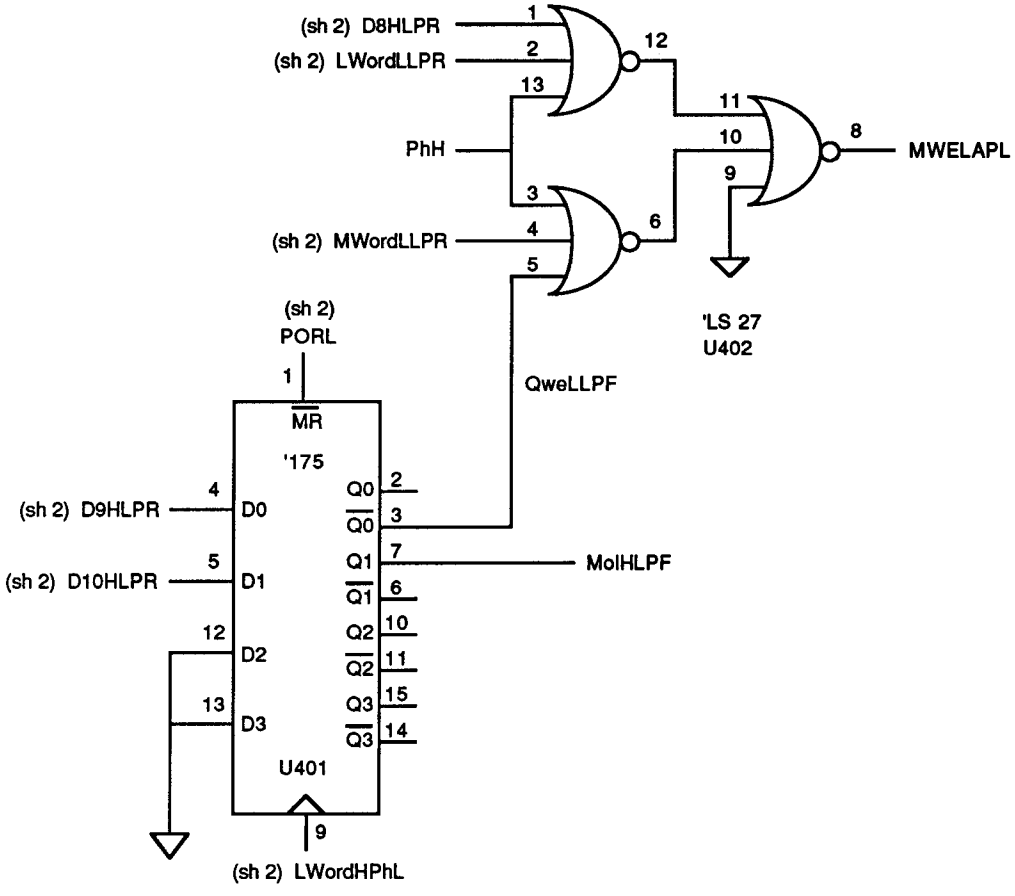
$\overline{\text{LWord}}$ · $\overline{\text{MWord}}$ · DoneHLPR · IpHLPF

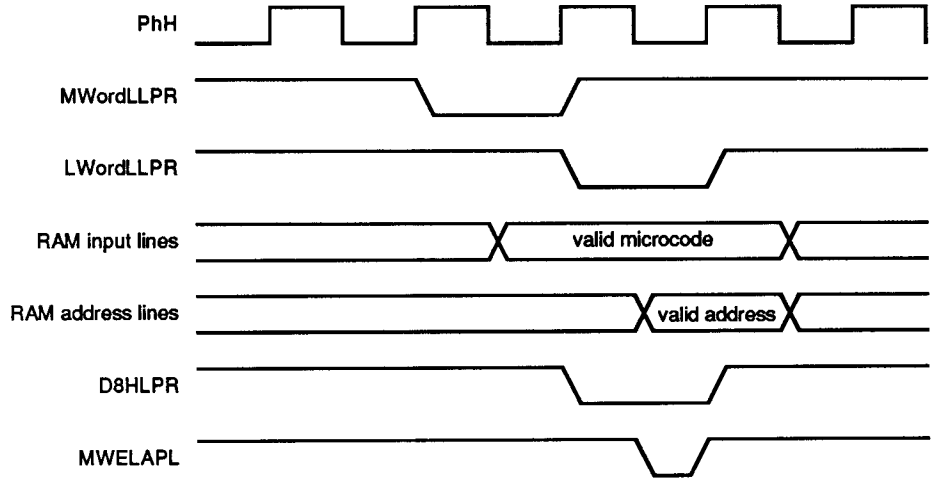
MWord + $\overline{\text{LWord}}$ · $\overline{\text{MWord}}$ · (

