SVT internal note N. 51

Hit Buffer Specifications and Design

Last revised: 15 September, 1996
Paola Giannetti, Franco Spinella

Contents

1 Introduction

The *Hit Buffer* (*HB*) is part of the *Silicon Vertex Tracker* (*SVT*), a trigger processor dedicated to the reconstruction of particle trajectories in the Silicon Vertex Detector (*SVX*) and Central Tracking Chamber (*CTC*) of the Collider Detector at Fermilab. A detailed description of the *Silicon Vertex Tracker* can be found in "SVT: Technical Design Report".

The overall architecture of SVT is shown in Figure 1. *Hit* coordinates from *Hit Finders* (*HF*) and *tracks* from the *CTC Track Finder* (*XFT*) are fed both into the *Associative Memory* (*AM*) and the *Hit Buffer*. The *Associative Memory* performs the first stage of a pattern recognition process and reconstructs particle trajectories with a limited spatial resolution. These coarse resolution track candidates are named *roads*. *Roads* are transmitted from the *Associative Memory* to the *Hit Buffer*, which associates each *road* with the proper *hit* coordinates and XFT *tracks* and transmits them to the *Track Fitters* (*TF*) for the final and refined track finding process.

The *Hit Buffer* shall meet the functional and hardware requirements specified in this document.

2 Communications

Data flow in and out the *Hit Buffer* as data streams. Each stream enters or leaves the *Hit Buffer* through a dedicated connector and cable. There are two input streams — called respectively *hit* stream and *road* stream — and one output stream — called *road-info* stream.

2.1 SVT Protocol

A uniform communication protocol is used for all data transfers throughout the SVT system. Data flow through unidirectional links connecting one source to one destination. The protocol is a simple pipeline transfer driven by an asynchronous *Data Strobe* (DS_ in the following text, it is an active low signal). To maximize speed, no handshake is implemented on a word by word basis. A *Hold* signal is used instead as a loose handshake to prevent loss of data when the destination is busy (it is an active low signal and will be called HOLD_ in the following text). Data words are sent on the cable by the source and are strobed in the destination at every positive going DS_ edge. The DS_ is driven asynchronously by the source. Correct DS_ timing must be guaranteed by the source (more details on the data transfer timing are given later in this document).

Input data are pushed into a FIFO buffer. The FIFO provides an *Almost Full* signal that is sent back to the source on the *HOLD_* line. The FIFO is popped by whatever processor sits in the destination device. If the destination processor does not keep up with the incoming data rate, the FIFO becomes *Almost Full* and the *HOLD_* signal is sent to the source. The source responds to the HOLD_ signal by suspending the data flow. Using *Almost Full* instead of Full gives the source plenty of time to stop (the equivalent of 127 DS_ cycles).

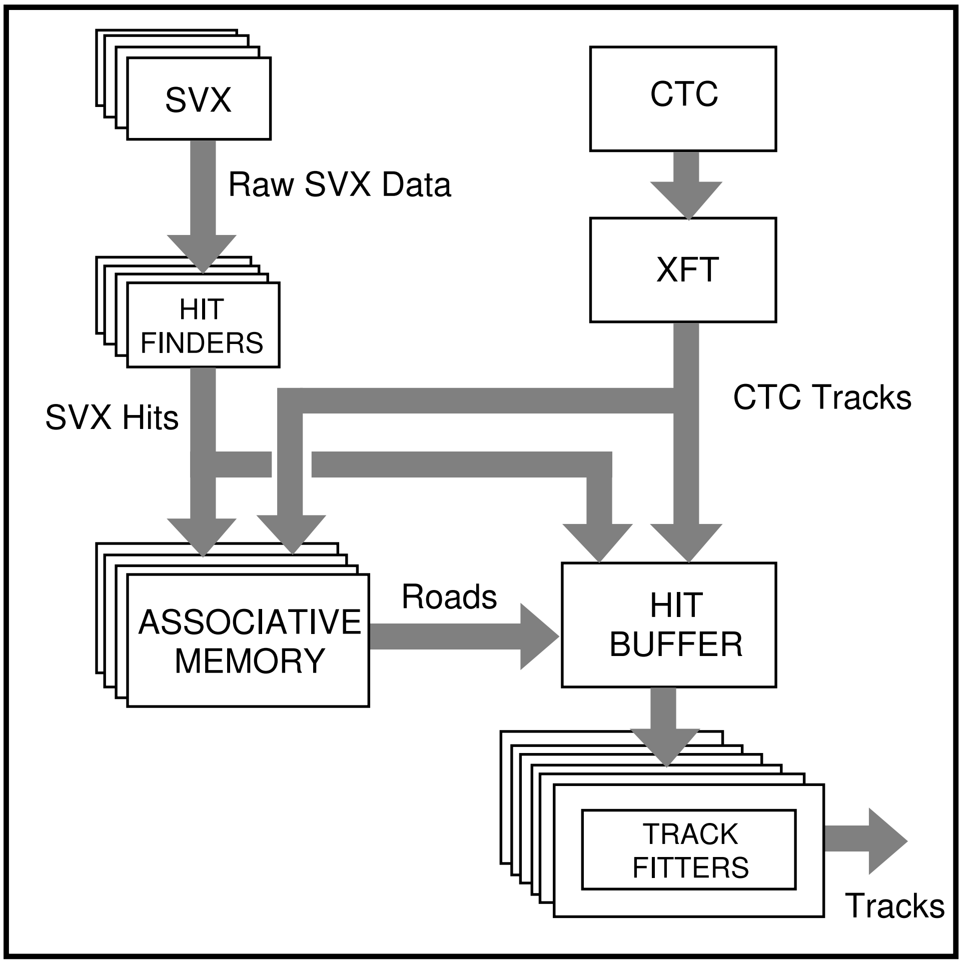

Figure 1: SVT architecture

The source is not required to wait for an acknowledge from the destination device before sending the following data word, allowing the maximum data transfer rate compatible with the cable bandwidth even when transit times are long.

Signals are sent over flat cable as differential TTL.

The maximum DS_ frequency is 30 MHz.

2.2 Packets and Events

On each cable there are 21 data bits, *Data Strobe* (DS_), *HOLD_*, *End Packet* (*EP*), *End Event* (*EE*), for a total of 25 signals (50 wires). Data are sent as packets of words. The first word of each packet is called *head*. In the simplest case a packet may consist of a single word — the *head* — otherwise, if more than 21 bits are needed, a packet may consist of two or more words. The words following the *head*, in a multi-word packet, are collectively called *body*. The EP bit marks the last word of each packet — the *End Packet* word. In one-word packets, of course, the word has EP=1.

The EE bit is used to mark the end of the data stream for the current event. The complete sequence of words in a data stream is called an *Event*. The *Hit Buffer* will assert EE on the output stream after it has received an EE in each input stream and has no more data to output. The last word of each Event — the *End Event* word — has a special format. It has EE=1 and EP=1. The data field in the *End Event* word is used for Event Tag (8 bits), Parity (1 bits), and Error Flags (12 bits), as shown in Table 1. These bits are processed by the *Hit Buffer* as specified below under "Exception Handling".

bit range	20-9	8	7-0
signal name	Error Flags	Parity	Event Tag

Table 1: Data field in the *End Event* word

3 Hit Buffer Data

3.1 Input data

In the *hit* stream, driven by the *Hit Finders* and the *CTC Track Finder*, each packet is called a *hit*. Each word contains a *hit* coordinate in the data field (21 bits). *Hits* may be one word long or more depending on what kind of detector the *hit* is coming from (SVX or XFT). The Layer number (3 MSb) is contained only in the packet head. XFT *hits* are also called *tracks*.

In the *road* stream, driven by the *Associative Memory*, each packet is called a *road*. All *roads* are one word in length. The 17 LSb of the data field are the *road* number and the 4 MSb are the ϕ sector. Internally, the *Hit Buffer* does not use the ϕ sector, but it passes this information to the board downstream.

HF & XFT → AM & HB

<table>
<tr><td>2</td><td>1</td><td>0</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>bit number</td></tr>
<tr><td></td><td>EP</td><td colspan="3">layer(3)</td><td colspan="3">barrel(3)</td><td colspan="15">abscissa(15)</td><td>SVT hit</td></tr>
<tr><td></td><td>EP</td><td colspan="3">layer(3)</td><td colspan="3">barrel(3)</td><td colspan="15">abscissa(15)</td><td>SVT hit</td></tr>
<tr><td></td><td>EP</td><td colspan="3">layer(3)</td><td colspan="3">barrel(3)</td><td colspan="15">abscissa(15)</td><td>SVT hit</td></tr>
<tr><td></td><td></td><td colspan="3">layer(3)</td><td colspan="13">ϕ(13)</td><td colspan="5">Pt(5)</td><td>XFT track</td></tr>
<tr><td></td><td></td><td colspan="21">additional info(21)</td><td></td></tr>
<tr><td></td><td>EP</td><td colspan="21">additional info(21)</td><td></td></tr>
<tr><td></td><td></td><td colspan="3">layer(3)</td><td colspan="13">ϕ(13)</td><td colspan="5">Pt(5)</td><td>XFT track</td></tr>
<tr><td></td><td>EP</td><td colspan="21">additional info(21)</td><td></td></tr>
<tr><td></td><td>EP</td><td colspan="3">layer(3)</td><td colspan="3">barrel(3)</td><td colspan="15">abscissa(15)</td><td>SVT hit</td></tr>
<tr><td>EE</td><td>EP</td><td colspan="12">Error Flags(12)</td><td>PA</td><td colspan="8">Event Tag(8)</td><td>end event</td></tr>
</table>

AM → HB

<table>
<tr><td>2</td><td>1</td><td>0</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>bit number</td></tr>
<tr><td></td><td>EP</td><td colspan="4">ϕ sector(4)</td><td colspan="17">road number(17)</td><td></td></tr>
<tr><td></td><td>EP</td><td colspan="4">ϕ sector(4)</td><td colspan="17">road number(17)</td><td></td></tr>
<tr><td></td><td>EP</td><td colspan="4">ϕ sector(4)</td><td colspan="17">road number(17)</td><td></td></tr>
<tr><td>EE</td><td>EP</td><td colspan="12">Error Flags(12)</td><td>PA</td><td colspan="8">Event Tag(8)</td><td>end event</td></tr>
</table>

3.2 Output data

In the output stream, packets are variable in length. Each packet, called a *road-info*, contains a variable number of *hits* and is terminated by an End Packet word (EP=1) that contains a *road*. The output format of the *hits* is identical to the input format described above with EP bits removed. The output format of the *road* is identical to the input format.

3.3 Data processing

Within the *Hit Buffer*, *hits* are stored in the Hit List Memory (HLM). A structured data base is built on the fly so that each *road* number can then be used as a key to directly access lists of *hits*. Incoming *hits* are sorted into a number of classes according to coordinate value ranges. These classes are called SuperStrips. One *hit list* is maintained for each SuperStrip, all these lists are built on the fly in the *Hit Buffer* as the *hits* are received on input.

