

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
HAYSTACK OBSERVATORY**

WESTFORD, MASSACHUSETTS 01886

3 February 2006

Telephone: 781-981-5407  
Fax: 781-981-0590

To: Mark 5 Development Group  
From: Will Aldrich  
Subject: Data Input Module Mark 5B I/O Board Theory of Operation

#### INTRODUCTION

The data input module (DIM) is an FPGA configuration which allows VSI bit streams to be formatted and sent to the FPDP bus for recording in a Mark 5B system. It also has the capability of cascading the data streams to another DIM downstream. The DIM configuration (together with the Mark 5B I/O board) conforms to the requirements of the VSI-H specification. [1]

The same physical board is used to extract the recorded data via the FPDP bus and forward the data to a correlator. This operation is accomplished by a different configuration called the Data Output Module (DOM). The DOM is not discussed in this document.

#### REFERENCES

1. "VLBI Standard Hardware Interface Specification – VSI-H", Rev1.1, 12 December 2002 (available at <http://web.haystack.edu/vsi/index.html>)
2. "Mark 5B Design Specifications – Mark 5 Memo #019", Whitney, A. R. and Cappallo, R. W., 24 November 2004. (available at <ftp://web.haystack.edu/pub/mark5/index.html> )
3. "Mark 5B DIM Parameters and Procedures" – Mark 5 Memo #24.3, Whitney, A. R. and Aldrich, W, H, 8 November 2005 (available at <ftp://web.haystack.edu/pub/mark5/index.html> )
4. "Mark 5B I/O Board Physical Description" – Mark 5 Memo #TBD, Aldrich, W. H, 7 December 2005. (available at <ftp://web.haystack.edu/pub/mark5/index.html> )

#### BLOCK DIAGRAM

A block diagram of the DIM is shown in figure 1. The block diagram will serve as a guide to a high level description of the DIM operation. Detailed descriptions of the various blocks are given in a later section.

The primary path for active bit streams starts on the upper left hand side of figure 1. The bus signal BS[31:0] has one signal per bit stream and is sampled by the rising edge of the input clock, cki. The bit streams pass through a series of registers and a multiplexer before being clocked into a block entitled "Reduced Crossbar".

At the reduced crossbar the bit streams that are flagged as active by the bit stream mask (BSM) are concentrated in a continuous set adjusted to the least significant (lowest numbered) bit positions on the bus RXB[31:0]. The active signals are represented diagrammatically by the thicker lines on the figure.

The block entitled "Fan Out" accumulates the active signals from the reduced crossbar to fill up a 32 bit word. The number of active signals (and "1"s in the BSM) must be a power of two from 1 to 32. Thus the number of active signals fit evenly into a 32 bit word. In the degenerate case of all

32 bit streams being active the reduced crossbar and the fan out blocks reduce to simple 32 bit registers.

The packed data streams pass through another multiplexer and are registered and presented to a FIFO which has a depth of 512 words of 36 bits each.

There are two clock domains in the DIM data path. The clock domain boundary is indicated on the block diagram and can be seen to pass through the FIFO and through its controller. All operations to the left of the domain boundary are synchronous to the clock cki. All operations to the right of the boundary are synchronous to the clock ckp. The clocks cki and ckp are asynchronous with ckp running at a higher rate to allow the insertion of header data into the bit stream sequence without requiring cki to pause. Headers are inserted into the data stream in a non-data-replacement mode.

Bit stream data is extracted from the FIFO in response to ckp which is also the FPDP "PSTB" signal.

Each frame of data consists of a four-word header followed by 2500 data words. In the block diagram a multiplexer and a register follow the FIFO. The multiplexer allows the insertion of the header under the influence of the frame controller. Headers are computed on a frame-by-frame basis by the header generator. The final register is updated on the falling edge of ckp and holds the data for presentation to the FPDP bus.

Crosshatched registers hold operating parameters, which are loaded via the PCI bus on the motherboard. These parameters are static in the sense that they do not change during the time that a recording is being made. They are set in advance of the start of a recording and are left constant until the recording is complete.

There are a few blocks which are not in the data path whose function is described here. The clock generator interface provides the link between the PCI bus and the on-board clock generator. The clock generator can be set to provide a cki clock when there is no VSI clock. The TVR and TVG (test vector receiver and test vector generator) check and generate the test vector sequence as specified in the VSI-H specification. The DIM can thus check an incoming sequence for errors by comparing with the required test vector sequence. In addition, the DIM can generate the test vector sequence for recording as a diagnostic aid. The TVG can operate in a counting mode, also for diagnostic purposes.

The PPS counter produces a pair of output signals. One output is the signal "PPS" which is a signal with a period of one second and a duration of one cki clock period. This output can be synchronized with any of the three input PPS signals in such a way that the rising edge of "PPS" will match the rising edge of the selected input at the time of synchronization. After synchronization the input signal has no effect on the PPS counter. The second output is the signal "Ilen" (input latch enable) which allows decimation of the input bit streams if it is required by the operational mode.