$\overline{\text{MWord}}$ · $\overline{\text{LWord}}$ · (

Microcode Address Select Logic

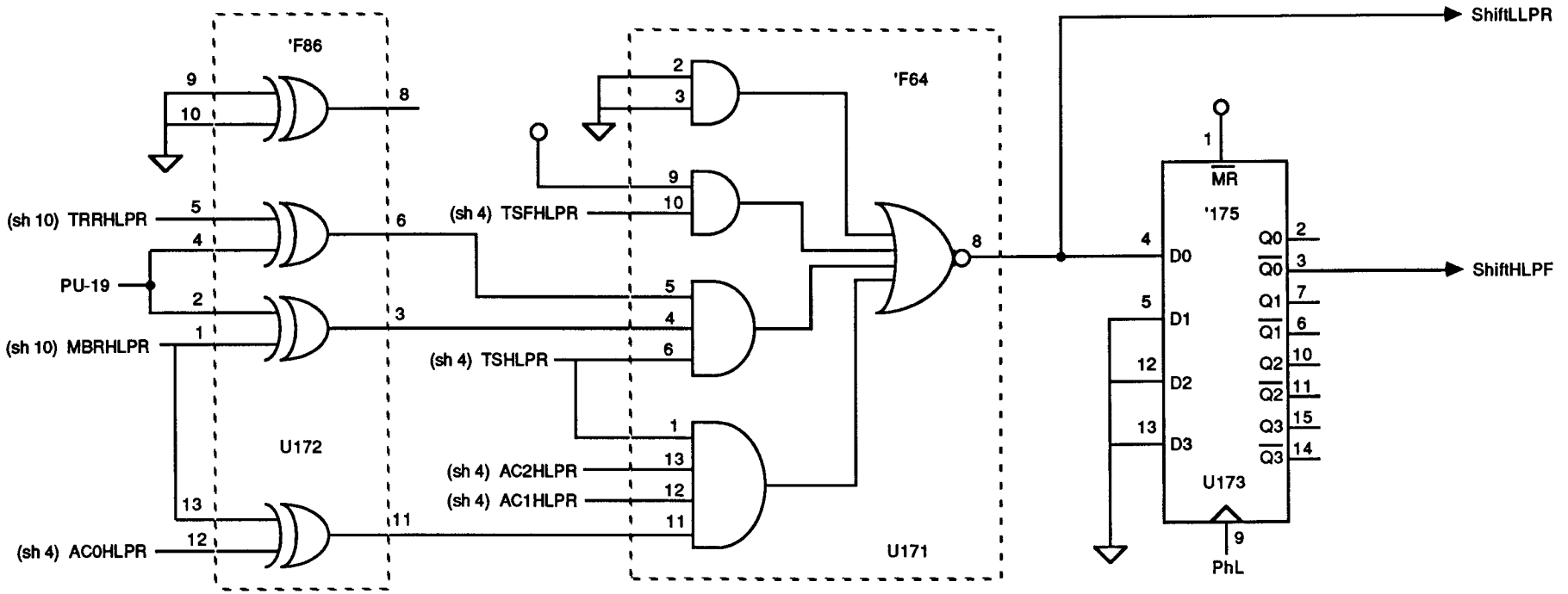
UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 5 of 11	Date: 21-Jul-86	Rev: A
Comments:		





Microcode Load Control

UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 6 of 11	Date: 21-Jul-86	Rev: A
Comments: "Marvelous...." -- Gene Shallot, New York Times		