The *Hit Buffer* works as a well organized library, where books arrive in arbitrary order and are quickly arranged on bookshelves by basic subjects — one shelf for each subject. The librarian also maintains a list of complex, interdisciplinary topics. Each complex topic (e.g. "CMOS VLSI Design") points to several specific basic subjects (e.g. "MOS transistor", "VLSI technology", "Logic design", and "System design"). Usually, the same basic subject will be pointed by many different topics. When a complex topic is later requested by the user, the librarian looks at the basic

HB → TF

<table>
<tr><td>2</td><td>1</td><td>0</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>bit number</td></tr>
<tr><td></td><td></td><td colspan="3">layer(3)</td><td colspan="3">barrel(3)</td><td colspan="15">abscissa(15)</td><td>SVT hit</td></tr>
<tr><td></td><td></td><td colspan="3">layer(3)</td><td colspan="3">barrel(3)</td><td colspan="15">abscissa(15)</td><td>SVT hit</td></tr>
<tr><td></td><td></td><td colspan="3">layer(3)</td><td colspan="3">barrel(3)</td><td colspan="15">abscissa(15)</td><td>SVT hit</td></tr>
<tr><td></td><td></td><td colspan="3">layer(3)</td><td colspan="13">ϕ(13)</td><td colspan="5">Pt(5)</td><td>XFT track</td></tr>
<tr><td></td><td></td><td colspan="21">additional info(21)</td><td></td></tr>
<tr><td></td><td>EP</td><td colspan="4">ϕ sector(4)</td><td colspan="17">road number(17)</td><td>trailer</td></tr>
<tr><td></td><td></td><td colspan="3">layer(3)</td><td colspan="13">ϕ(13)</td><td colspan="5">Pt(5)</td><td>XFT track</td></tr>
<tr><td></td><td></td><td colspan="21">additional info(21)</td><td></td></tr>
<tr><td></td><td></td><td colspan="3">layer(3)</td><td colspan="3">barrel(3)</td><td colspan="15">abscissa(15)</td><td>SVT hit</td></tr>
<tr><td></td><td>EP</td><td colspan="4">ϕ sector(4)</td><td colspan="17">road number(17)</td><td>trailer</td></tr>
<tr><td>EE</td><td>EP</td><td colspan="12">Error Flags(12)</td><td>PA</td><td colspan="8">Event Tag(8)</td><td>end event</td></tr>
</table>

subjects pointed by the topic and all books on these subjects are quickly found and delivered.

The book corresponds to the *hit*, the basic subject to the SuperStrip, the complex topic to the *road*, and the set of bookshelves to the Hit List Memory. During the "Write Mode", *hits* are copied from the input and immediately organized in the Hit List Memory, depending on the SuperStrip they belong to. *Roads* are made of SuperStrips (one SuperStrip per layer). During the "Read Mode", each *road* received by the *Hit Buffer* requires that the Hit Lists corresponding to that *road* are sent from the Hit List Memory on the output stream, one SuperStrip (i.e. one layer) after the other.

To provide a very fast access to the Hit List Memory, the *Hit Buffer* has large look-up tables where pointers into the Hit List Memory are stored. The "SuperStrip Map" (SS Map) provides the starting address of the Hit List where each *hit* must be written when it arrives (Write Mode), while the "Associative Memory Map" (AM Map) provides the starting addresses of the Hit Lists to be sent on output for each *road* (Read Mode). The contents of the SS Map and of the AM Map are essentially the same, but the two maps implement different data base structures with different addressing schemes.

These look-up tables work in the following way:

- **SS Map: from *hits* to SuperStrips**

  The SuperStrip each *hit* belongs to is determined by the layer number and the value of the coordinate in the packet head. Possible body words, containing additional coordinates, will be ignored for this purpose. The mapping from coordinate to SuperStrip is defined by a look-up table — the SS Map — implemented as a RAM where the 17 MSb of the coordinate are used as an address and the corresponding "SuperStrip ID" is stored in each location.

  Two informations are encoded in the SuperStrip ID:

  1. the starting address of the relevant Hit List in the Hit List Memory (base address),

2. the space reserved for the Hit List of that SuperStrip in the Hit List Memory (size).

Referring to the first information, we say that the SuperStrip ID "points" to a memory location, as shown in Table 2. The SuperStrip ID code is described

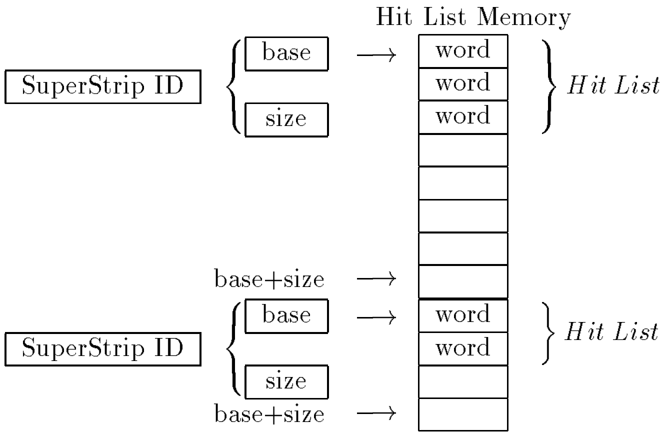

Table 2: SuperStrip ID and Hit List structure in the Hit List Memory. Two informations are encoded in the SuperStrip ID: base and size. Base is the starting address of the memory space reserved for the SuperStrip. Size is the number of reserved memory cells.

in Section 8. A word counter exists for each SuperStrip to remember the current length of the corresponding Hit List. All these counts are stored in a separate *Hit Count* block described in Section 8.

- **AM Map: from Roads to SuperStrips**

  Each *road* defines an array of SuperStrips, one SuperStrip per detector layer — we say that a *road* is made of a number of SuperStrips. The number of layers is determined by the physical structure of the detector and the *Hit Buffer* is capable of accommodating from one to seven layers. *Roads* are identified by integer numbers. The correspondence between *road* number and SuperStrips is defined by a look-up table — the AM Map — implemented as a RAM where *road* number and Layer number are used as an address and the corresponding SuperStrip ID is stored in each location.

  When a *road* number is received, it is used to access the look-up table and sequentially retrieve the pointers to all Hit Lists corresponding to the relevant SuperStrips, one per detector layer.

  These pointers are used to access the corresponding lists. All *hits* retrieved in this way are queued to form an output packet. The last word of the packet contains the *road* number.

4 Hit Buffer Architecture

The main points about the *Hit Buffer* architecture are shown in Figure 2 and listed below:

- the *Hit Buffer* is synchronized by an internal clock signal, all state transitions happen on the clock rising edge.
- Input data are asynchronously loaded by upstream modules into two FIFO buffers: Hit-FIFO and Road-FIFO. These FIFO's are read via two FIFO Controllers that convert the asynchronous FIFO protocol to the *Hit Buffer* internal synchronous pipeline protocol. This protocol and the FIFO Controller are described in detail in another document (SVT note 34).
- After initialization the *Hit Buffer* is in Write Mode, it turns into Read Mode after *End Event* is received on the *hit* input stream, and it goes back to Write Mode after *End Event* is received on the *road* input stream and sent on output to the next board. In the design Write Mode is distinguished by WR flag = 1 and Read Mode by WR flag = 0. During normal operation the *Hit Buffer* starts each event in Write Mode, switches after a while to Read Mode and stays there until the end of the event, then clears all hit lists and switches to Write Mode again.
- Data-flow from the two input FIFO's to the output connector is coordinated by a Finite State Machine (FSM). The FSM requests data from the input FIFO's and then latches them in an internal latch — called Middle Latch or ML in the following — which separates the look-up tables (SS Map and AM Map) from the Hit List Memory. Middle Latch is subdivided in two sections: ML DATA and ML ADD. When in Write Mode, ML DATA contains the *hit* to be written in the Hit List Memory and ML ADD contains the SuperStrip ID with the pointer to the Hit List where it will be written (this information is fetched from SS Map). When in Read Mode, ML DATA contains the *road* number and ML ADD contains the SuperStrip ID with the pointer (this information is fetched from AM Map) to the Hit List to be fetched and sent on the output stream.

The *Hit Buffer* can therefore be thought of as a 4 stage pipeline (Figure 2):

1. from input FIFO's to FIFO REGISTER under control of the FIFO Controllers HFCTR and RFCTR (see SVT note 34).
2. from FIFO REGISTER to Middle Latch, via SS Map and AM Map, under control of FSM.
3. from Middle Latch to OUT REGISTER, via the Hit List Memory, under control of FSM. Addressing of the Hit List Memory is done with the help of the Hit Count and of the Hit List Control logic (HLC).
4. from OUT REGISTER to the input FIFO of the next board downstream, under control of FSM.

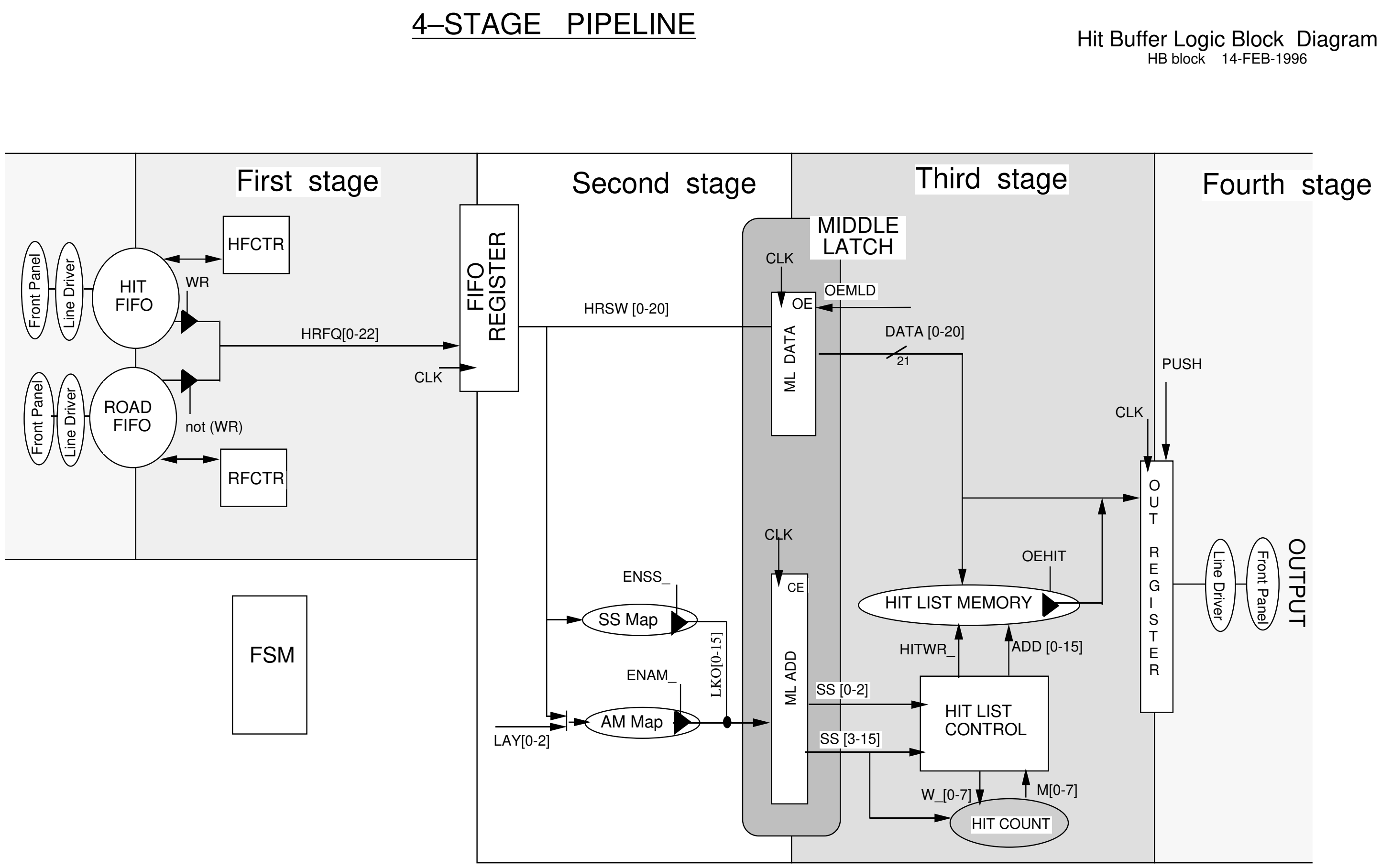

Figure 2: *Hit Buffer* architecture

4.1 A few rules and notes

In the next sections we explain the way in which data flow among the different components of the *Hit Buffer*, the behaviour of the Finite State Machine that controls such flow, and the details of all components. The following rules apply in what follows.