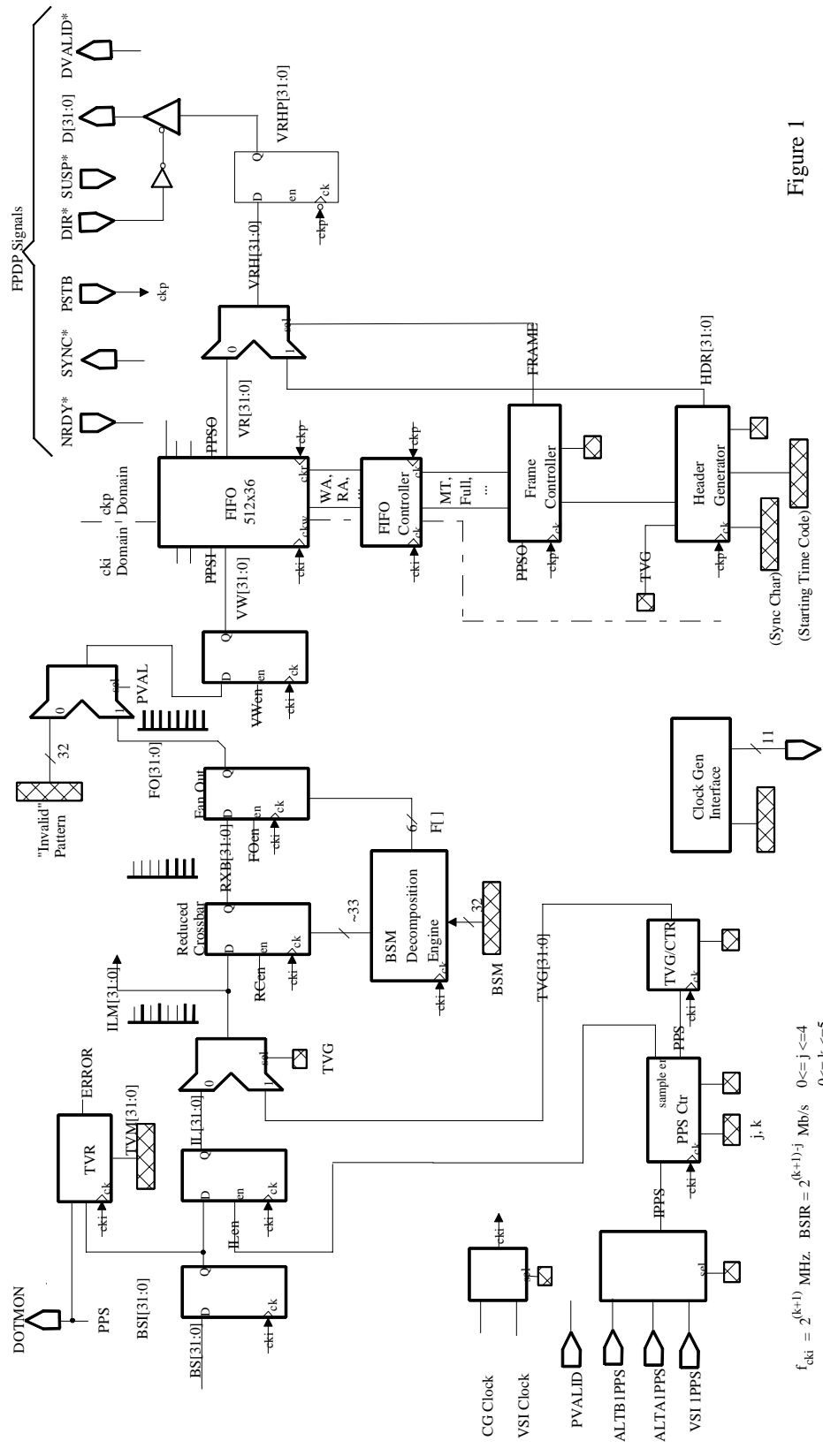


Figure 1  
 MARK 5B  
 DIM Block Diagram  
 W. Aldrich  
 12/7/05

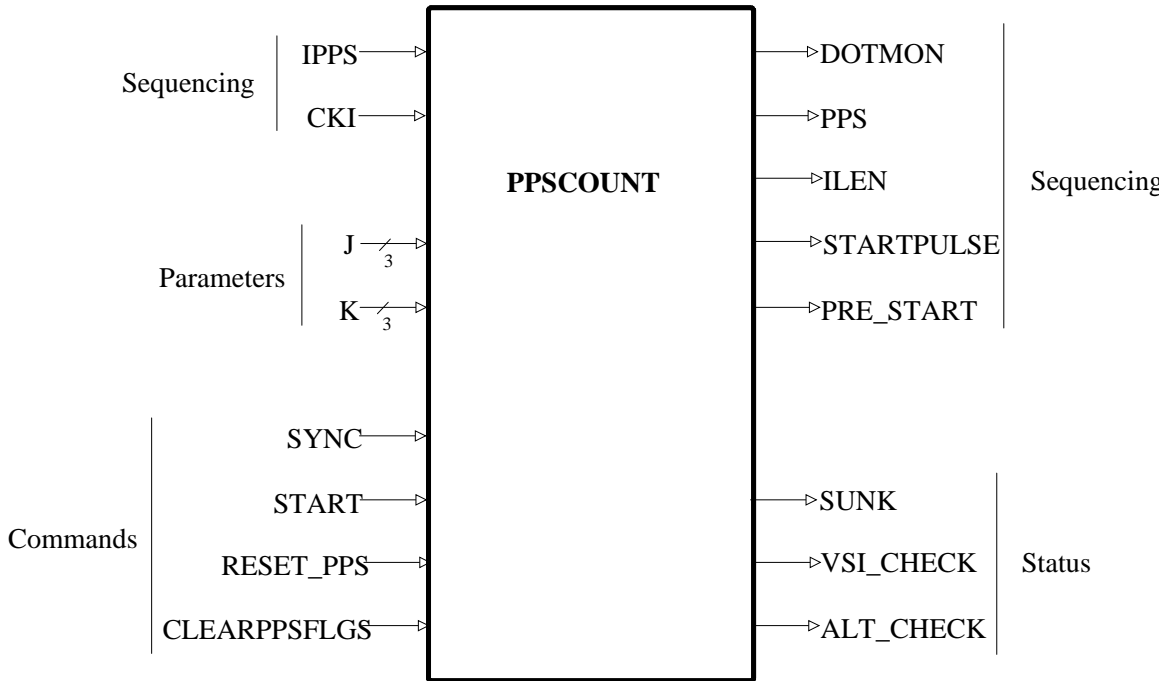
$$f_{cki} = 2^{(k+1)} \text{ MHz. } BSR = 2^{(k+1) \cdot j} \text{ Mb/s } \begin{matrix} 0 \leq j \leq 4 \\ 0 \leq k \leq 5 \\ j < k \end{matrix}$$

Parameter registers are written from and read back to PLX Local Bus  
 => Static Parameter (Does not change during a scan)

## BLOCK DESCRIPTIONS

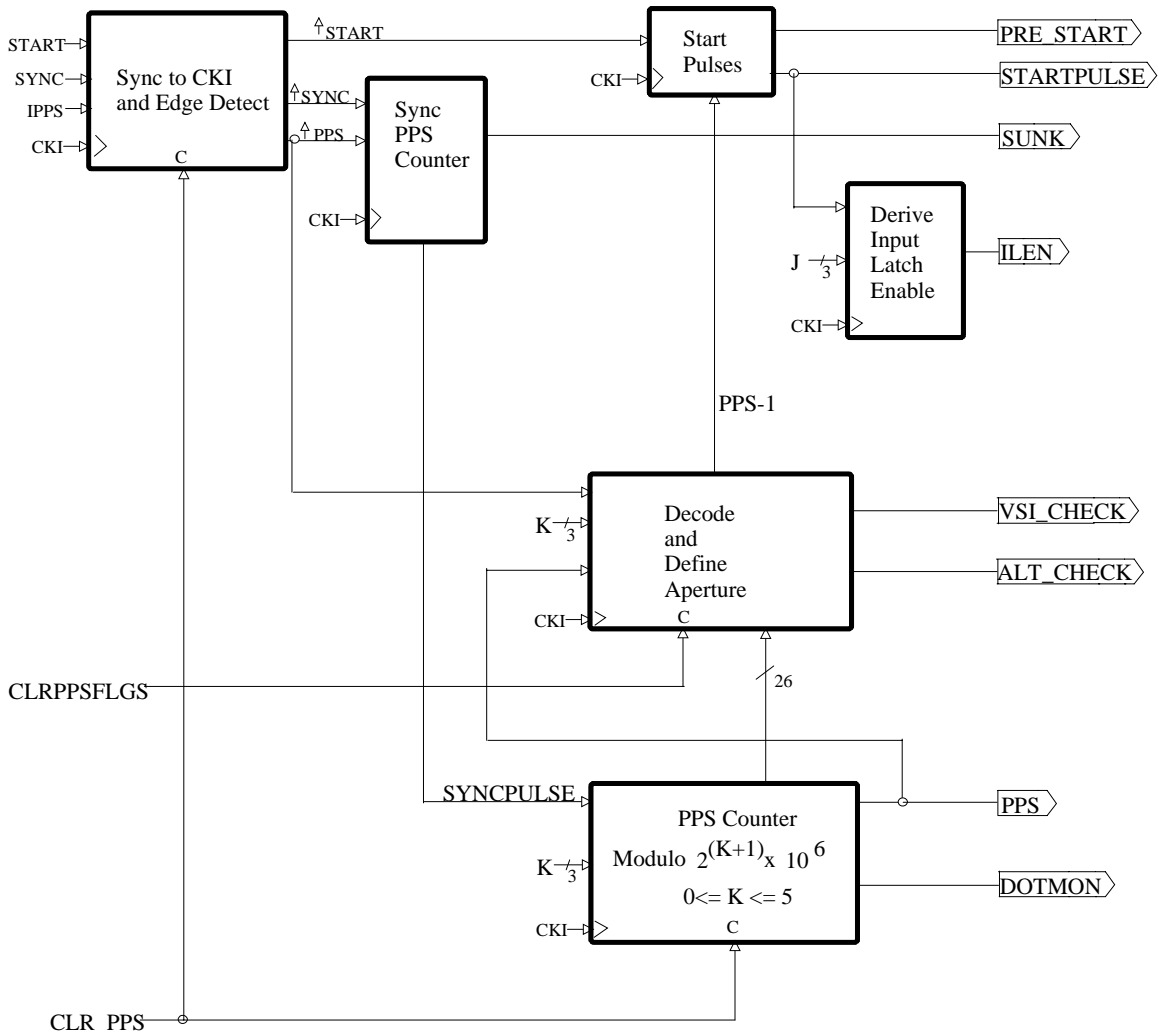
### *PPS COUNTER*

The inputs and outputs of the block entitled “ppscount” are shown in figure 2 below. Note that this block contains more detail than the corresponding block in the DIM Block Diagram of Figure 1. It is closer to the block with the same title which appears in the schematic set produced in the XILINX “Engineering Capture System” (ECS). Readers should have no trouble relating block descriptions to schematic representations.



**FIGURE 2**  
**THE PPS BLOCK**

Inputs and outputs characterized as “sequencing” are the operating signals which contribute to DIM operation. “Parameters” are values which are software settable and which remain constant during a recording scan. “Commands” are very similar to parameters in that they are software controlled bits which are usually set to cause a specific action to take place. “Status” signals are condition-sensing bits which are software readable.



**FIGURE 3**  
**INSIDE THE PPS BLOCK**

Figure 3 shows a further breakdown of the ppscount block and is referenced in the following descriptions of ppscount behavior.

The primary sequencing inputs IPPS and CKI are the selected one per second input signal and the input clock respectively as described in the overview of Figure 1. All sequencing operations take place with respect to the rising edge of CKI. Consequently, the sequencing inputs along with the SYNC and START command bits are synchronized in the upper left hand block of Figure 3. The rising edges of the synchronized signals are represented by signals with the “up-arrow” prefix. These signals are one CKI period in duration.

A basic operation in the DIM is the synchronization of the internal signal “PPS” to the selected one-per-second input. This is accomplished by the software causing the command bit “SYNC” to make a “0” to “1” transition. The block “Sync PPS Counter” will then be set to detect the next occurrence of the leading edge of IPPS and will issue the signal SYNCPPULSE one time only. SYNCPPULSE will cause PPS to occur for one period of the CKI clock. PPS will occur continuously at one second intervals provided CKI is running at the frequency  $2^{*(k+1)}$  MHz. (Note that the value of the parameter “K” must be set to agree with the frequency of CKI prior to the SYNC command.) Once PPS synchronization takes place the internal signal PPS continues to

run at 1 Hz. with no further requirement from IPPS and the status signal “SUNK” will become a “1”.

In the absence of a synching command, the signal PPS will run at 1 Hz, but with arbitrary phase. The signal “DOTMON” has a “0” to “1” transition at the same time as PPS, but it is stretched (remains a “1”) in order to make it easy to find on an oscilloscope. DOTMON remains set for 131,074 CKI periods and then becomes “0” for the rest of the second. DOTMON has a duty cycle of about 7% at a CKI frequency of 2 MHz and a 0.4% duty cycle at a CKI frequency of 32 MHz.

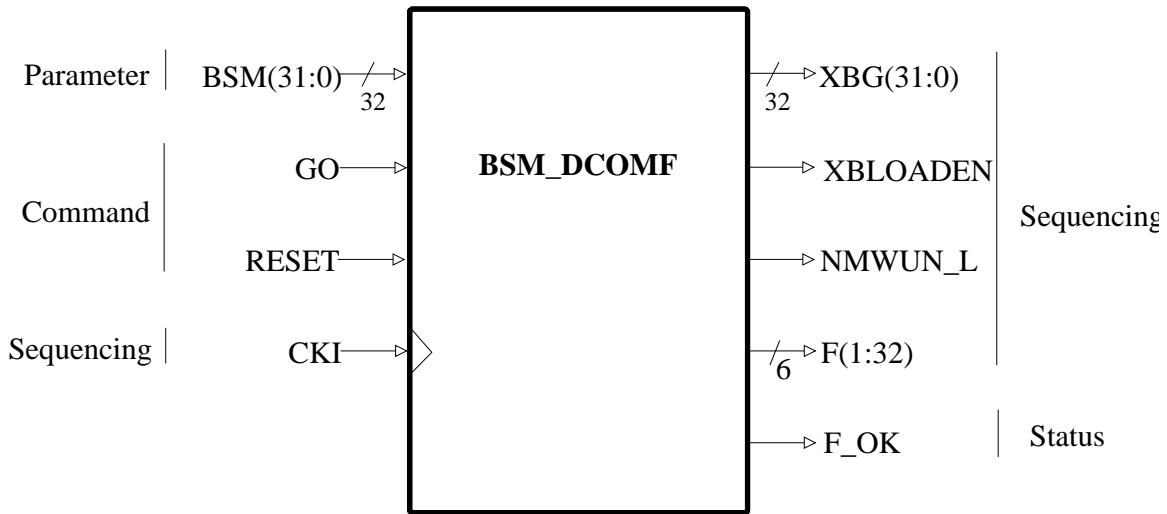
The block “Decode and Define Aperture” has access to all 26 bits of the PPS counter and can therefore anticipate the occurrence of the PPS signal. The signal PPS-1 occurs one clock period before PPS and is discussed later. Two comparisons are made between PPS and the selected input signal, IPPS. One comparison requires exact synchronization between IPPS and PPS once the pipeline delay is accounted for. This should be the case when a VSI data input is being used. If the comparison fails, the status bit VSI\_CHECK becomes a “1” and will remain so until cleared by the command CLRPPSFLGS. A second comparison allows for some drift between PPS and IPPS and requires that IPPS lie within an aperture of three clocks before until three clocks after PPS. If this comparison fails, the bit ALT\_CHECK is set to “1” and can only be cleared by the signal CLRPPSFLGS.