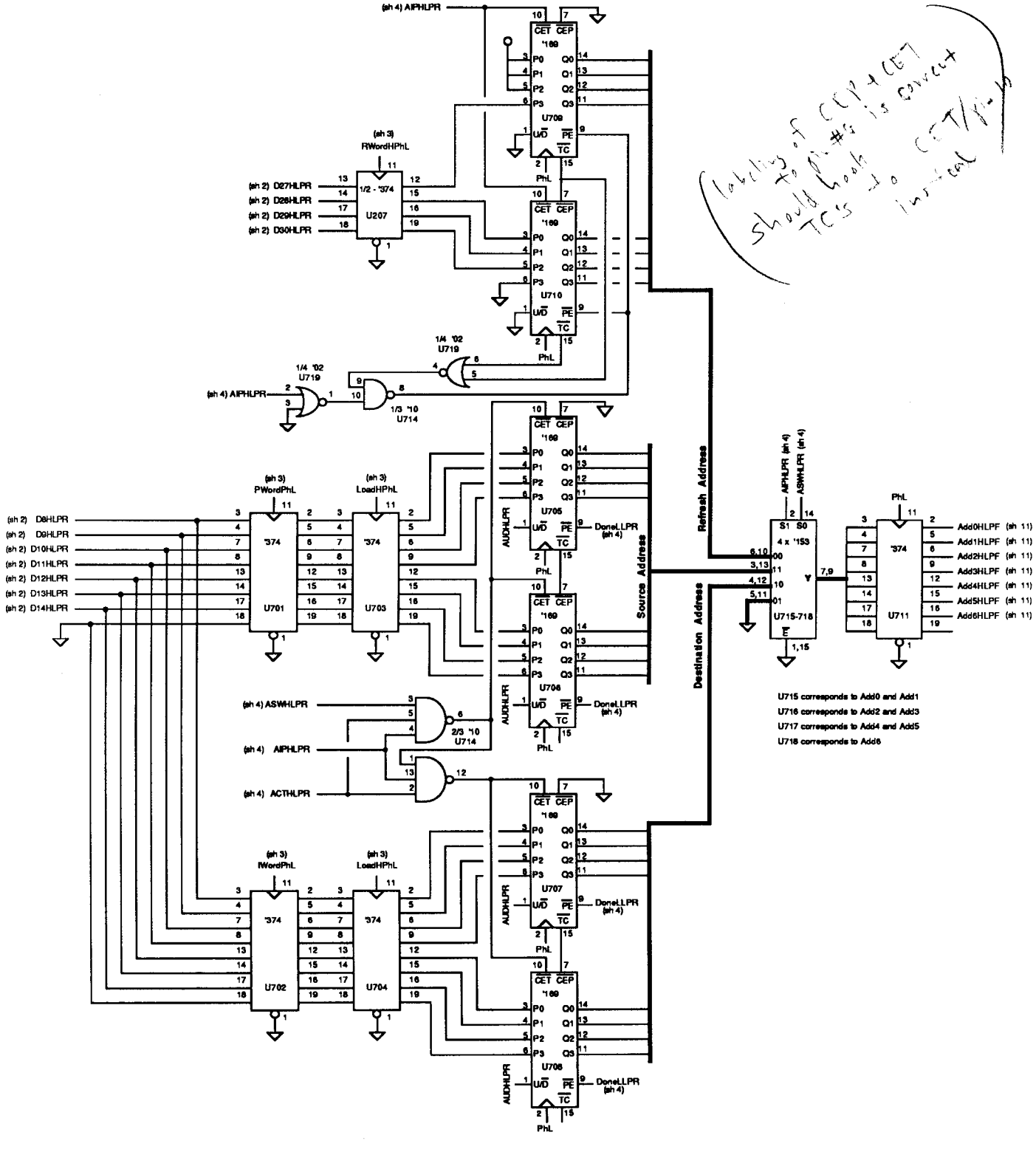
Shift Control Logic

Shift = ForcedShift + CondShift · $\overline{TRR}$ · $\overline{MBR}$

+ CondShift · MBR · looking for MBR