1. All numerations (bits, bus lines...) start from 0.

2. A signal means a wire. Therefore it has a unique name which has to be the same in the schematics, in the simulation, and in the documentation. Such a name is UPPERCASE. To make the name more meaningful, lower case characters may be added freely whenever allowed by space, compilers, etc.

3. The logical operator NOT is indicated by either ".NOT." or "!". The logical operator AND is indicated by either ".AND." or "*". The logical operator OR is indicated by either ".OR." or "+".

4. The polarity of control signals is often indicative. In actual implementation, control signals may be turned from high-active to low-active ones (or viceversa) if this is called for by the components actually chosen as registers and RAM's.

5. By default all signals have positive (high-active, 1=TRUE) logic. Negative logic (low-active, 0=TRUE) signals will be flagged by putting an underscore (_) at the end of the name. In situation that do not allow underscores (e.g. ABEL) the letter "X" will be used.

5 Data-flow

5.1 Write Mode

Figure 3 shows data-flow in the *Hit Buffer* during the Write mode.

In this schematic drawing, all signals entering from or exiting to the sides of the figure are connected to the FSM and are described in Section 6. The box labeled "INPUT FIFO 1" represents the "Hit part" of the input logic described above, including the FIFO REGISTER.

21-bit data words — the *hit* coordinates — are read from the Hit-FIFO under the HPOP-HPOK protocol (SVT note 34) handled by the FSM. The 17 MSb of the data word in each packet head are sent as address to the SS Map, whose output, the SuperStrip ID, is latched in ML ADD controlled by LEADD. The packet head is identified as the first word after an End Packet word, which is in turn signaled by the flag LEP. The packet body, if any, must be propagated to the Hit List Memory without changing the SuperStrip ID stored in ML ADD.

The full 21-bit data words, including those in the packet body, are also latched in ML DATA, controlled by LEDATA, as data for the Hit List Memory. The address for the Hit List Memory is generated by the Hit List Control on the basis of two informations: the SuperStrip ID stored in ML ADD and the content of the Hit Count. In one clock cycle a read-modify-write operation is performed on the Hit

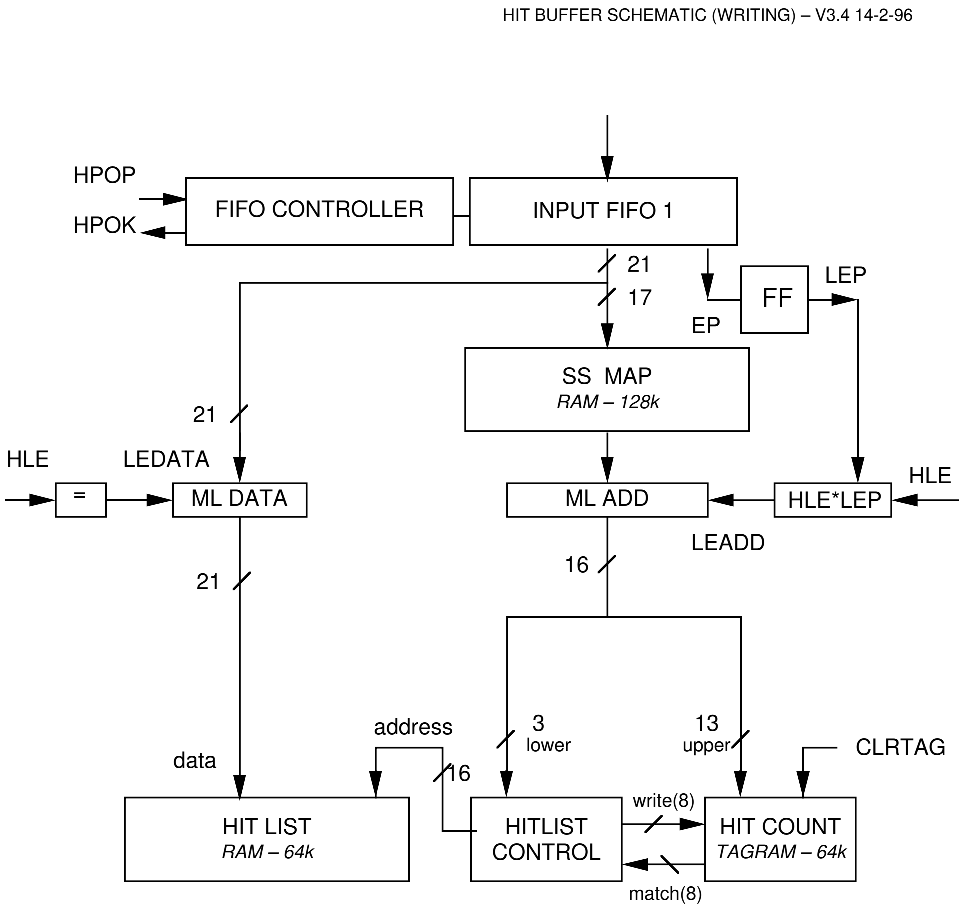

Figure 3: *Hit Buffer* data-flow: Write mode

Count by the Hit List Control — the number of existent words in the current Hit List is read, incremented by 1, and written back. In the next clock cycle the data word is written into the Hit List Memory. The Hit List Control and the Hit Count are described in Section 8. The Write Mode ends when the EE bit goes up in the input *hit* stream.

5.2 Read Mode

Figure 4 shows data-flow during the Read Mode.

In this schematic drawing, all signals entering from or exiting to the sides of the figure are connected to the FSM and are described in Section 6. The box labeled "INPUT FIFO 2" contains the *road* related part of the input logic, including the FIFO REGISTER. The multiplexer MUX indicated at the bottom is the logical representation of what is more succinctly represented in the diagram of Figure 2, where the DATA[0-20] bus combines the 3-state outputs of ML DATA and Hit List Memory.

21-bit data words — the *roads* — are read from the Road-FIFO under the RPOP-RPOK protocol (SVT note 34) controlled by the FSM. They are latched in ML DATA, controlled by LEDATA, so that they are ready to be sent on the output stream at the end of the *road-info* packet. The 17 LSb are the *road* number (the 4 MSb are the ϕ sector) and make up the least significant part of the address to the AM Map. The 3 MSb of such address are the Layer number and come from COUNTER1 — a counter controlled by FSM through the signals INCR1 and CLR1. After a valid word is read from the Road-FIFO, the AM Map output (i.e. a SuperStrip ID) is latched into ML ADD, controlled by LEADD. The SuperStrip ID is kept in ML ADD until all *hits* belonging to that SuperStrip are sent out, at which time the FSM increments the layer counter (sending INCR1) and latches a new SuperStrip ID. This process goes on until the last layer is reached. This condition is flagged by the LAST1 signal being asserted by the comparator CMP1 which compares the current layer number with a predetermined number of layers NLAY[0-2]. NLAY[0-2] is set, within the range 0 to 7, by a dip-switch located on the board. In most cases, there will be only one word in each Hit List, therefore a new word will be latched in ML ADD per clock cycle.

The SuperStrip ID stored in ML ADD is used to address all *hits* belonging to that SuperStrip and stored in the Hit List Memory. The addresses of successive *hit* words are generated by the Hit List Control on the basis of two informations: the SuperStrip ID and the output of COUNTER2. COUNTER2, controlled by FSM through the signals INCR2 and CLR2, counts the number of words read out from a Hit List. The count limit for this counter, stored in the Hit Count, is retrieved and tested by the Hit List Control. Hit List Control generates the EMPTY signal if no *hits* are in the current Hit List, and LAST2 once all *hits* have been processed, i.e. when the content of COUNTER2 is equal to the number of *hit* words in the current Hit List.

One clock cycle is lost in case of zero *hits*, in which a valid LAST2 is also asserted, as if there was only one *hit*. EMPTY is therefore used to prevent sending any output word (disabling PUSH assertion).

The data out of the Hit List Memory are sent on the output stream, one word

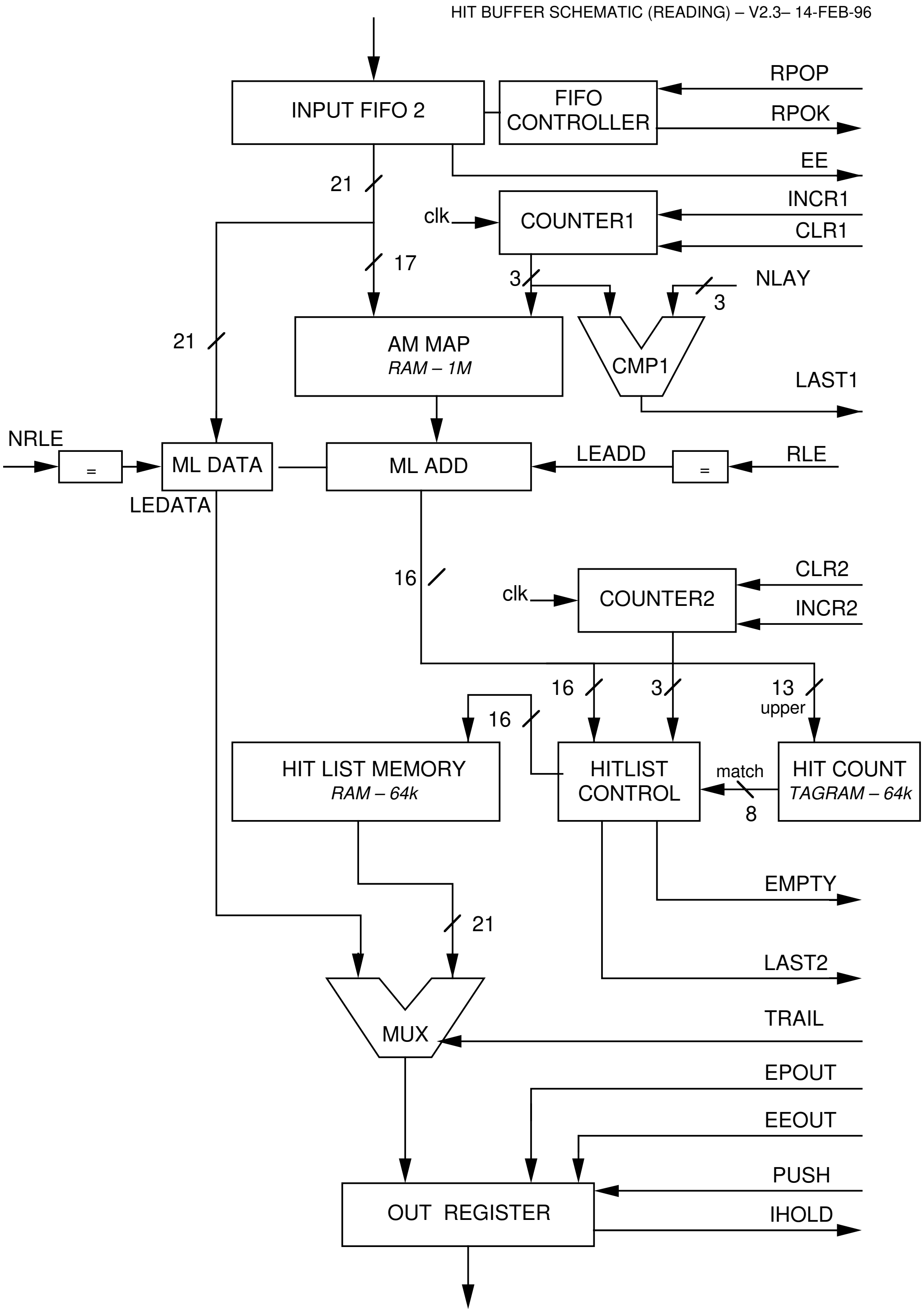

Figure 4: *Hit Buffer* data-flow: Read mode

per clock cycle, by asserting PUSH. PUSH assertion is disabled when HOLD_ is received from the output cable and internally signaled by IHOLD.