Once the PPS synchronization has been accomplished, the actual onset of data recording can begin. The parameter “J” controls decimation and must be set to allow each  $(2^{**}J)$ th data value to be taken. Note that  $J=0$  implies no decimation. Recording is primed with the command signal START making a “0” to “1” transition. The implication of this event is that data taking will commence on the next IPPS tick. Data is taken in response to the signal ILEN. The signal PRE\_START corresponds to the rising edge of the START command. The signal STARTPULSE initiates taking of the first data sample and begins the computation of the input latch enable signal (ILEN). Detailed examples of the startup sequences are shown in the SEQUENCE EXAMPLES section of this document.

### *BSM DECOMPOSITION ENGINE*

The input and output signals from the BSM decomposition block are shown in figure 4 below. The task of the BSM decomposition block is to analyze the 32 bit parameter, BSM, and determine a sequence of gating functions which appear on the bus entitled XBG(31:0). In addition, the block determines the functions F1, F2, F4, F8, F16, and F32. These functions are disjoint and specify how many active bit streams the bit stream mask specifies. If the analysis shows exactly a power of two bits set in the BSM the flag F\_OK will be set to “1”.

The bit stream decomposition is a set up operation which must be done prior to any recording. The operation is started when there is a “0” to “1” transition on the “GO” command line (software controlled). Operations are synchronized to the clock CKI and in the worst case takes 135 clock periods to complete.



**FIGURE 4**

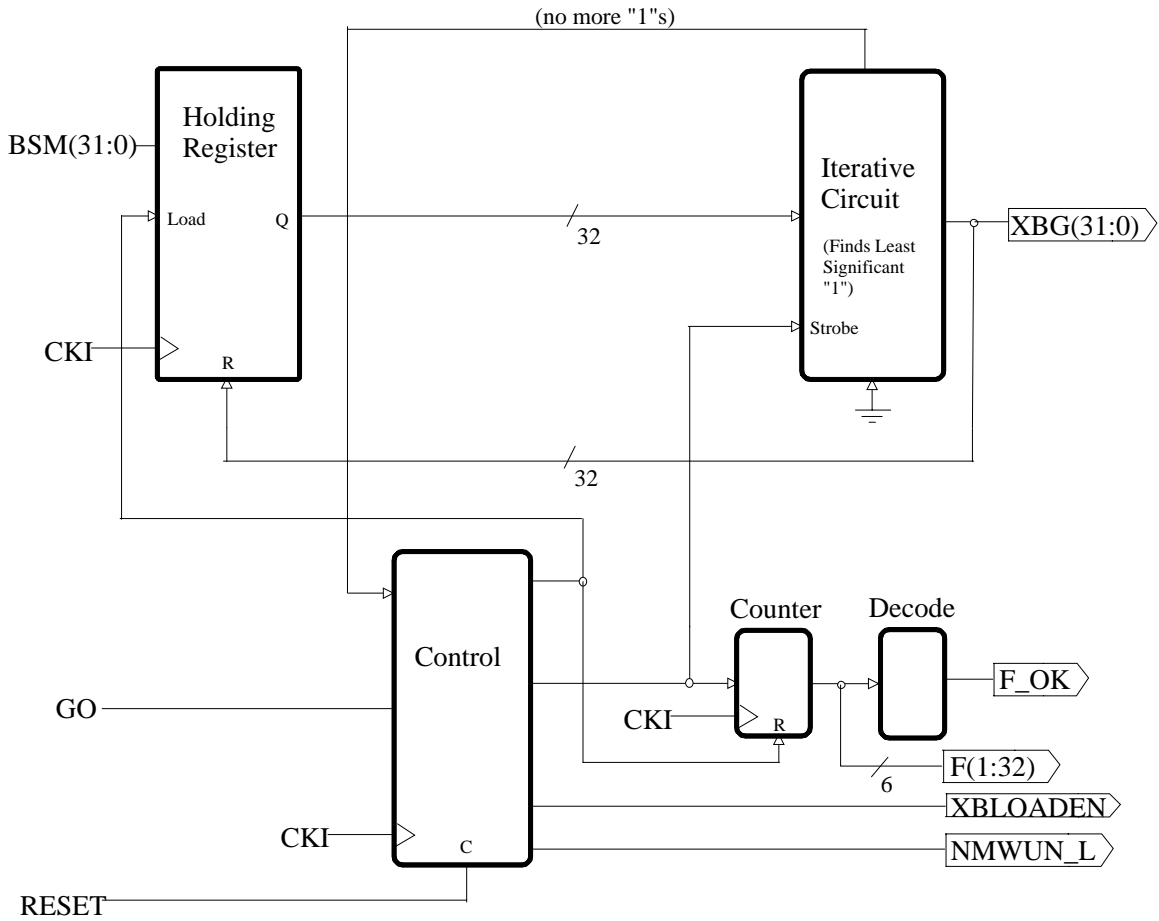
**BSM DECOMPOSITION BLOCK**

The details of the decomposition operation are explained in conjunction with the detailed block diagram of figure 5. When a transition from “0” to “1” is detected on signal “GO” the control block begins operation by loading the 32 bit BSM into the holding register, resetting the counter and beginning the sequencing of two state machines whose outputs cause the count/strobe signal and the XBLOADEN signal.

The iterative circuit examines the contents of the holding register and locates the least significant “1”. When the signal strobe is emitted by the control, a 32 bit crossbar gating function is output on the bus XBG(31:0). This function has a single “1” in the position of the least significant “1” in the holding register. The function is fed back to the holding register where it resets the least significant “1”. The function is forwarded to the reduced crossbar and is interpreted there in response to the signal XBLOADEN. This process will be repeated until the holding register is completely cleared, at which point the control will be informed that there are no more “1”s. The control will inform the outside world with the signal NMWUN\_L.

Each time a “1” is reset in the holding register, the counter is incremented so that the counter will contain the number of ones in the BSM when the process terminates. If the number is 1, 2, 4, 8, 16, or 32 the flag F\_OK is set to “1”. The bits of the counter are also output to the rest of the DIM to control further processing.

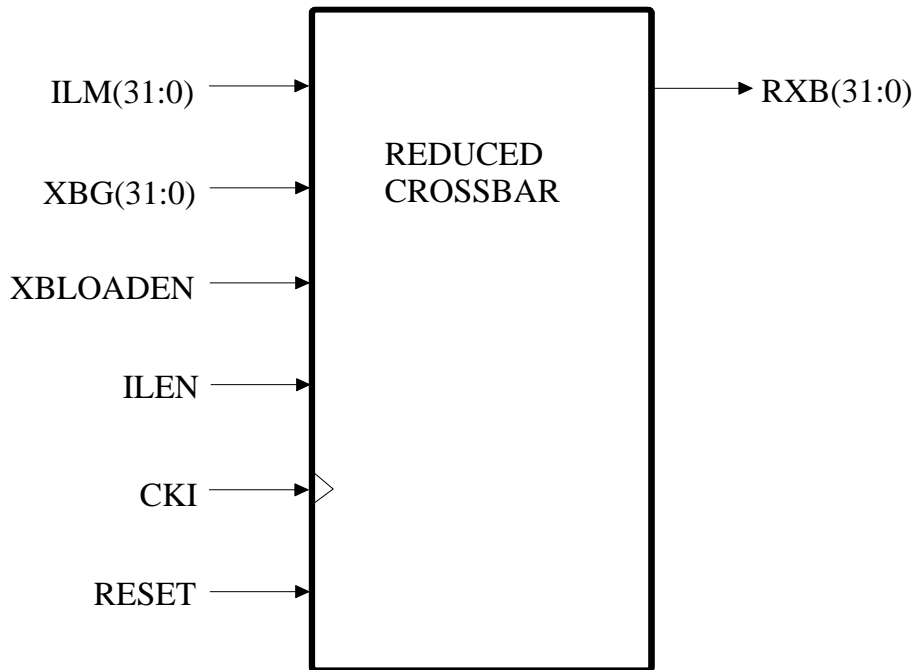
The XBLOADEN signal occurs exactly sixteen times per BSM decomposition and it causes the reduced crossbar switch to load gating function corresponding to each active position. Only the first sixteen gating functions are required because the next legal count above 16 is 32 which means that all bit streams are in use. In the case of all bits of the BSM set to “1”, the counter and the iterative circuit process all 32 bits. This is necessary to insure that there is not an erroneous number of “1” s in the BSM.