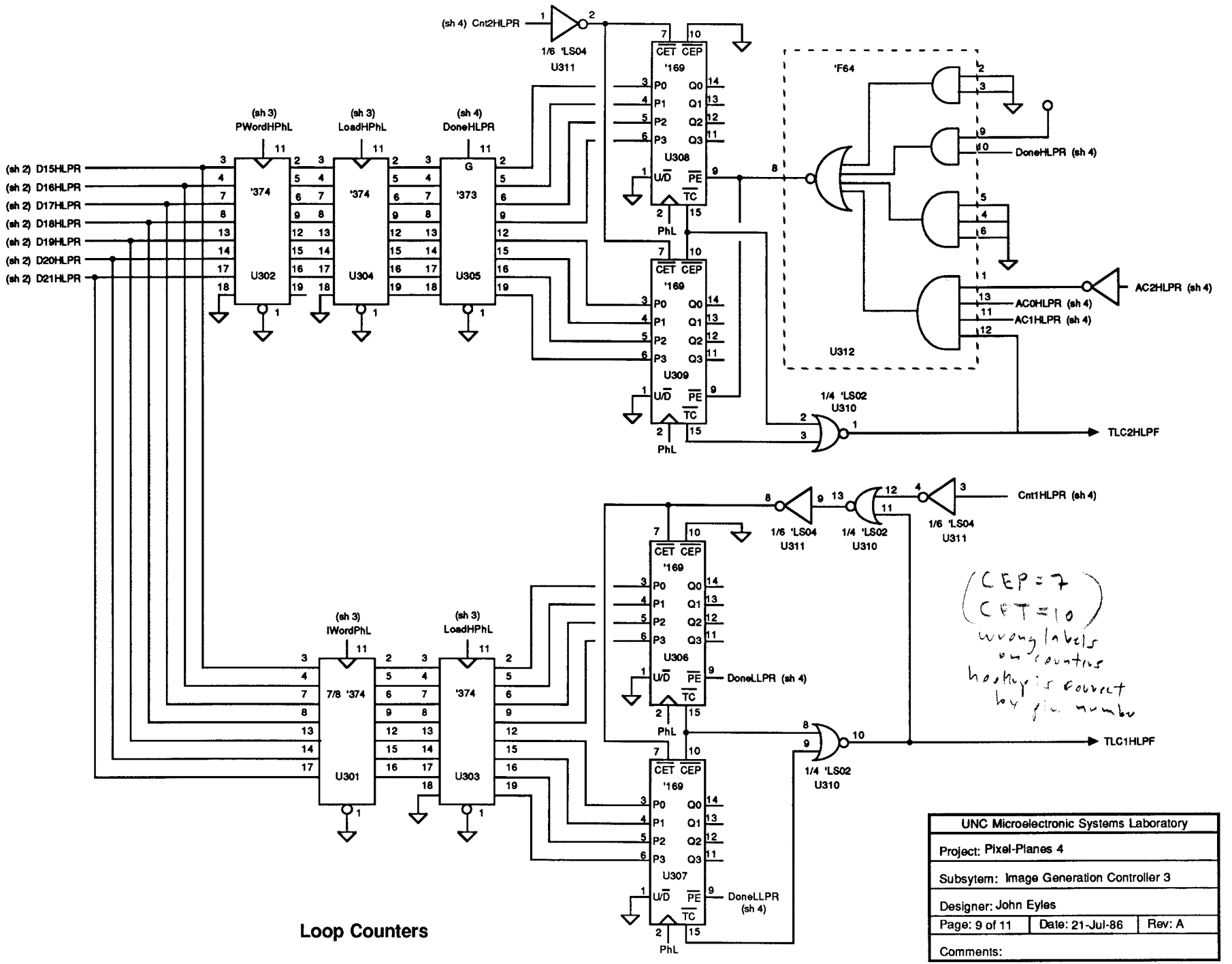
+ CondShift · $\overline{MBR}$ · looking for TRR

UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 7 of 11	Date: 21-Jul-86	Rev: A
Comments:		