When the FSM senses the LAST2 related to the last layer (i.e. LAST2 * LAST1), it understands that it has finished processing one *road* and appends the *road* identifier, stored in ML DATA, to the output stream, by asserting TRAIL, EPOUT, and PUSH (this is the last word of a *road-info* packet).

In response to the *End Event* on the *road* stream, signaled by EE, the FSM asserts EEOUT to append an *End Event* word to the output stream. The content of the *End Event* word written into the OUT REGISTER is obtained from the *End Event* REGISTER, where the Event Tag, Parity, and Error Flags have been stored (see Sections 10 and 10.1 for a description of the *End Event* REGISTER).

6 Finite State Machine

Figure 5 shows the state diagram for the FSM and the equations that define the values of its output signals.

The FSM has a 4-bit state register, as shown in the following table:

FSM state bit	3	2	1	0
bit name	CLRTAG	TRAIL	READY	SMQ0

The FSM is asynchronously reset by the FSMINIT signal, which is the logical OR of INIT and TMOD (see Section 13).

After FSMINIT, the FSM seats in state 0 waiting for data in the input Hit-FIFO. State transitions happen on clock rising edge, the next state is prepared according to the current state and to the status of the input lines, as indicated by arrows. The outputs are combinatorial logic functions of the machine state and of the inputs. In Figure 5, state dependent parts of the assignments are indicated inside the relevant state box (e.g. EEOUT is asserted while in state S12). State independent parts of the assignments are indicated in the top left box labeled "All States" and are meant to be logically OR'ed (indicated with $+\ldots$) with the state dependent part derived from the state diagram. E.g., the full logic expression for the CLR2 output is: CLR2 = NEWLAY .OR. (.NOT.IHOLD .AND. State=4).

6.1 FSM inputs

EE Source: FIFO REGISTER. Explanation: "End Event" from either input stream, bit 22 from FIFO REGISTER (i.e. 23rd bit).

EMPTY Source: M-LINES ENCODER in Hit List Control. Explanation: This signal means that the SuperStrip currently present at the output of ML ADD does not have any *hit*.

IHOLD Source: HOLD SYNC. Explanation: "Internal HOLD" is the asynchronous external HOLD_ from the output connector, syncronized by a flip-flop. It signals a request to halt data pushing from the downstream board. IHOLD is

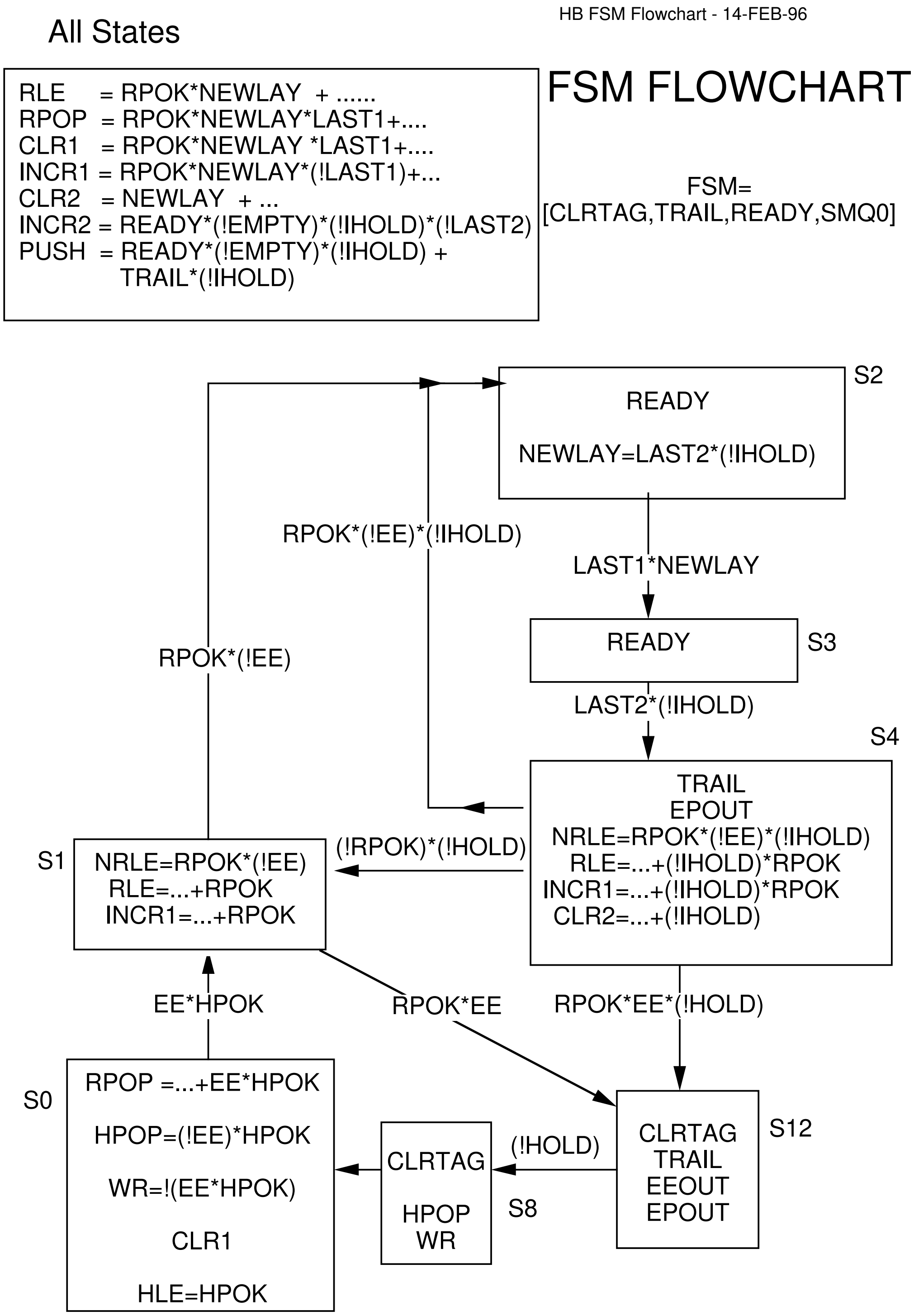

Figure 5: *Hit Buffer* Finite State Machine state diagram

valid during the full clock cycle (synchronous signal) and can be tested at any time. Suspension of data pushing may be delayed by a few clock cycles after detection of HOLD_.

HPOK Source: Hit-FIFO Controller HFCTR. Explanation: "Hit Pop OK" means that data at the output of FIFO REGISTER are valid and can be latched at the next clock rising edge. More details in SVT note 34.

LAST1 Source: Comparator CMP1. Explanation: this signal means that the layer currently being processed is the last in a *road*.

LAST2 Source: Comparator CMP2 in Hit List Control. Explanation: LAST2 means that the current output of the Hit List Memory, i.e. the word that will be latched in OUT REGISTER at the next clock rising edge, is the last word in the current Hit List.

LEP Source: EP Flip Flop. Explanation: "Late EP bit" signals the End Packet word with a one-cycle delay.

RPOK Source: Road-FIFO Controller RFCTR. Explanation: "Road Pop OK" means that data at the output of FIFO REGISTER are valid and can be latched at the next clock rising edge. It has the same meaning as HPOK, but for the *road* stream. More details in SVT note 34.

6.2 FSM outputs

CLR1 Destination: COUNTER1. Explanation: "CLeaR1" clears the content of COUNTER1 at next clock rising edge.

CLR2 Destination: COUNTER2. Explanation: "CLeaR2" clears the content of COUNTER2 at next clock rising edge.

CLRTAG Destination: Hit Count. Explanation: " CLeaR TAG ram" clears the Hit Count contents all at once, at the end of the Read Mode. It must be asserted for two consecutive clock cycles since the Hit Count is implemented with chips IDT71B74 (see Section 8), which require a reset pulse width greater than 50 nsec.

EEOUT Destination: OUT REGISTER. Explanation: "End Event OUTput bit" to be registered in the output word at the next clock rising edge.

EPOUT Destination: OUT REGISTER. Explanation: "End Packet OUTput bit" to be registered in the output word at the next clock rising edge. EPOUT is actually equivalent to TRAIL (i.e. EPOUT = TRAIL), but for clarity we continue to use both names.

HLE Destination: AUXILIARY LOGIC (see Section 7), HIT LIST CONTROL and EE REGISTER. Explanation: a new word is available in the FIFO REGISTER from the *hit* stream. "Hit Latch Enable" enables the FIFO REGISTER output to be latched into ML DATA and the SS Map output to be latched into ML ADD. It is also needed by the Hit List Control and is one of the control lines CTRLINES to the EE REGISTER.

HPOP Destination: Hit-FIFO controller HFCTR. Explanation: "Hit POP" requests a new word from the Hit-FIFO. Requested data (if available) will be latched into FIFO REGISTER at the next clock rising edge. See SVT note 34 for more details.

INCR1 Destination: COUNTER1. Explanation: "INCRement1" increments by 1 the content of COUNTER1 at next clock rising edge.

INCR2 Destination: COUNTER2. Explanation: "INCRement2" increments by 1 the content of COUNTER2 at next clock rising edge.

NRLE Destination: AUXILIARY LOGIC and EE REGISTER. Explanation: a new word is available in the FIFO REGISTER from the *road* stream. "New Road Latch Enable" enables the FIFO REGISTER output to be latched into ML DATA. It is also one of the control lines CTRLINES to the EE REGISTER.

PUSH Destination: OUT REGISTER, DS_ GEN. Explanation: "PUSH" sends one word on the output stream.

RLE Destination: AUXILIARY LOGIC and EE REGISTER. Explanation: "Road Latch Enable" enables the AM Map output to be latched into ML ADD. It is also one of the control lines CTRLINES to the EE REGISTER.

RPOP Destination: Road-FIFO controller RFCTR. Explanation: "Road POP" requests a new *road* from the Road-FIFO. Requested data (if available) will be latched into FIFO REGISTER at the next clock rising edge. See SVT note 34 for more details. RPOP has the same meaning and use as HPOP, but refers to the *road* stream.

TRAIL Destination: OUT REGISTER Explanation: "TRAIL" indicates whether the word to be sent in output is either a *hit* word (TRAIL = 0) or an End Packet / *End Event* word (TRAIL = 1). TRAIL is equal to EPOUT. TRAIL is one of the FSM state bits.

6.3 FSM internal signals

There are a few signal in Figure 5 that are neither external inputs nor outputs, they are convenient names for signals generated and used internally by the FSM, defined in order to simplify the diagram.

NEWLAY Used in Read mode only. Means that the FSM wants the SuperStrip address corresponding to the next layer in the current *road* to be latched in ML ADD at the next clock rising edge.

READY Is one of the FSM state bits. It is also used in Read Mode to indicate that valid data are present at Middle Latch output. Data are valid while READY is asserted.

WR Explanation: "WRite mode" defines the *Hit Buffer* to be in Write Mode (WR = 1) or Read Mode (WR = 0).

7 FSM Auxiliary Logic

The Auxiliary Logic is a slave hardware sharing all the FSM signals (inputs, outputs and internal signals) to generate control lines.

The Auxiliary Logic receives also inputs from other pieces of hardware, like the VME slave or the Fifo Controllers, whose actions need to be combined with the main FSM actions to generate the global control lines. The multiplexing between the VME and the *Hit Buffer* state machine actions is controlled by the signal Test MODe (TMOD, see Section 13).