**FIGURE 5**  
**INSIDE THE BSM DECOMPOSITION BLOCK**



## REDUCED CROSSBAR



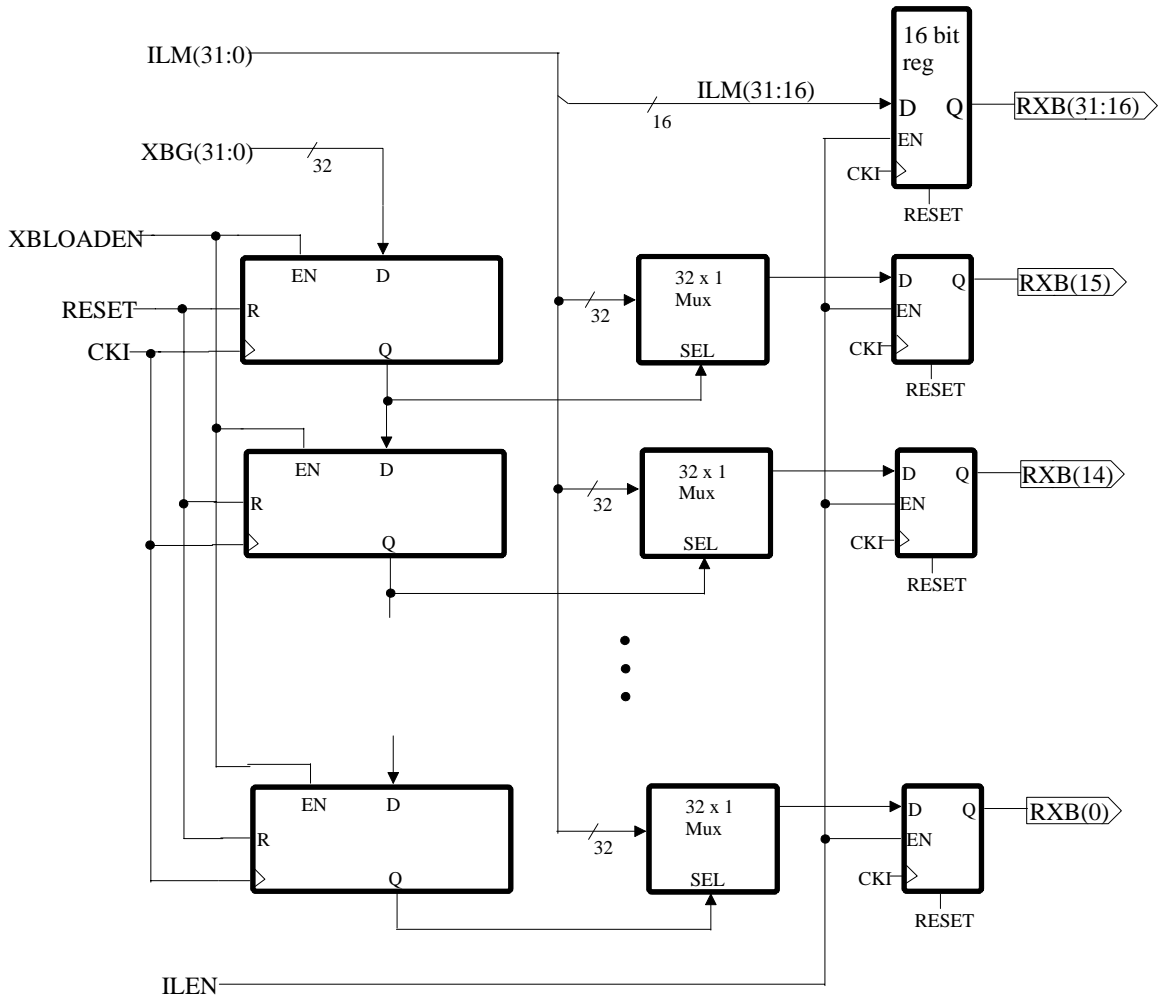
**FIGURE 6**

### **REDUCED CROSSBAR BLOCK**

Figure 6 shows the block description of the reduced crossbar and its input and output signals. All signals are of the sequencing type in that there are no command or status connections to the block.

Description of the operation of the reduced crossbar follows the operation shown in figure 7 which is a higher detail block diagram of the circuit.

The bit streams enter the block on the bus ILM(31:0). The crossbar gating functions enter the block on the bus XBG(31:0) during the process of bit stream mask decomposition. There are exactly 16 gating functions which are shifted into the holding registers in response to the 16 occurrences of the signal XBLOADEN. The gating functions remain static during a scan. They cause the active bit streams to be directed to the block outputs by the operation of the 32 x 1 multiplexers. The least significant bit stream appears on signal RXB(0), the next on RXB(1), and so on. The active bit streams fill in to adjacent positions on the bus RXB(31:0) starting with bit "0". Inactive bit streams in the lower 16 positions do not appear on the output. (Their gating function is 0x00000000.) Bit streams 31 to 16 always appear on the upper 16 bit positions of the output, but they are only used in the case that the number of "1"s in the bit stream mask is 32.

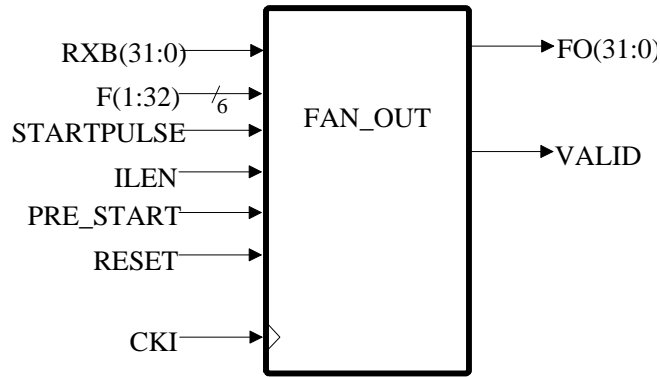


**FIGURE 7**

**REDUCED CROSSBAR INSIDE**

*FAN OUT*

Figure 8 shows the inputs and outputs associated with the “FAN\_OUT” block. This block follows the reduced crossbar in the data path and therefore has signals RXB(31:0) as data inputs. It uses the signals F(1:32) as derived from the BSM decomposition to decide how many bit streams are active on the RXB bus. These bit streams are packed into the output register, starting with the lowest significant bits, until a 32 bit word is assembled. The 32 bit word appears on the output bus FO(31:0). The signal ILEN signifies when the input data is valid and the signal VALID signals when the 32 bit word is assembled. All operations are synchronous with CKI. The signal PRE\_START is a normalizing signal which has the same effect as RESET on the internal sequencing of the block. Actual operation begins with the occurrence of the signal STARTPULSE.

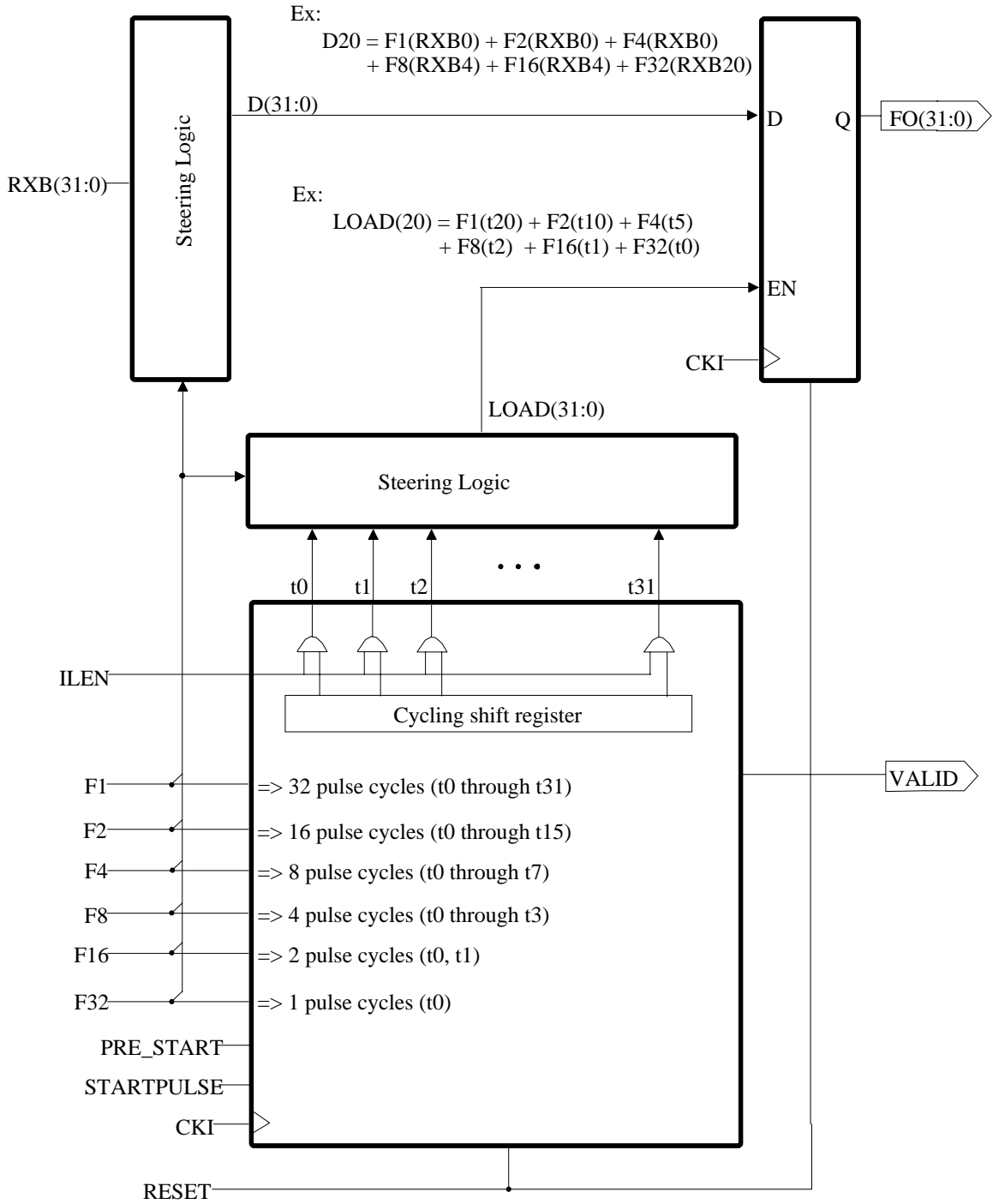


**FIGURE 8**

**FAN\_OUT BLOCK**

The inside structure of the fan out block is shown in figure 9. A cycling shift register in the control block controls sequencing. The number of pulses in each cycle is governed by the number of times the active bit streams must be accumulated to construct a 32 bit word. Each cycle assembles a 32 bit word whose availability is signaled by the output signal VALID. Logically, VALID is coincident with each “t0” pulse, starting with the second.

The signal PRE\_START primes the control section and has the same effect as a reset. It occurs in response to a software request for recording, but is not timed with respect to the 1PPS. Actual operation begins when the signal STARTPULSE occurs. This pulse has a precise timing relation to the 1PPS signal. Each cycle of the shift register starts with the signal t0. Successive pulses will load later sets of signals into the output register until the register contains the 32 bits which will be passed on in response to the VALID signal. Examples of this processing are shown in the sequencing section of this document.