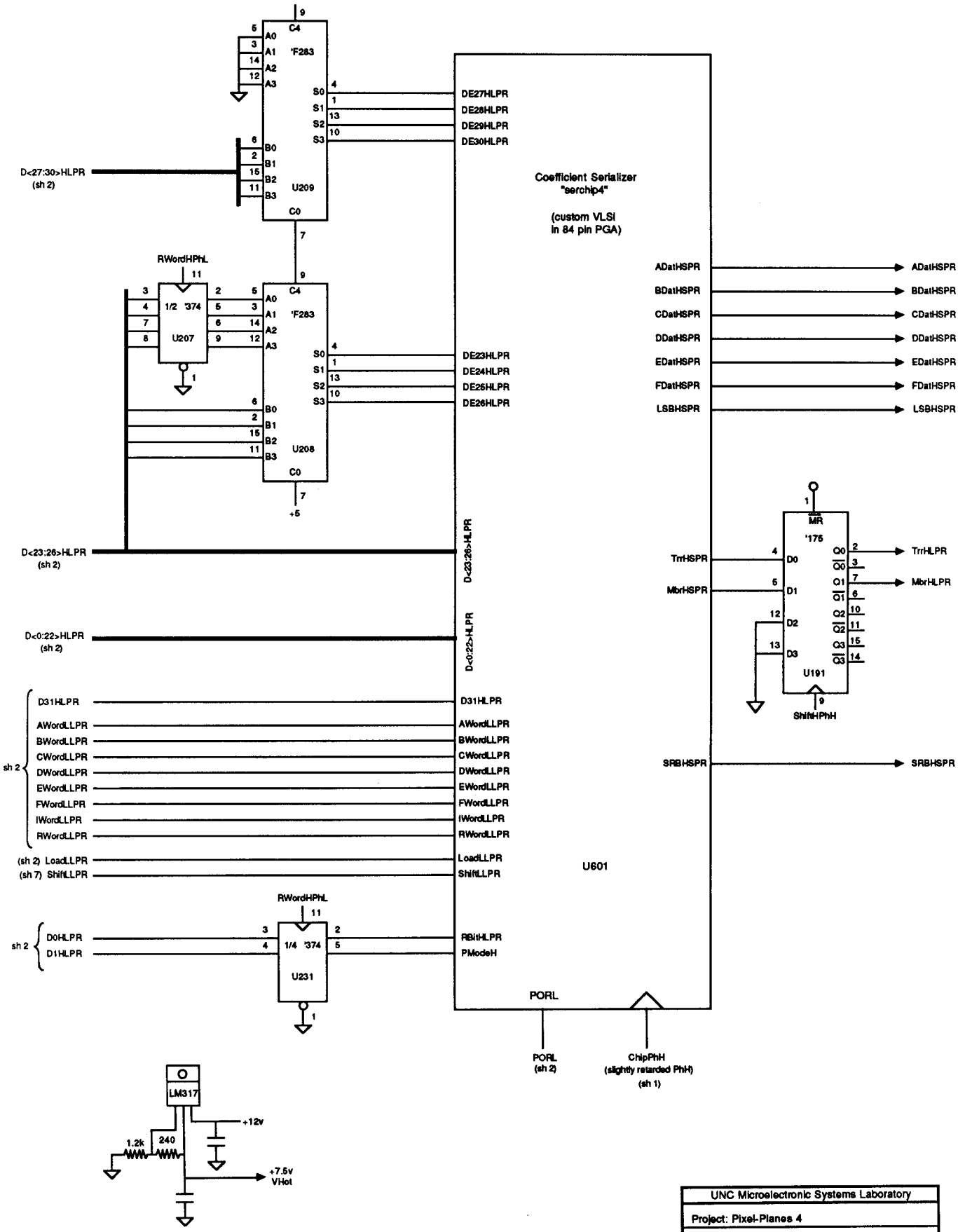


Pixel Memory Address Logic

UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 8 of 11	Date: 21-Jul-86	Rev: A
Comments:		