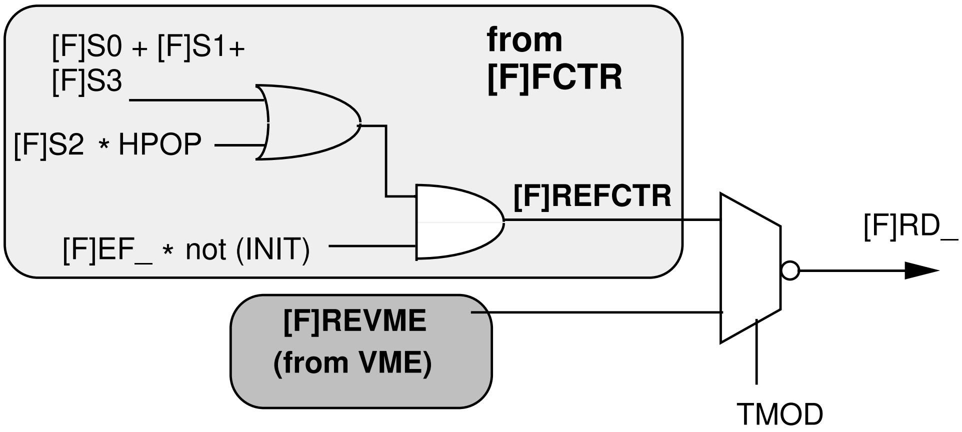

Figure 6: Fifo read enable generation. The symbol [F] should be replaced by H or R to obtain the two drawings relative to the Hit or Road fifo.

The main control functions of the Auxiliary Logic are:

1. Control of the pipeline stages, generating clock enables to Middle Latch registers (LEDATA and LEADD as shown in Figures 3 and 4) and read enables to fifos (HRD_ and RRD_). In Figure 6 it is shown how the fifo output control is implemented. The symbol [F] should be replaced by H or R to obtain the two drawings relative to the Hit or Road fifo. The signals HREVME and RREVME (represented by [F]REVME in Figure 6) allow a fifo to be read by VME. The VME action (enabled by TMOD=1) is multiplexed to the action of the standard "Fifo Controller" (enabled by TMOD=0), described in the document SVT note 34. The synchronous [F]fifo push out a word on the next clock rising edge, if the [F]RD_ is active (low).

2. Control of buses, generating the output enables of the three state buffers. Three buses are in the *Hit Buffer* : the fifo's output bus (HRFQ[0-22] in Figure 2), the map's output bus (LKO[0-15] in Figure 2), and the data bus going to the output register (DATA[0-20] in Figure 2). In all these cases the

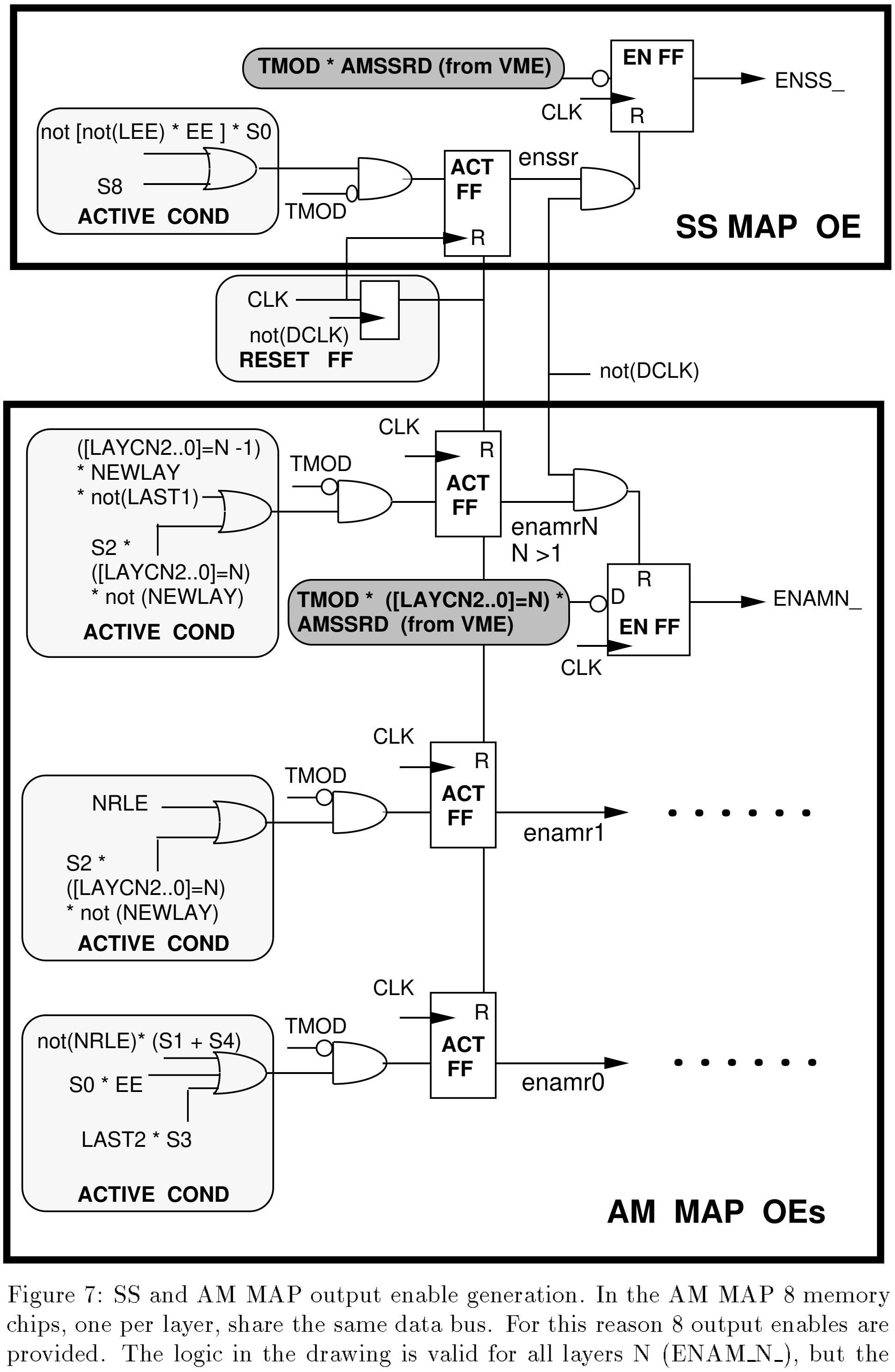

Figure 7: SS and AM MAP output enable generation. In the AM MAP 8 memory chips, one per layer, share the same data bus. For this reason 8 output enables are provided. The logic in the drawing is valid for all layers N (ENAM_N_), but the reset signals enamr1 and enamr0 for the layers 1 and 0 are drawn independently because different.

generated output enables are shaped to guarantee no conflict between the different objects that can drive the bus. When the driving of a bus must be changed inside a clock cycle, the control logic is built so that the "active enable" is disabled before the "inactive enable" is activated. This function is implemented using Flip-Flops to generate the enables. The D inputs are used to synchronously disable the enables (5 nsec of delay from the clock rising edge), while asynchronous resets, whose timing is controlled by a double frequency clock, are used to activate the enables with a delay that is roughly a quarter of clock cycle.

Figure 7 shows the most difficult case, the control logic for the LKO bus. Eight enables to the AM MAP (ENAM_[0-7]) and the enable to the SS MAP (ENSS_) are generated in the EN FFs, multiplexing the VME action with the FSM action. If FSM has the control of the bus (TMOD=0), each enable is disabled at each clock rising edge (5 nsec of delay). If an ACTIVE CONDition shown in the figure is TRUE the corresponding ACTive FF will be able to activate the output enable trough the asynchronous reset whose timing is controlled by the double frequency clock DCLK. The RESET FF turn off all ACTive FFs at each clock cycle, so that the asynchronous reset generated by them is short and cannot override the synchronous set of EN FFs at the next clock rising edge.

Figure 8 shows the control logic for the DATA bus. The same strategy is used since the EN FFs are synchronously disabled and asynchronously activated by the ACT FFs whose action is delayed a quarter of clock cycle with respect to the synchronous action.

Figure 9 shows the control logic for the HRFQ bus. The logic is simpler because the Hit and Road fifos are active in well separated phases of the event processing (during WRITE MODE is active the Hit-FIFO and during READ MODE the Road-FIFO). However, in the last clock cycle of the FSM state 0 (the end of S0 is the end of the WRITE MODE) the bus must be released by the Hit-FIFO and taken by the Road-FIFO. It is necessary to disable quickly WR_ to allows RD_ to be activated in the same clock cycle.

3. Control of the memories, generating the chip select signals. Chip selects are not continuously activated to reduce power dissipation on the board. Figure 10 shows this control logic.

4. Control of output data flow. Figure 11 shows the generation of the output Data Strobe (DS_ is active low to be used directly as it is in the downstream Fifo), the output data bus (OUT[0-22]) and the output Spy Buffer bus (OSPY[0-22]) used by the Spy Buffer system, devoted to data flow monitoring (see Section 11).

7.1 FSM Auxiliary Logic inputs

Here we list the inputs that are not described in the Section 6.

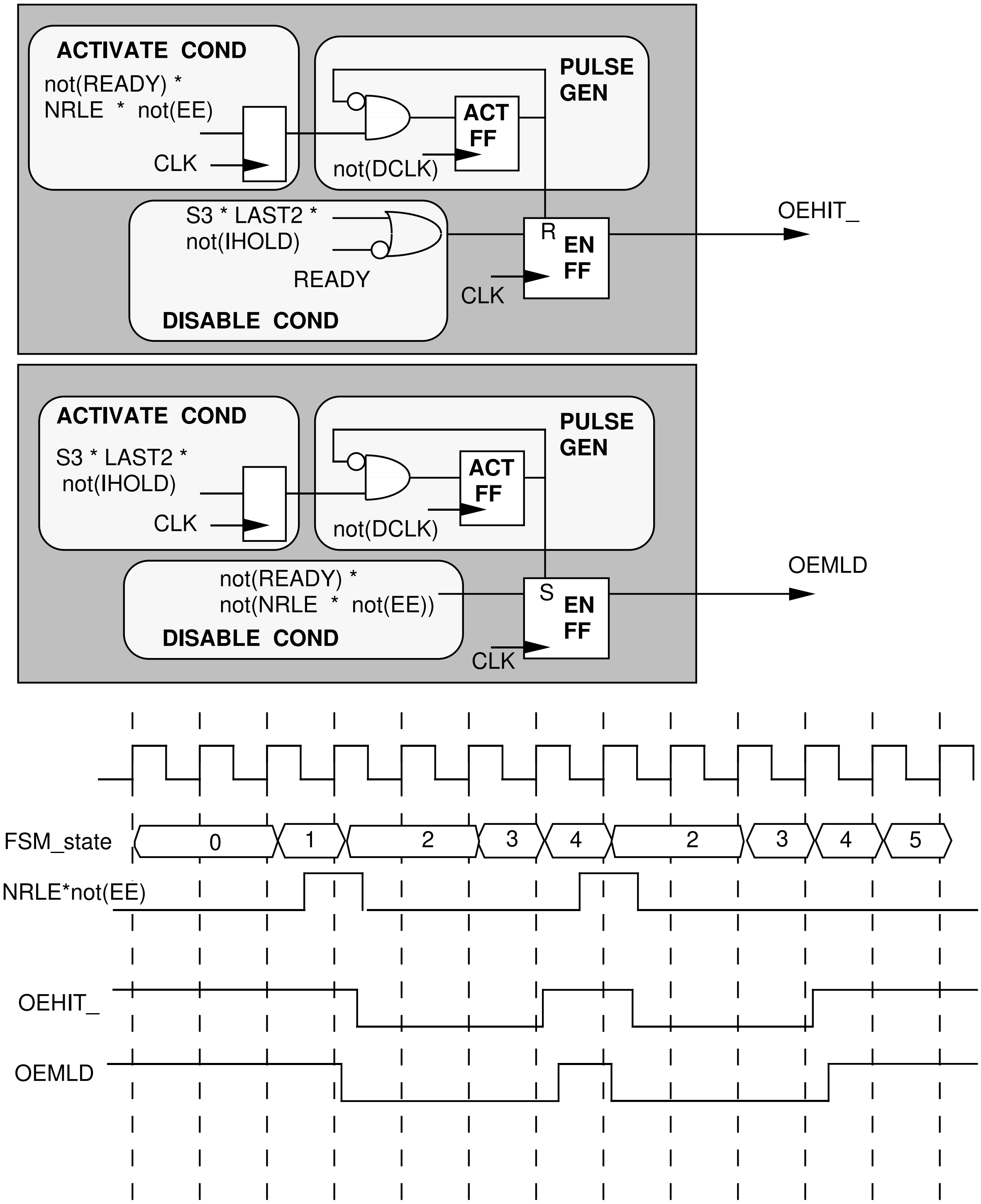

Figure 8: Hit List Memory and Middle Latch DATA output enable generation.