**Figure 9**

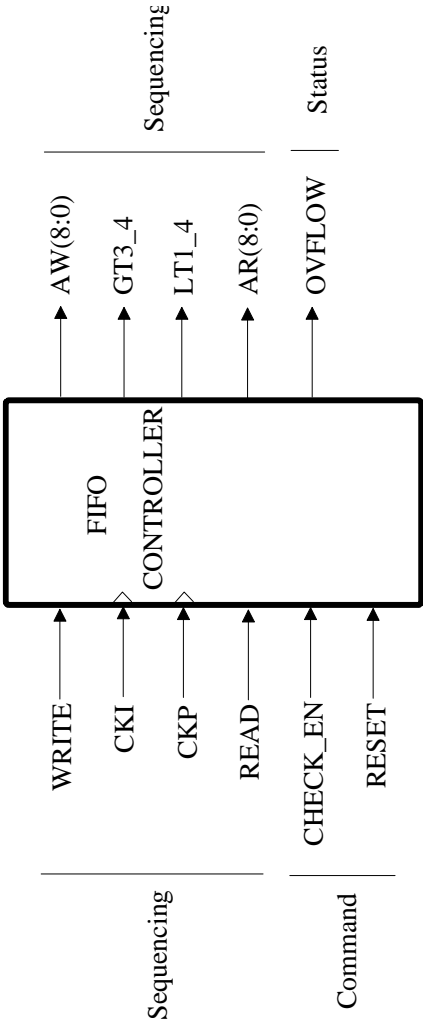
**FAN\_OUT INSIDE**

*FIFO AND FIFO CONTROLLER*

The FIFO is built from a 512 word x 36-bit dual-port buffer memory. This component is a block in the FPGA fabric and is ideal for the FIFO application as the two ports are completely independent. The “A” port is used only for writing which is done in synchronism with the CKI clock. The “B” port is used only for reading which is done at the rate of the CKP clock which is

also the rate of the FPDP bus operation. The CKP rate is normally 33 MHz for FPDP operations and 66 MHz for FPDP II operation.

The FIFO controller block is shown in figure 10 which identifies the input and output signals. The primary task of the FIFO controller is to produce the addresses which are required at the memory ports. These addresses are AW(8:0) for the write port and AR(8:0) for the read port. Writing is done by the signal WRITE which is synchronous with the CKI clock. RESET initializes the controller and by the signal READ which is synchronous to the CKP clock. RESET initializes the controller and the signal CHECK\_EN allows the controller to test for the overflow condition which is signaled by OVFLOW becoming "1". The two flags GT3\_4 and LT1\_4 are synchronous with the CKP clock and indicate that the FIFO is "Greater Than ¾ full" and "Less Than ¼ full" respectively.



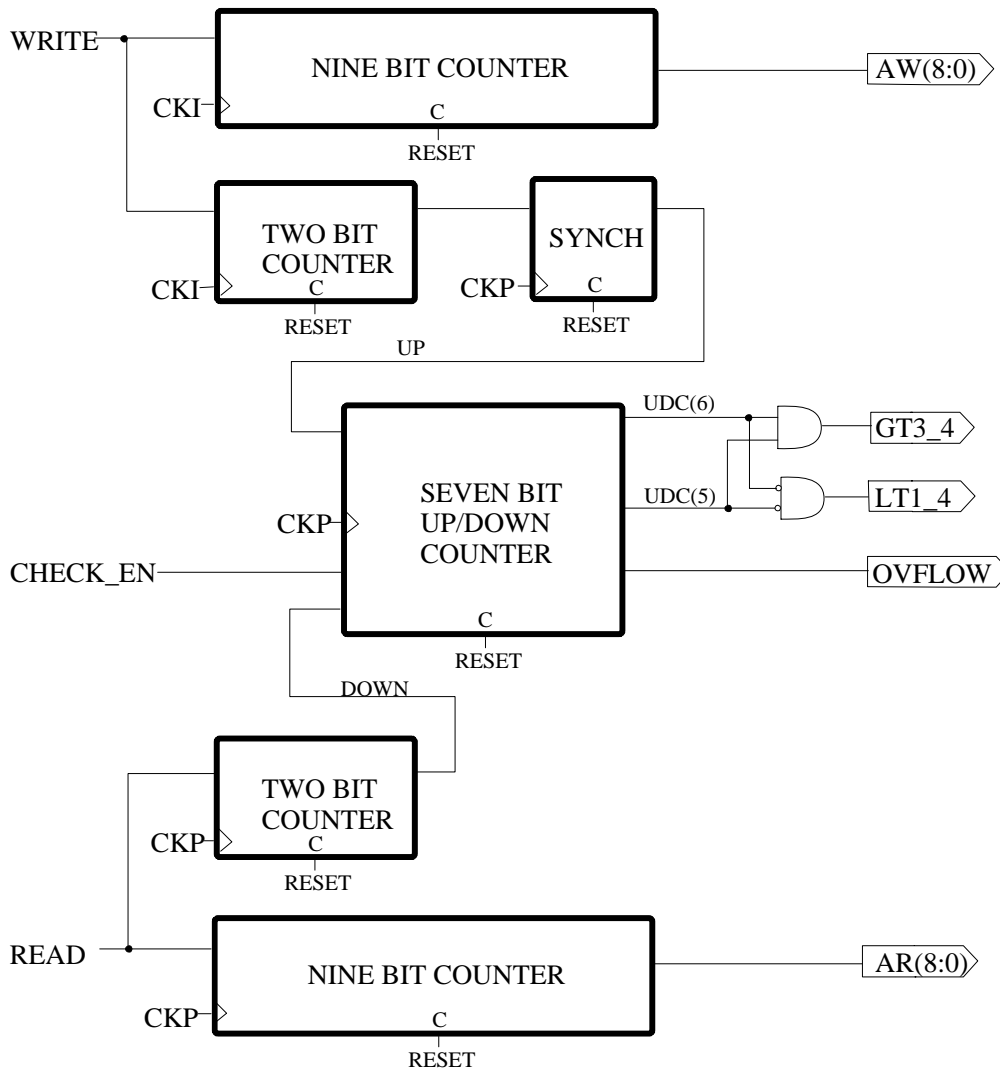
**FIGURE 10**

**FIFO CONTROLLER**

Figure 11 shows the inside of the FIFO controller and shows how the output signals are derived. The nine-bit read and write addresses are produced by nine-bit binary counters which increment in response to WRITE and READ signals. No particular count of these address counters is decoded, they visit all 2<sup>9</sup> possible states and roll over from maximum count to minimum count. It is convenient to think of the FIFO as a circular buffer with AW(8:0) being a "write pointer" and AR(8:0) serving the function of read pointer.

The status of the FIFO is determined by an up/down counter which counts up for each four entries into the FIFO and counts down for each four extractions from the FIFO. Note that the counter operates synchronously with the CKP clock so the up count from the input section must be synchronized to the CKP clock. The counter starts out at zero when the FIFO is empty. If the counter reaches maximum count, the FIFO is full. An attempt to write another word at that point will cause overflow.

Two flags are derived by decoding the most significant bits of the counter. If both bits are "1", the count is greater than or equal to ¾ full. If both bits are "0", the count is less than or equal to ¼ full. Logic external to the FIFO controller attempts to keep the FIFO between ¼ and ¾ full. This is done by throttling the READ when necessary as the READ is always at a higher rate than the WRITE. The WRITE is therefore free to run at its constant desired rate. Note also that the read must be suspended whenever a frame header must be inserted into the FPDP data stream.



**FIGURE 11**  
**FIFO CONTROLLER INSIDE**

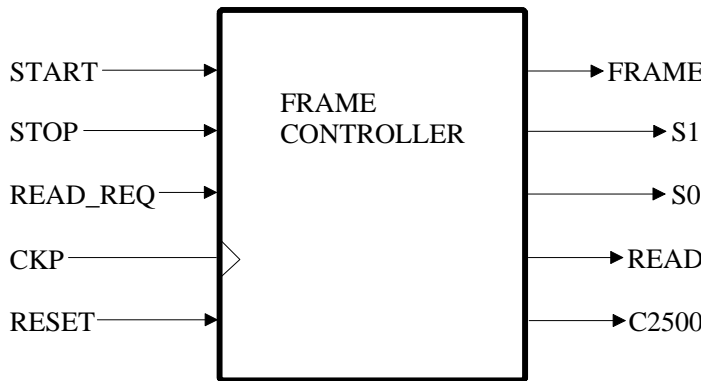
*FRAME CONTROLLER*

The frame controller block is shown in figure 12 which enumerates the input and output signals. START and STOP are closely related to software commands, but are synchronized to the clock CKP. All operations inside the frame controller are synchronized to CKP.

The frame controller operates on the FPDP side of the FIFO and its primary task is to meter out data words to be driven onto the FPDP bus. It also decides when a frame header is to be inserted into the data stream. The signal FRAME causes the data source for the FPDP bus to be switched to the header generator. If FRAME is not asserted, the data source is the FIFO output. S1 and S0 are encoded to select the four words of the frame header in sequence.

The signal READ advances the output pointer of the FIFO as was discussed in the FIFO controller section. C2500 is a single period pulse which indicates that data is to be paused while a header is output.

READ\_REQ, “read request”, is interlocked with FIFO status signals and with FPDP handshake signals.