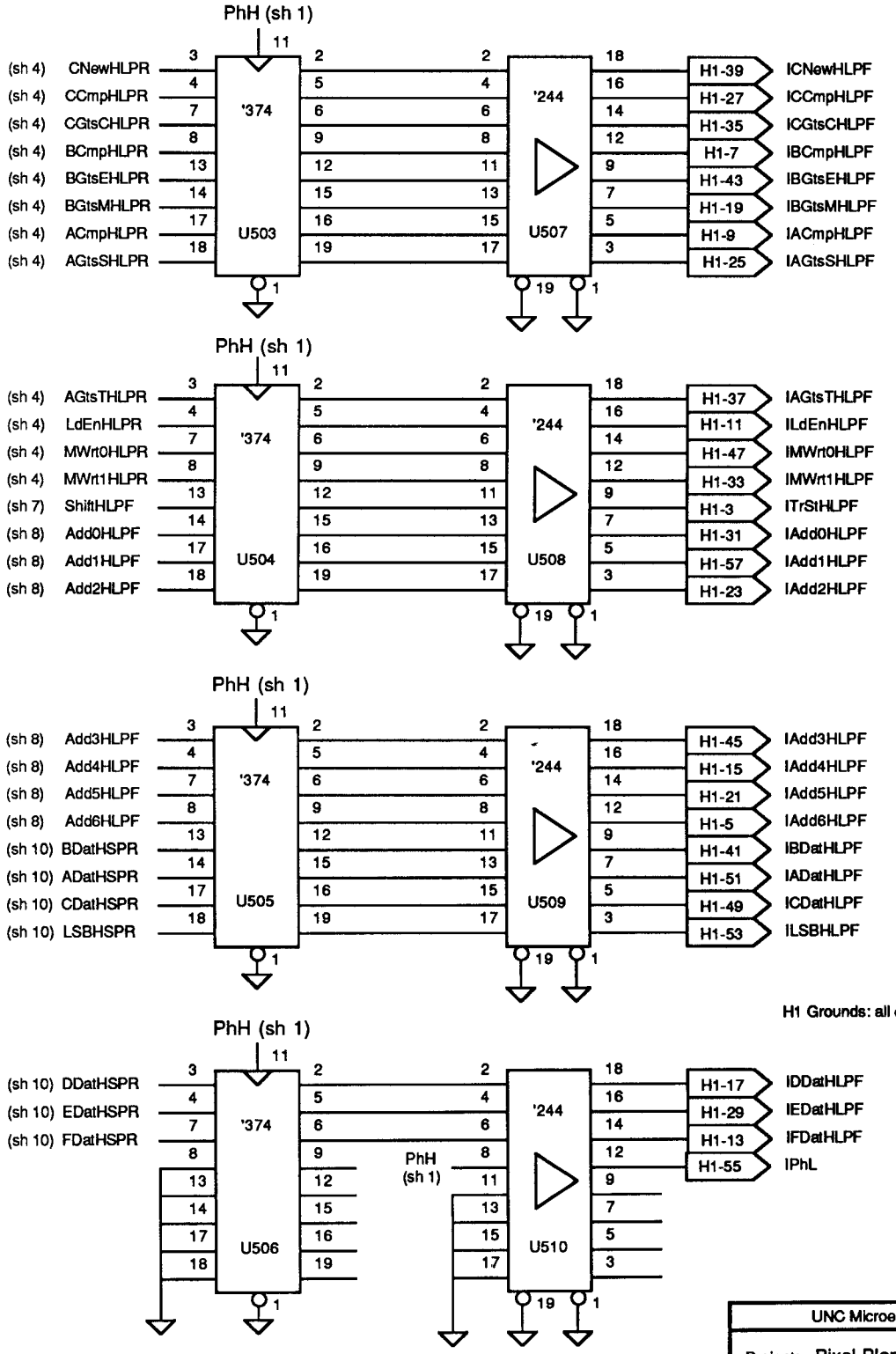


UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 9 of 11	Date: 21-Jul-86	Rev: A
Comments:		



UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 10 of 11	Date: 21-Jul-86	Rev:A
Comments:		