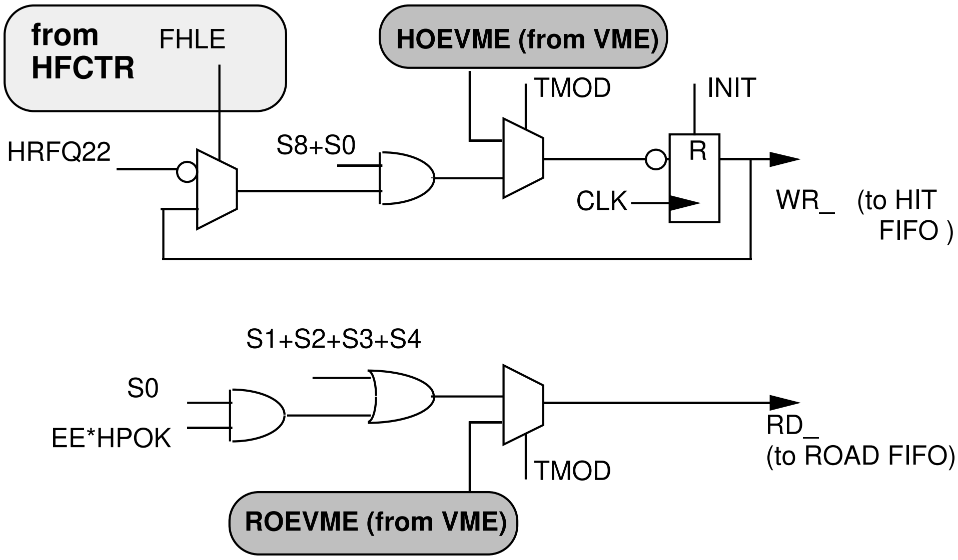

Figure 9: Fifo's output enable generation.

AMSSRD Source: VME interface. Explanation: "AM SS map ReaD" activates the output enable to the AM or SS MAP from the VME interface (see Figure 7).

AMVME_ Source: VME interface. Explanation: "AM map select from VME" is the chip select enable to read or write the memory from the VME interface (see Figure 10).

FHLE Source: Hit Fifo Controller. Explanation: "Fifo Hit Latch Enable" is the clock enable to the Fifo Register from the Hit Fifo ConTRoller. It is active when a new word available from the Hit Fifo should be latched in the Fifo Register (see Figure 9).

HREFCTR Source: Hit Fifo Controller. Explanation: "Hit Read Enable from Fifo Controller" is the read enable (see Figure 6) to the Hit Fifo from the internal state machine HFCTR, the Hit Fifo ConTRoller.

HRFQ22 Source: Hit-FIFO, Road-FIFO. Explanation: "Hit Road Fifo Q output bit 22" is the end event bit in the HRFQ bus (see Figure 9).

HREVME Source: VME interface. Explanation: "Hit Read Enable from VME" is the read enable to the Hit Fifo from the VME interface (see Figure 6).

HOEVME Source: VME interface. Explanation: "Hit fifo Output Enable" is the output enable to the Hit Fifo from the VME interface (see Figure 9).

LAYCN[0-2] Source: COUNTER1. Explanation: "LAYer CouNter" is the output of the counter COUNTER1 in Figure 3 and keeps track of the number of the layer currently processed.

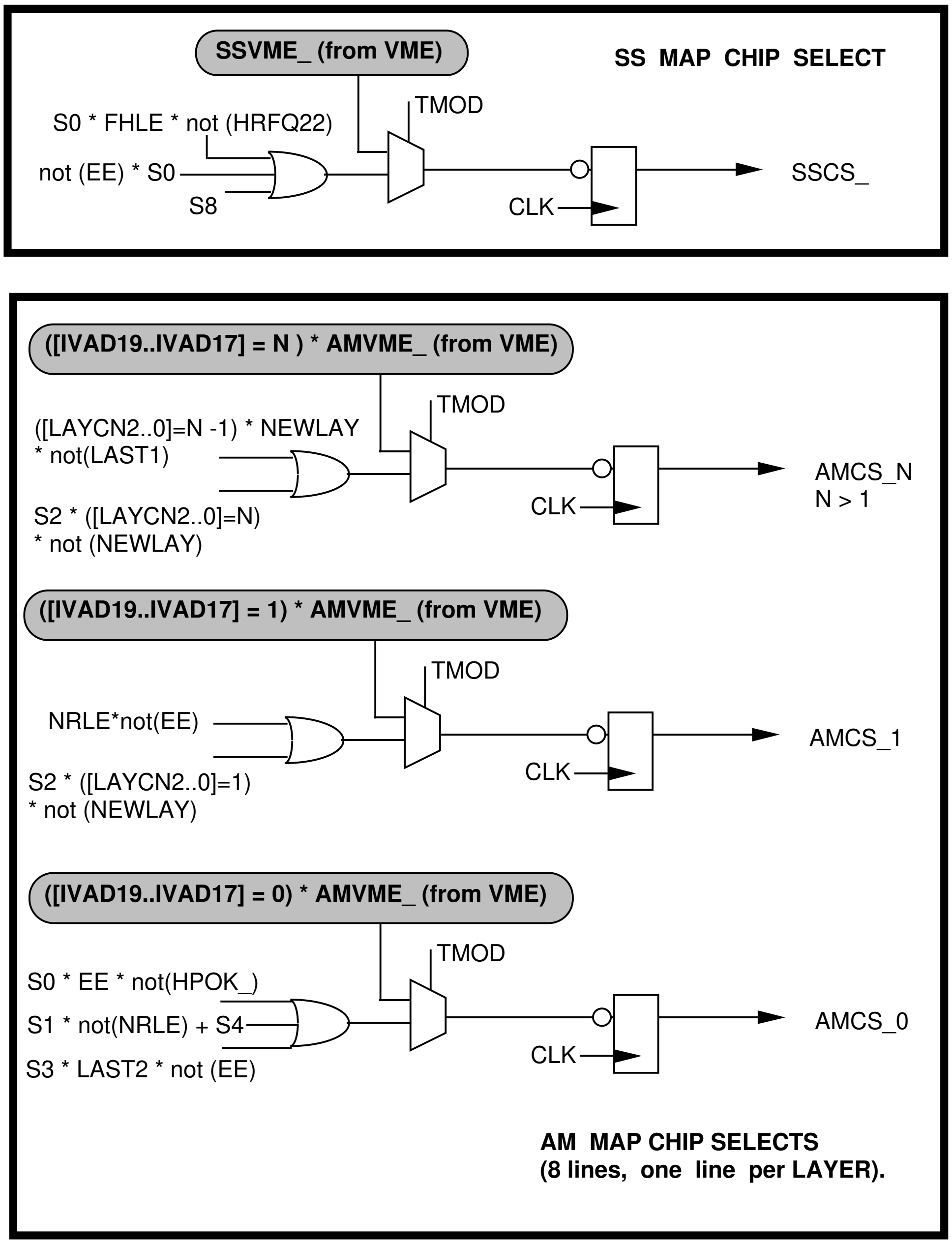

Figure 10: SS and AM MAP chip selects generation.

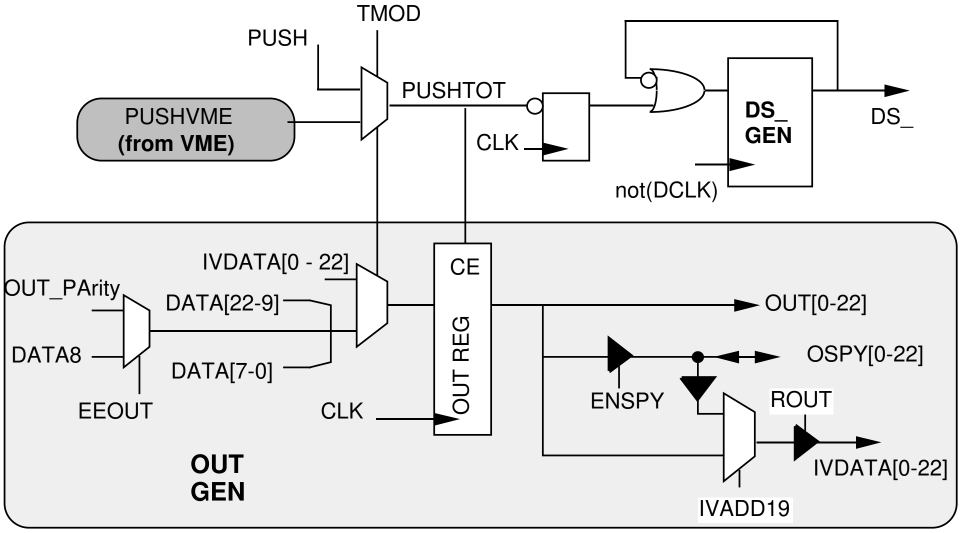

Figure 11: Output data and Data Strobe (DS_) generation.

PUSHVME Source: VME INTERFACE. Explanation: "PUSH from VME" allows to write a word in the output register from VME (see Section 14.1) and to generate in the following cycle the Data Strobe (DS_) sending one word on the output stream (see Figure 11).

RREFCTR Source: Road Fifo Controller. Explanation: "Road Read Enable from Fifo Controller" is the read enable (see Figure 6) to the Road Fifo from the internal state machine RFCTR, the Road Fifo ConTRoller.

RREVME Source: VME interface. Explanation: "Road Read Enable from VME" is the read enable to the Road Fifo from the VME interface (see Figure 6).

ROEVME Source: VME interface. Explanation: "Road fifo Output Enable" is the output enable to the Road Fifo from the VME interface (see Figure 9).

SSVME_ Source: VME interface. Explanation: "SS map select from VME" is the chip select enable to read or write the memory from the VME interface (see Figure 10).

TMOD Source: VME interface. Explanation: "Test MODe" puts the *Hit Buffer* in Test Mode. When the *Hit Buffer* is in Test Mode (as long as TMOD is TRUE) it is possible to access internal data from VME (see Section 13).

7.2 FSM Auxiliary Logic outputs

AMCS_[0-7] Destination: AM Map. Explanation: RAM chip selects. Since 8 memory chips, one per layer, share the same data bus, 8 chip selects are provided.

DS_ Destination: Line Driver of Output Connector. Explanation: Data Strobe signal is sent in output to the downstream board where is used to write data in the input fifo.

ENAM_[0-7] Destination: AM Map. Explanation: RAM chip output enables. Since 8 memory chips, one per layer, share the same data bus, 8 output enables are provided. These signals are shaped to guarantee no conflict on the LKO bus.

ENSS_ Destination: SS Map. Explanation: RAM chip output enable. Since the SS Map share the data bus with the AM Map, ENSS_ is shaped to guarantee no conflict on the LKO bus.

HRD_ Destination: Hit-FIFO. Explanation: read enable for the Hit-FIFO. If HRD_ is active the Hit-FIFO push out a word at the next clock rising edge.

LEADD Destination: ML ADD. Explanation: "Latch Enable ADDress" causes the present output of the look-up table (SS Map while in Write mode, AM Map while in Read mode) to be latched at the next clock rising edge. LEADD is simply defined as: LEADD=RLE+HLE*LEP (see Figures 3 and 4)

LEDATA Destination: ML DATA. Explanation: "Latch Enable DATA" causes the present output of the FIFO REGISTER to be latched at the next clock rising edge. LEDATA is simply defined as: LEDATA=NRLE+HLE (see Figures 3 and 4).