**Figure 12**

**Frame Controller**

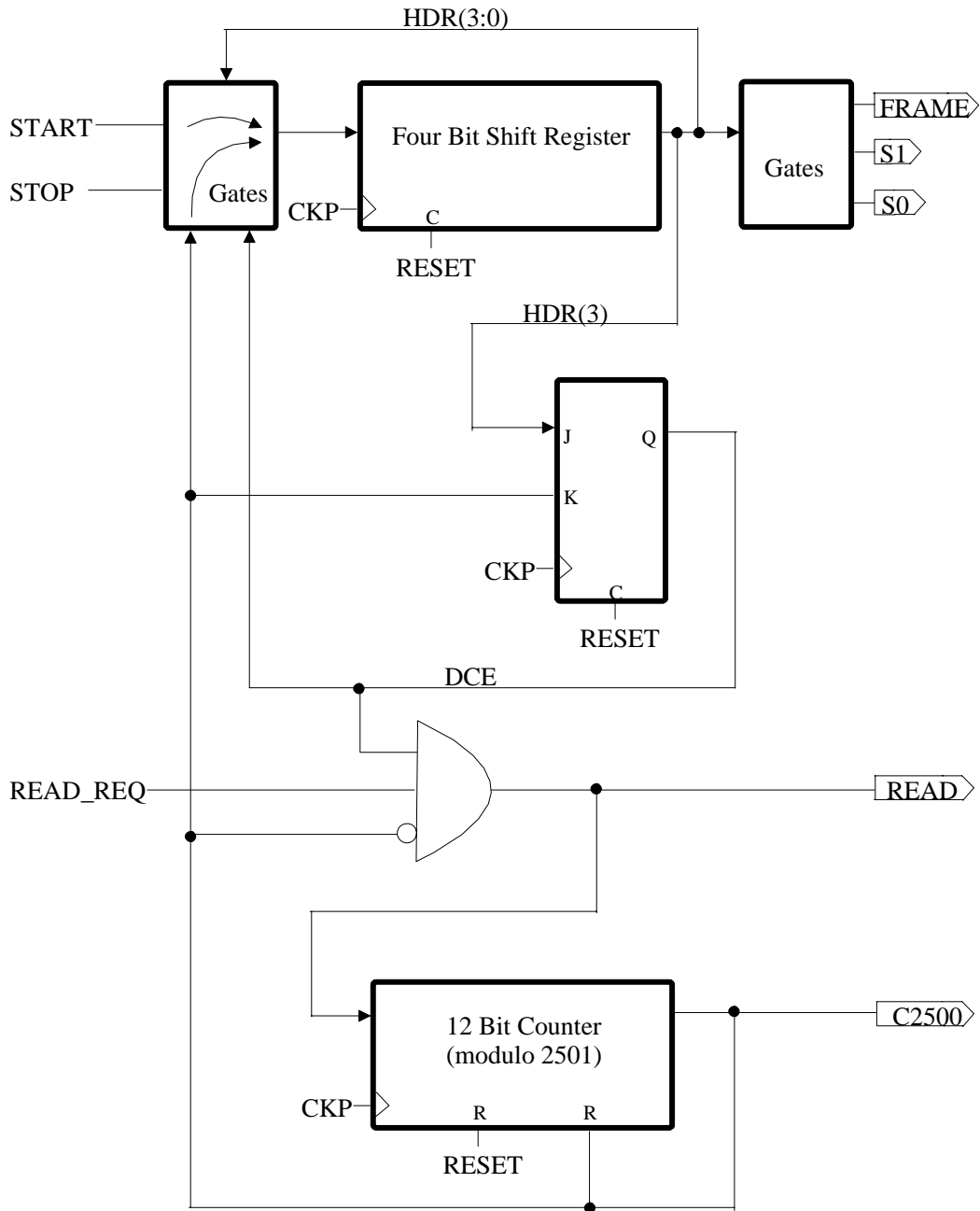
Details of the frame controller are described with the aid of figure 13 which shows some of the interior structure of the block.

The first output of a scan is the first header which is output in response to the synchronized START signal. This is sent to the FPDP bus even though there is no data available for some time as the FIFO is just beginning to be filled. At the end of the first header, the flip flop DCE is set, but no READ is produced until READ\_REQ is asserted in response to the FIFO achieving a status of  $\frac{3}{4}$  full. When READ\_REQ is asserted, the READ signal is forwarded to the FIFO control and the counter is incremented.

The counter increments continuously and data is output to the FPDP bus unless READ\_REQ is withdrawn by FPDP handshake protocol or by the FIFO becoming less than  $\frac{1}{4}$  full. The count of 2500 indicates that the 2500 data words of the frame have been output and it is time to insert another four word header. Note that the signal C2500 performs several internal functions inside the frame controller. It resets the counter synchronously, de-asserts the READ, resets the flip flop DCE, and restarts the four bit shift register.

The first data word of the next frame actually appears on the FPDP bus twice, once before the header and once after the header in its proper place. The first appearance is not transferred on the bus, but is dis-allowed by negation of the handshake signal DVALID. This little bit of awkwardness is a consequence of the fact that data must be clocked out of the dual port buffer RAMs.

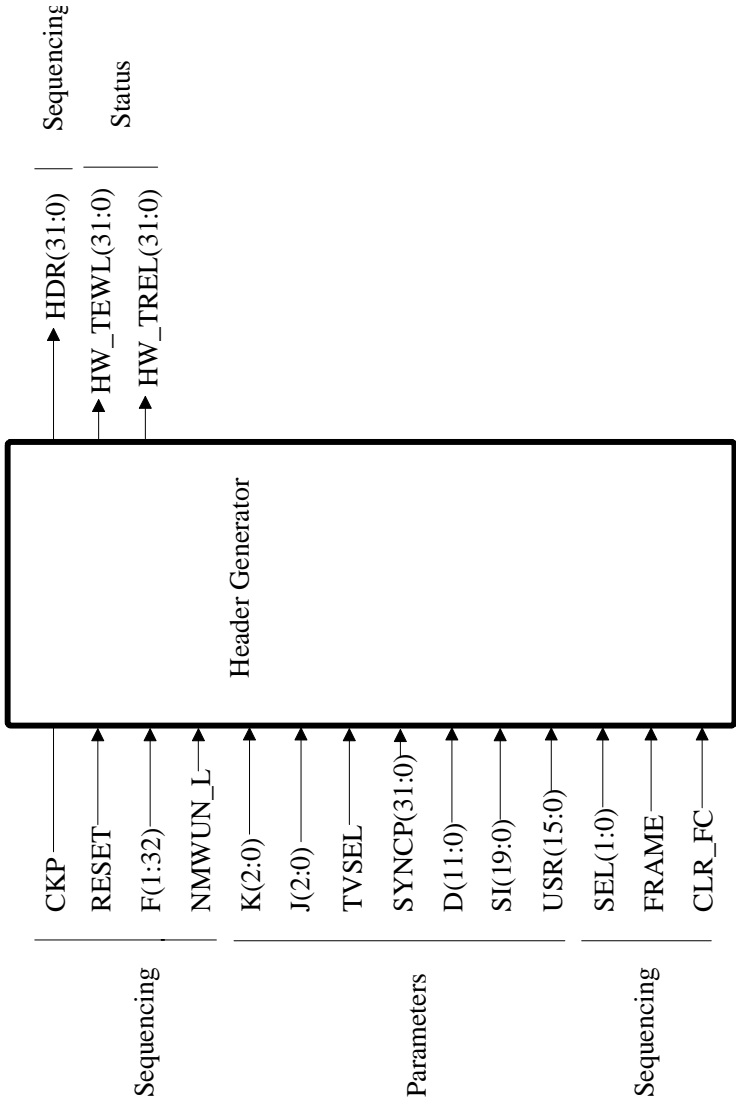
The sequence of header, 2500 data words, header, 2500 data words, ... continues until the STOP signal is given. The frame controller is not sensitive to the STOP until the 2500 data words of the current frame are transferred. The next header will not be produced as the STOP prohibits the four bit shift register from cycling.



**Figure 13**  
**Frame Controller Inside**



## HEADER GENERATOR



**Figure 14**

### Header Generator Block

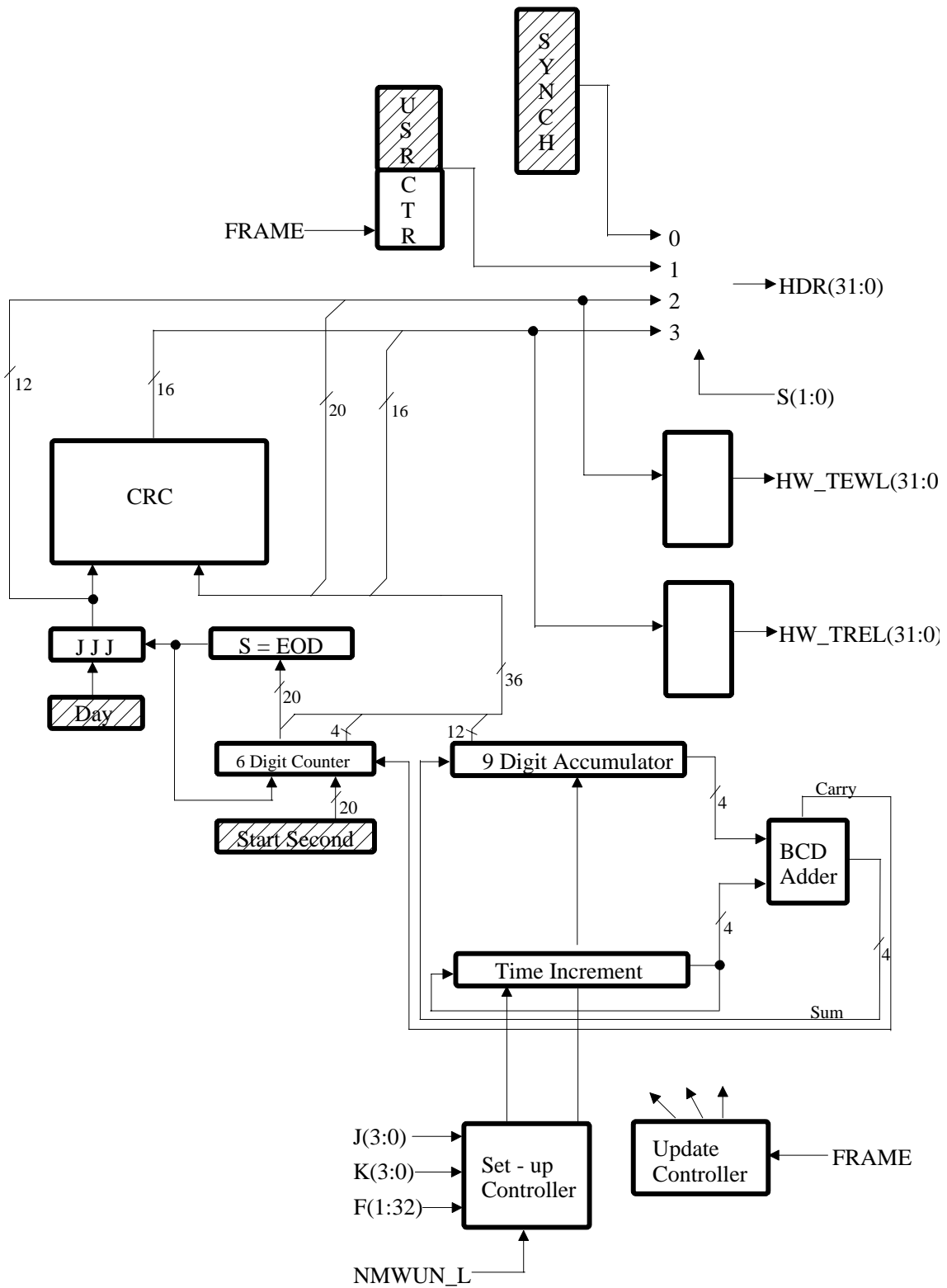
The header generator calculates the headers which are inserted into the data stream at the beginning of each 2500 word block of data. Headers consist of four 32-bit words whose content is specified in reference [2].