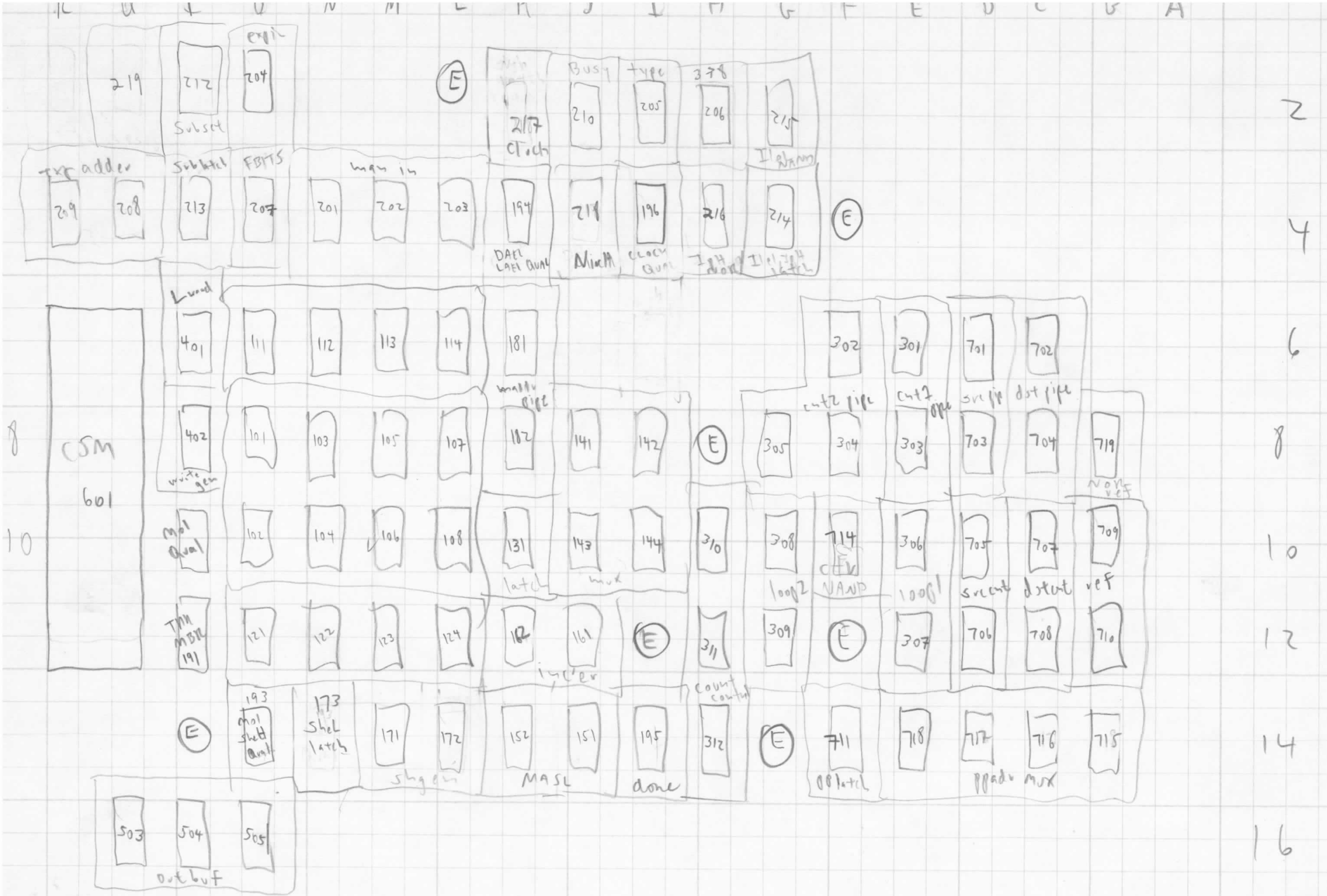
Coefficient Serializer



NOTE: the clock sense of outputs on the H1 connector is reversed, to be compatible with the frame-buffer nomenclature

H1 Grounds: all even numbered pins, 1, 59

UNC Microelectronic Systems Laboratory		
Project: Pixel-Planes 4		
Subsytem: Image Generation Controller 3		
Designer: John Eyles		
Page: 11 of 11	Date: 11-Jul-86	Rev: A
Comments:		



A B C D E F G H I J K L M N O P Q
2
4
6
8
10
12
14
16
219
212
204
Subset
exp adder
209
208
Subbtch
213
FBITS
207
man in
201
202
203
217
Clock
Busy
210
type
205
206
215
194
DAEL LAEL QUAL
218
196
Clock Qual
216
214
E
Load
401
111
112
113
114
181
302
301
701
702
CSM
601
402
write gen
101
103
105
107
182
141
142
305
304
303
703
704
719
cut2 pipe
dst pipe
NOR ref
Mol Qual
102
104
106
108
131
143
144
310
308
714
306
705
707
709
latch
mux
loop2
NAND
loop1
ref
TMM MBR
191
121
122
123
124
161
311
309
307
706
708
710
193
173
Shel latch
171
172
152
151
195
312
711
718
717
716
715
count control
MASL
done
00 latch
mux
503
504
505
out buf