OEHIT_ Destination: Hit List Memory. Explanation: enables the Hit List Memory output. This signal is shaped to guarantee no conflict between the Hit List Memory (OEHIT_ is the enable) and ML DATA (OEMLD is the enable) on the DATA bus.

OEMLD Destination: ML DATA. Explanation: enables the ML DATA output. This signal is shaped to guarantee no conflict between the Hit List Memory and ML DATA on the DATA bus.

RD_ Destination: Road-FIFO. Explanation: enables the Road-FIFO output. This signal is shaped to guarantee no conflict of the Hit and Road-FIFO on the HRFQ[0-22] bus.

RESTAG_ Destination: Hit Count. Explanation: RESTAG_ is the Hit Count reset. It is simply defined as RESTAG_=NOT(INIT+CLRTAG).

RRD_ Destination: Road-FIFO. Explanation: read enable for the Road-FIFO. If RRD_ is active the Road-FIFO push out a word at the next clock rising edge.

SSCS_ Destination: SS Map. Explanation: RAM chip select. Since the SS Map share the data bus with the AM Map, one chip select is provided.

WR_ Destination: Hit-FIFO, FIFO REGISTER, SPY BUFFER CONTROL (see Section 11). Explanation: enables the Hit-FIFO output, identifying the right time *hits* flow to the FIFO REGISTER and to the Hit Spy Buffer. This signal is shaped to guarantee no conflict of the Hit and Road-FIFO on the HRFQ[0-22] bus.

8 Hit List Control

The *Hit Buffer* has the capability of storing more then one *hit* for each SuperStrip — one Hit List per SuperStrip. In addition, the maximum number of words in a Hit List — the reserved size — is allowed to vary as a function of the SuperStrip, up to a maximum of 8. Additional words above the maximum allowed for a SuperStrip are discarded: the Hit List Memory content is never overwritten during an event. This overflow situation is signaled by asserting an error bit. The size reserved for each SuperStrip may be 2, 4, or 8. Since the Hit List Memory is 64k words deep, the maximum number of Hit Lists that the *Hit Buffer* will handle is between 2^{13} and 2^{15}.

The size reserved for a SuperStrip is encoded in the 2 LSb of the SuperStrip ID. For the current SuperStrip, these are the 2 LSb of the 16-bit output of ML ADD — SS[0-1] in Figure 2. The code is shown in Table 3.

Size	SS1	SS0
2	x	0
4	0	1
8	1	1

Table 3: SS code for the maximum number of words in a Hit List. The size reserved in the Hit List Memory is encoded in the SuperStrip ID.

With the appropriate timing, the Hit List Control generates the 16-bit address of each word in the Hit List Memory — ADD[0-15] in Figure 2. The 13 MSb of the address (ADD[3-15]) are obtained from the 13 MSb of the SuperStrip ID coming from ML ADD (SS[3-15]). The 3 LSb of the address (ADD[0-2]) are obtained by combining the 3 LSb of the SuperStrip ID (SS[0-2]) and the information stored in the Hit Count (while in Write Mode) or in COUNTER2 (while in Read Mode). Details of the Hit List Memory addressing will be given in the following subsections.

8.1 Hit Count

The Hit Count keeps track of the number of words currently stored in each Hit List acting as a set of counters. The Hit Count is implemented as a TAG RAM using chips IDT71B74. This allows a fast reset of all counters at the end of each event. A simplified block diagram of such a chip is shown in Figure 12. The IDT71B74S10 is a high-speed address comparator consisting of a static RAM organized as 8k x 8 bit and an 8-bit comparator. It continuously compares the input data to the content pointed by the address lines and sets the Match line to 1 or 0 according

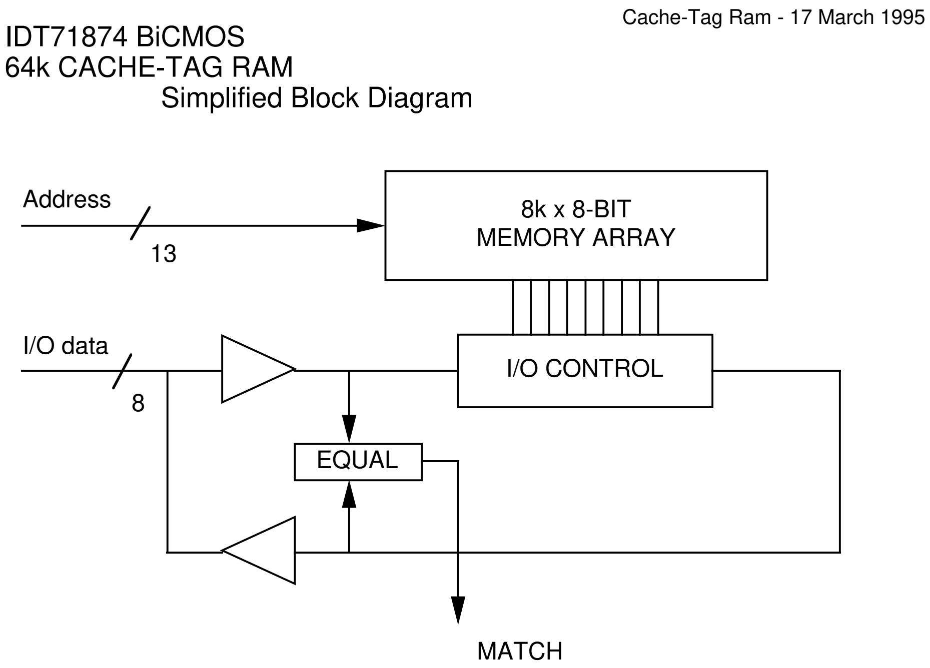

Figure 12: Simplified block diagram of a TAG RAM chip

to successful/unsuccessful comparison. The IDT71B74 also provides a single RAM clear control, which clears all words in the internal RAM to zero when activated.

In the Hit Count, 8 chips are connected as shown in Figure 13. Note that the chips are used as if they were 8k x 1 bit. The CLRTAG signal resets all TAG RAM content to zero. The CLRTAG signal will have to be asserted for two consecutive clock cycles, to make it long enough to satisfy chip specifications.

The Hit Count keeps track of the number of words currently stored in each Hit List acting as a set of unary counters, the counter bits being the output of the Match lines. According to the SuperStrip ID least significant bits (SS[0-2]) the Hit Count can be seen as a single bank of 8k 8-bit unary counters, 2 banks of 8k 4-bit unary counters, or 4 banks of 8k 2-bit unary counters. Table 4 shows (with a # symbol) which of the 8 Match lines must be combined according to the possible values of SS[0-2] to yield the unary counter bits. All of these counters are cleared simultaneously at the beginning of each event by the CLRTAG signal. The current number of words stored in the Hit List is obtained by converting from unary to binary the M lines flagged with #. In Write Mode, the appropriate counter is increased every time a word is stored into the Hit List Memory. This is done by the Hit List Control that activates the proper Write line (W_[0-7]) to the Hit Count, as indicated in Table 5.

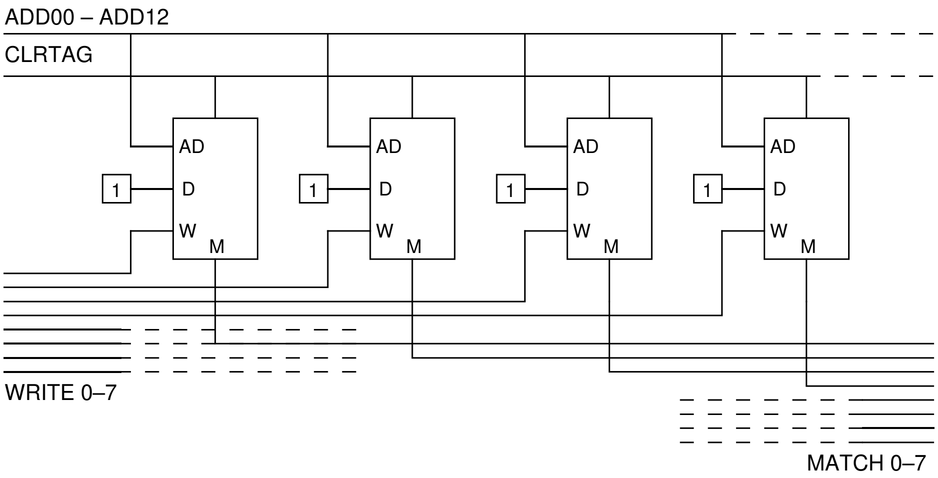

Figure 13: Hit Count

SS2	SS1	SS0	M0	M1	M2	M3	M4	M5	M6	M7
0	0	0	#	#						
0	1	0			#	#				
1	0	0					#	#		
1	1	0							#	#
0	0	1	#	#	#	#				
1	0	1					#	#	#	#
x	1	1	#	#	#	#	#	#	#	#

Table 4: Match lines to unary counter correspondence

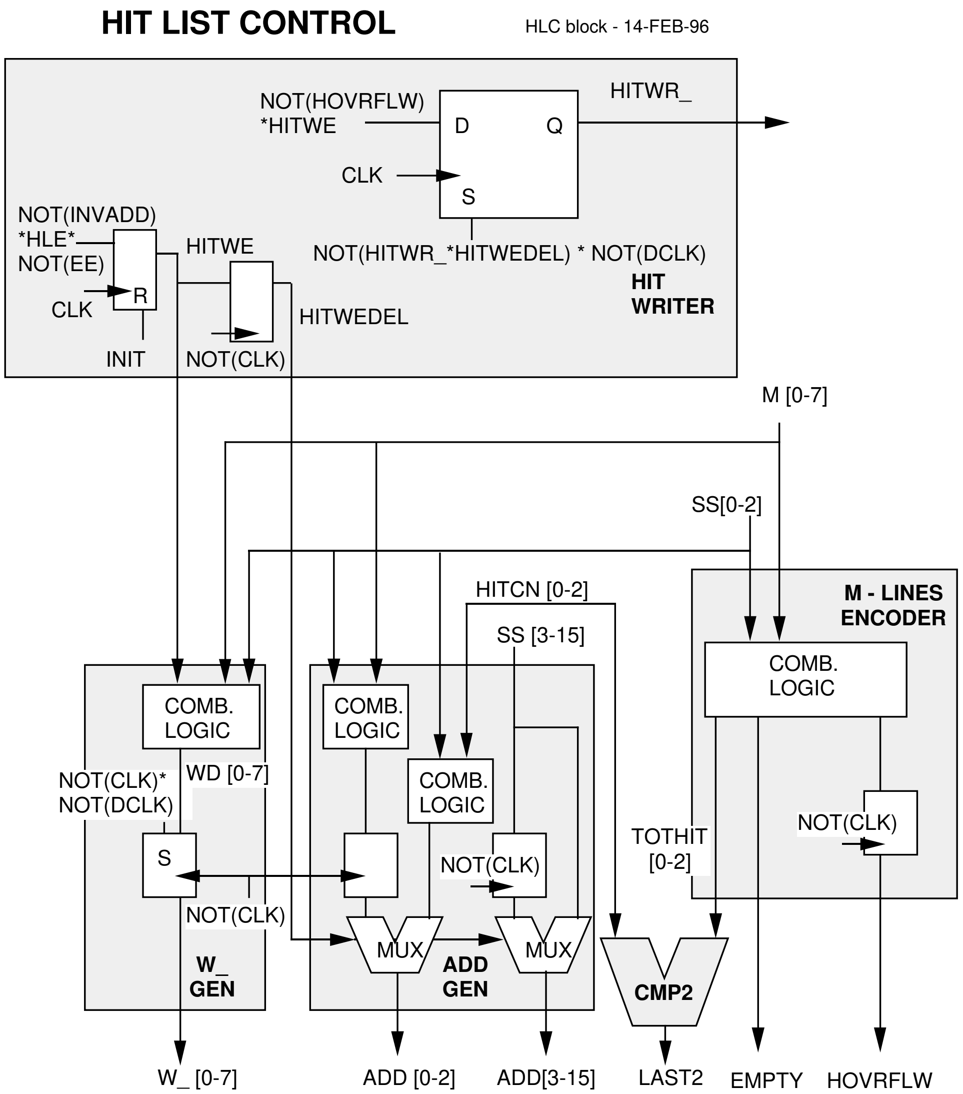

Figure 14: Hit List Control block diagram

SS2	SS1	SS0	W0	W1	W2	W3	W4	W5	W6	W7
0	0	0	1	M0						
0	1	0			1	M2				
1	0	0					1	M4		
1	1	0							1	M6
0	0	1	1	M0	M1	M2				
1	0	1					1	M4	M5	M6
x	1	1	1	M0	M1	M2	M3	M4	M5	M6

Table 5: Write lines assertion rule. Each W_ line is asserted (i.e. given a low pulse) when a 1 appears in the corresponding row-column. Mn indicates that W_ is asserted if Mn is 1.