The block diagram of figure 14 identifies the inputs and outputs of the header generator. The specifics of the headers are functions of a number of parameters that are inputs in the center of the diagram in figure 14. K(3;0) and J(3;0) specify the sample frequency and the decimation count respectively. TVSEL is a flag which the software sets to cause the output sequence to be the TVG sequence. The fact that the data represents the TVG sequence is contained in a bit in each header. The first word in each header is the synch pattern which appears on the input SYNC(31:0). This pattern is set by the DIM to be 0xABADDEED and is not software changeable. D(11:0) is a three digit BCD number which specifies the modified Julian date of the start of the recording. SI(19:0) is a five digit BCD number which specifies the second of the start of the recording. Finally, USR(15:0) is a user specified pattern of 16 bits which is inserted into each header. Although it is expected that USR(15:0) will be constant during a recording, it is software accessible and could conceivably be changed from time to time during a recording. (If software elects to modify USR(15:0) during a recording it must be aware of the timing so that the modification does not occur at the time the header is actually being written on the FPDP bus. There is no hardware interlock to prevent errors in the event of a changing constant appearing on the FPDP bus.)

Sequencing inputs CKP, RESET, and F(1:32) are as found on other blocks on the output side of the DIM. NMWUN\_L is used to determine when the BSM decomposition is completed and the first header computation can be started.

FRAME and SEL(1:0) indicate that the header is being output and which word is currently being output. CLR\_FC allows the software to clear the frame counter.

HDR(31:0) is the output bus on which the headers appear, HW\_TEWL(31:0) and HW\_TREL(31:0) are output registers which are software accessible and contain the last two words of the most recent header.



**FIGURE 15**  
**HEADER GENERATOR ARCHITECTURE**

Figure 15 shows the architecture of the header generator. It is not quite as detailed as the other inside blocks of this document because it is a larger block and a detailed schematic is harder to follow.

The crosshatched values are those which are expected to be in place when the initial header is computed. The initial header is computed at the end of the BSM decomposition. This operation is done by the block entitled "Set-up Controller". It must know the values of  $K(3:0)$  and  $J(3:0)$  in order to compute the time increment from frame to frame. This increment is loaded into the "Time Increment" block which is a nine BCD digit shift register which re-circulates on itself. The "Nine Digit Accumulator" is initialized to zero and the start second value (five digits plus a trailing zero) is loaded into the "six Digit Counter. The frame counter is initialized to zero to complete the initialization. The final step in the computation of the first header is the computation of the CRC-16 which protects the 36 "time-bits" and the 12 "day-bits" which appear in the last two header words along with the CRC. The CRC block computes the CRC in a 16 bit at a time manner so the CRC is computed in three clock pulses of the CKP. Once that is done, the first header is assembled and ready to be output.

Each time a header is output (as determined by the frame controller) the header generator proceeds to compute the next header. This is governed by the block entitled "Update Controller" which detects the end of the FRAME signal. It then proceeds to add the time increment to the nine digit accumulator using the serial BCD adder to add one digit at a time. Once the nine digit accumulator is updated, the last carry out of the BCD adder is examined. If the carry is "1", the six digit counter is incremented. In the meantime, the frame count is incremented by the same FRAME signal which started the time update. Once the time update is complete the top five digits are examined for the count 86,400 (decimal). If that value is found in the counter, the top five digits are set to zero and the BCD counter which holds the day is incremented. Finally, a new CRC is computed over the new day and time bits to complete the new header.

LOCAL BUS INTERFACE

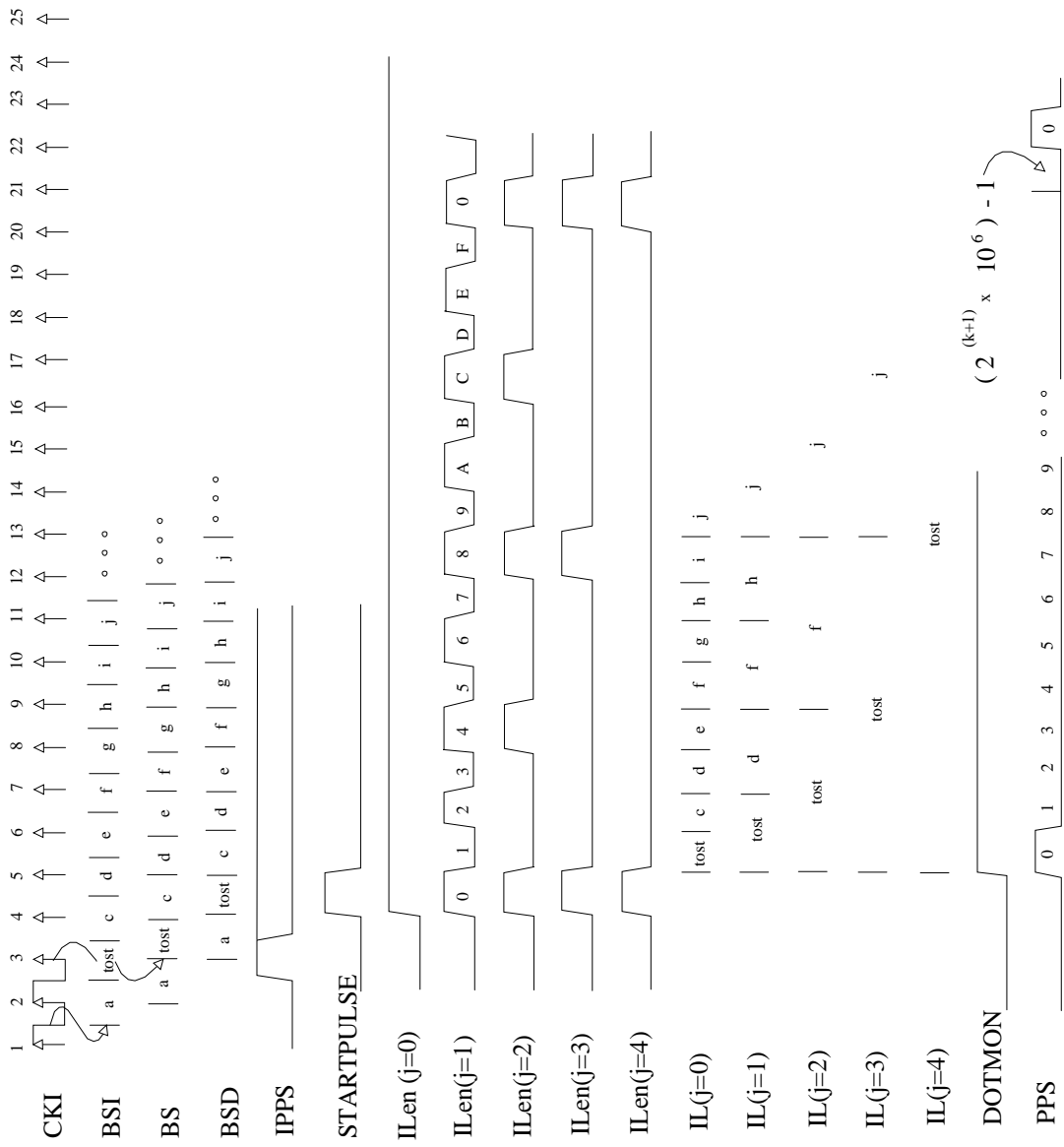
		Bits Within the Word																		
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
	0	-	-	REQ II	SEL DOT	SEL DIM	TVG	SEL TV/C	K2	K1	K0	J2	J1	J0	SEL PP1	SEL PP0	SEL CG			
	1	LED7 M	LED6 L	LED6 M	LED6 L	DIM CASC	EN VSI OUT	-	SPARE_L		-	-	-	-	ERF	CHK	GO COM			
	2	SPARE_H															SYNC 16			
Offset Address	3	IVP 15							IVP_WORD_L							IVP 0				
	4	IVP 31							IVP_WORD_H							IVP 16				
	5	BSM 15							BSM_WORD_L							BSM 0				
	6	BSM 31							BSM_WORD_H							BSM 16				
	7	USER 15							USER_WORD							USER 0				
	8	SI 15							START_TIME_L							SI 0				
	9	DI 11							START_TIME_H							DI 0	SI 19	SI 18	SI 17	SI 16
	A	-	-	-	-	-	CG WCK	CG R	CG UD	CG 7	CLOCK_GEN_BYTE						CG 0			
	B	RESET PPS	-	-	-	-	-	CLR PPSF	IN VAL	PV ENBL	PAUS	STRT STOP	-	SYNC PPS	SET UP	RESET				
R/W	C	TVRM 15							TVR_MASK_L							TVRM 0				
	D	TVRM 31							TVR_MASK_H							TVRM 16				
R	E	0	0	II	OF	ALT PPS	VSI PPS	SUNK	F_OK	REV 7	REV_BYTE						REV 0			
	F	HW2 15							HEADER_WORD_2_L							HW2 0				
	10	HW2 31							HEADER_WORD_2_H							HW2 16				
	11	HW3 15							HEADER_WORD_3_L							HW3 0				
	12	HW3 31							HEADER_WORD_3_H							HW3 16				
	13	0	0	0	DOT INT	DIM INT	0	0	0	0	0	0	0	0	0	0	0			

**FIGURE 16**  
**DIM PARAMETER TABLE**

The DIM parameters are described in detail by reference [3]. The table in figure 16 is presented to give a brief overview of the parameter space as seen by the software interface.

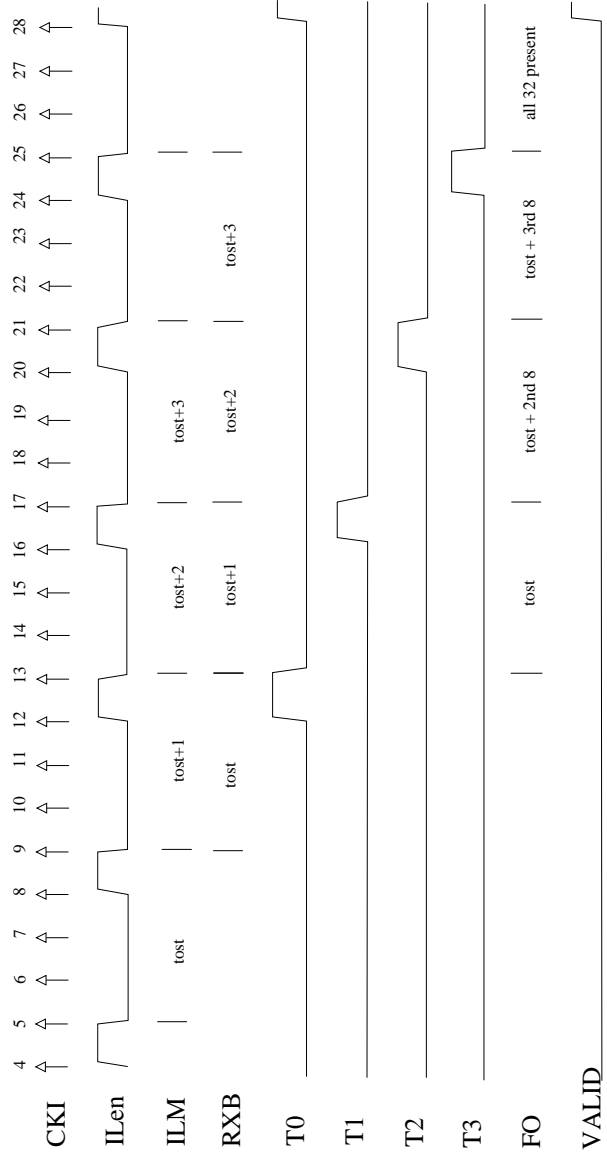
SEQUENCE EXAMPLES

The following set of sequence examples are presented for some particular cases and carry the data from the VSI input connector to the register preceding the FPDP bus. The sequences are cycle accurate and allow evaluation of pipeline delays.



**FIGURE 17**  
**INPUT SEQUENCES WITH DECIMATION**

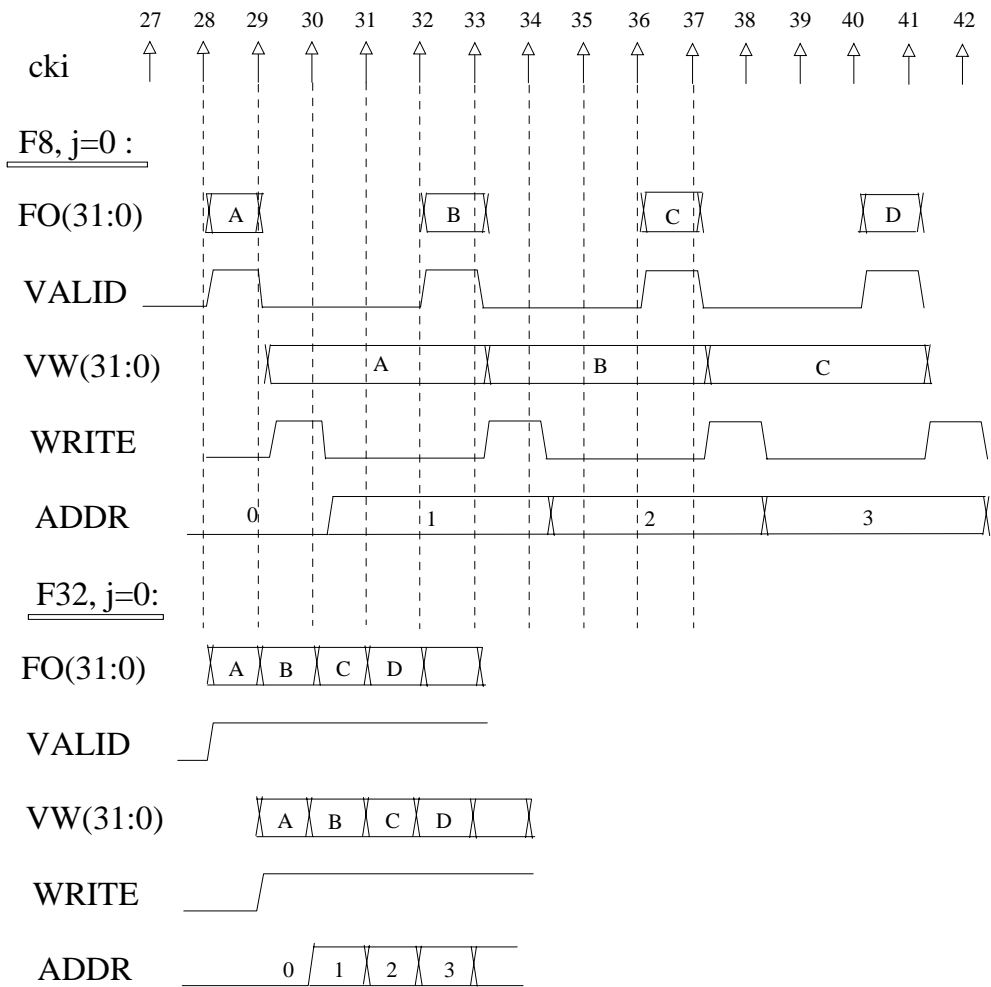
Figure 17 shows the effect of decimation on the input data sequence and the position of the TOST (Taken On Second Tick) bit for each decimation case.



**FIGURE 18**

**RXB TO FO FOR THE CASE F8, J=2**

Figure 18 shows the assembly of a 32 bit word in the fan out section for the case of F8, J=2.

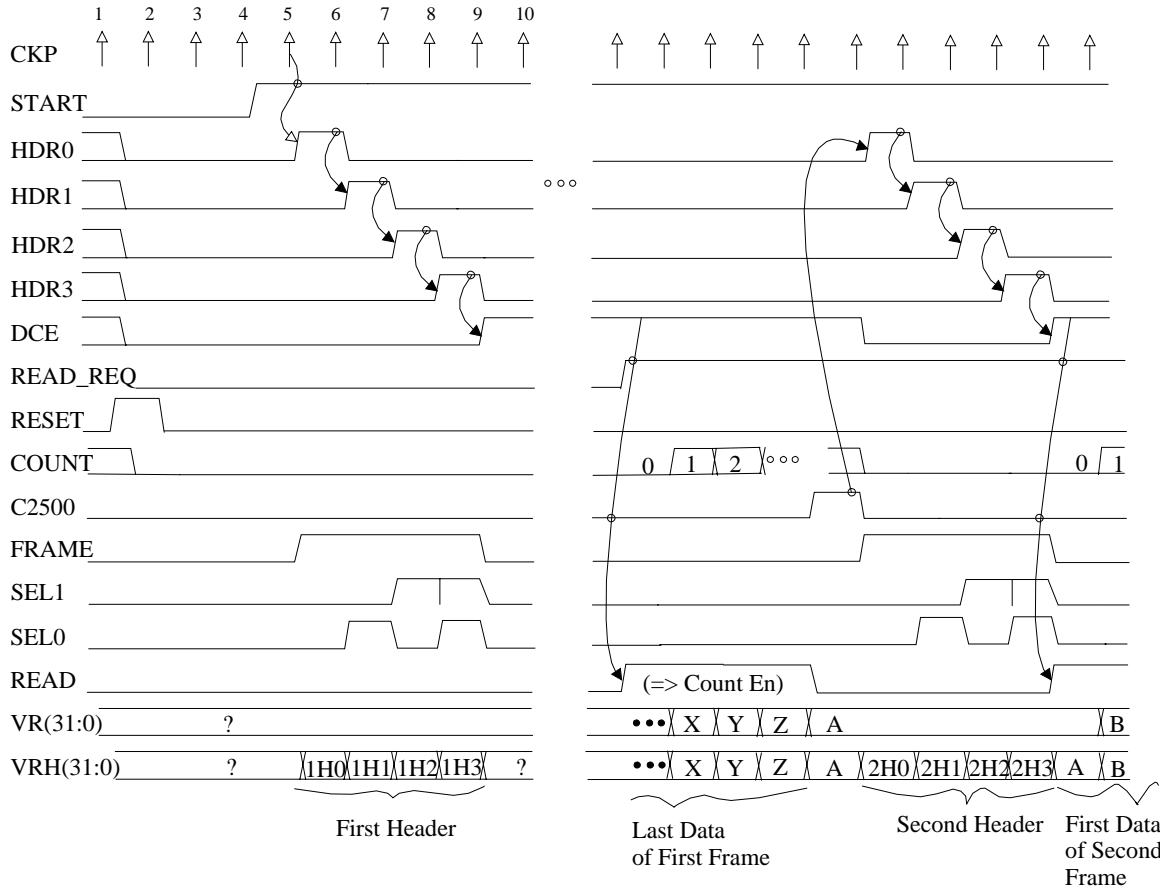


**FIGURE 19**

**FIFO INPUT SEQUENCES FOR TWO CASES OF F[] AND J**

Figure 19 shows the relation of the 32 bit words and the write address for two different cases.





**FIGURE 20**  
**FIRST AND SECOND HEADER SEQUENCES**

Figure 20 shows the sequencing of the first and second header sequences at the beginning of a scan. Note that the first header is sent out immediately on receipt of a start, but the data must await a READ\_REQ signal which is given once the FIFO has become  $\frac{3}{4}$  full and the FPDP bus is ready to receive the transfers. Note also that the time increment between headers as read from the time embedded in the headers is not the same as the actual time which elapses between the headers. The header time increments refer to time as the samples are taken, not to the times when the samples are placed on the disk.