8.2 Hit List Control block diagram

Figure 14 shows a block diagram of the Hit List Control that allows us to summarize its functions:

1. The block ADD GEN generates the address of each word in the Hit List Memory (ADD[0-15]).
2. The block W_ GEN generates the write lines to the Hit Count (W_[0-7).
3. The block M-LINES ENCODER generates the binary code of the current Hit List length using the M lines from the Hit Count (TOTHIT[0-2]). It also generates signals for flagging empty Hit Lists (EMPTY) and Hit List overflow (HOVRFLW).
4. The comparator CMP2 signals the last Hit List word (LAST2), when HITCN[0-2] from COUNTER2 match TOTHIT[0-2].
5. The block HIT WRITER generates the control signals necessary for writing a word in the Hit List Memory (HITWEDEL and HITWR_).

8.3 Hit List Control timing in the Read and Write Modes

Figure 15 shows the timing diagram of the Hit List Control signals. During the Write Mode, the functions of the Hit List Control are doubled with respect to those in Read Mode. While in Write Mode, in a single clock cycle a read-modify-write operation is performed by the Hit List Control on the Hit Count. For this reason the timing is more complicated in the Write Mode.

Write Mode

1. During the first half of the clock cycle, the Hit Count TAG RAM chips are read at the address SS[3-15] to generate the proper values of the Match lines

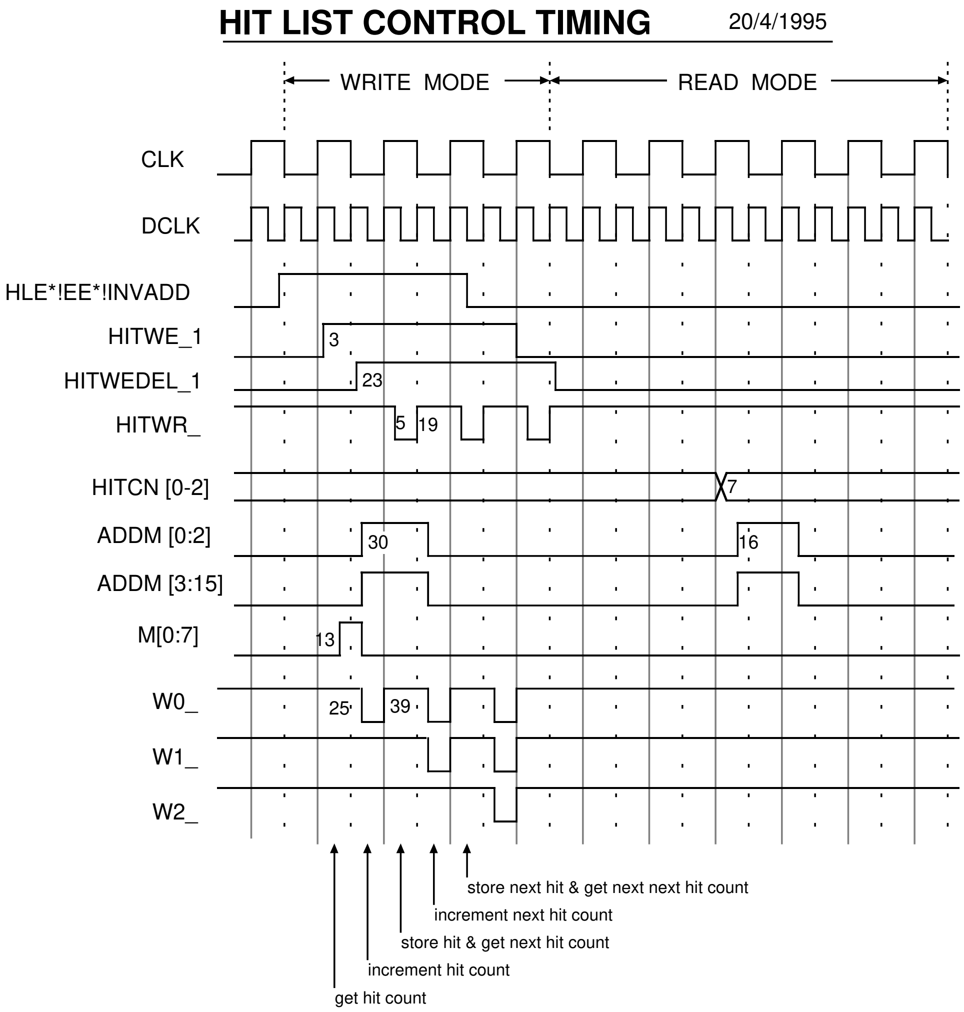

Figure 15: Hit List Control timing. The numbers by signal edges indicate simulated delay in nsec from clock rising edge, using the Xilinx EPLD X7336-5.

M[0-7]. These Match lines combine with SS[0-2] to generate the proper WD[0-7] and ADD[0-2] signals (blocks W_ GEN and ADD GEN in Figure 14), so that the proper counter in the Hit Count can be incremented in the second half of the clock cycle and the *hit* word may be stored in the Hit List Memory in the next clock cycle. During the first half of the clock cycle the previous *hit* word, if any, is stored in the Hit List Memory. The HOVRFLW signal is also generated in the first half of the clock cycle, according to Table 6. When

SS2	SS1	SS0	HOVRFLW
0	0	0	M1
0	1	0	M3
1	0	0	M5
1	1	0	M7
0	0	1	M3
1	0	1	M7
x	1	1	M7

Table 6: HitOVeRFLoW is set by one of the odd Match lines if activated when the relative SS[0-2] code is also selected. On the same raw it is shown the Match line and the corresponding code

HOVRFLW is asserted, the current word will not be stored into the Hit List Memory.

2. During the second half of the clock cycle, the W_[0-7] lines are used to increment by 1 the unary counter by properly writing into the TAG RAM chips (see Table 5). This write operation alters the status of the Match lines M[0-7].

3. The output signals W_ [0-7] and ADD[0-2] (see Figure 14), needed in the second half of the clock cycle, are functions of the Match lines. In order to keep them stable while the TAG RAM is being written, they are generated during the first half of the clock cycle and then saved in a flip-flop register latched by the clock trailing edge (see the blocks W_ GEN and ADD GEN in Figure 14). This flip-flop also guarantees a stable address in the first half of the next clock cycle, when the word is written in the Hit List Memory. For the same reason, also the lines ADD[3-15] are saved in flip-flops (see Figure 14) latched by the clock trailing edge.

Read Mode

During the Read Mode, the Hit List Control gets the total number of *hits* to be pushed on output, by decoding the unary counter (Match lines M[0-7]) into a binary code (TOTHIT[0-2]). Under control of SS[0-2], the M-LINES ENCODER (see Figure 14) picks the relevant Match lines, indicated in Table 4, from the Hit Count and generates TOTHIT[0-2], according to Table 7. The Match lines are valid soon after the ML ADD output is presented to the TAG RAM.

The Hit List Control continuously compares the binary code TOTHIT[0-2] to the COUNTER2 output (HITCN[0-2]) and asserts LAST2 when they match.

Unary counter			EMPTY	TOTHIT[0-2]
2-bit	4-bit	8-bit		
00	0000	00000000	1	0
01	0001	00000001	0	0
11	0011	00000011	0	1
	0111	00000111	0	2
	1111	00001111	0	3
		00011111	0	4
		00111111	0	5
		01111111	0	6
		11111111	0	7

Table 7: Unary counter to binary code conversion. The number of bits of the unary counter is defined in Table 4.

TOTHIT[0-2] zero code corresponds to both cases of one or no words in a superstrip. EMPTY is asserted to distinguish the second case, when all the relevant Match lines are zero (see Table 7).

8.4 More on Hit addressing

The address of the first *hit* word for each SuperStrip — i.e. the Hit List base address — is obtained from the word SS[15-0] coming out of ML ADD, as indicated in the 2nd column of Table 8.

SS ID	Hit List		Hit List Memory address			
SS[2-0]	Base	Size	ADD [15-3]	ADD[2]	ADD[1]	ADD[0]
x x 0	SS[15-1]	2	SS[15-3]	SS[2]	SS[1]	Count[0]
x 0 1	SS[15-2]	4	SS[15-3]	SS[2]	Count[1]	Count[0]
0 1 1	SS[15-3]	8	SS[15-3]	Count[2]	Count[1]	Count[0]
1 1 1	unused					

Table 8: Addressing the Hit List Memory. The Hit List base address and reserved size are encoded in the SuperStrip ID SS[0-15]. The address ADD[0-15], to a given cell in the Hit List Memory, is a combination of the base address and a word count COUNT[0-2].

To address a word in the Hit List Memory, the Hit List Control propagates SS[15-3] to ADD[15-3], with the appropriate timing. According to Table 8, the Hit List Control also propagates the proper part of the SS[2-0] bits to the ADD[2-0] lines and complements it on the right with the LSb of a word counter, as indicated in the 3 leftmost columns of the same table. The word counter field, Count[2-0], comes:

- during the Write mode, from the binary coded value of the Hit Count unary counter (cf. Table 4)
- during the Read mode, from the output of COUNTER2 (HITCN[2-0]).

8.5 Details of the block diagram

Here we summarize all parts and signals of the Hit List Control block diagram in Figure 14.

Signal Names

ADD[0-15] Source: ADD GEN. Destination: Hit List Memory. Explanation: "ADDress" bus for the Hit List Memory.

EMPTY Source: M-LINES ENCODER. Destination: FSM. Explanation: This signal means that the SuperStrip currently present at the output of ML ADD does not have any *hit*.

HITCN[0-2] Source: COUNTER2. Destination: ADD GEN, CMP2. Explanation: "HIT CounT" is the number of words read out from a Hit List, during the Read mode.

HITWE Source: HIT WRITER. Destination: W_ GEN. Explanation: Hit Write Enable triggers the increment of the appropriate unary counter in the Hit Count (that is the generation of the appropriate W_ lines in W GEN) and the writing of a word in the Hit List Memory at the next clock cycle (that is the generation of HITWR_ in HIT WRITER).

HITWEDEL Source: HIT WRITER. Destination: EE REGISTER, ADD GEN. Explanation: "HIT Write Enable DELay" is HITWE delayed by half a clock cycle to ensure the ADD bus is stable during actual *hit* writing. Infect HITWEDEL controls the MUXES in ADD GEN of Figure 14. Inside the EE REGISTER is used in the generation of the Internal Overflow error (see section 10.1).

HITWR_ Source: HIT WRITER. Destination: Hit List Memory. Explanation: "HIT WRite" is the write pulse to the Hit List Memory.

HLE See section 6.2. In the Hit List Control HLE is used by the HIT WRITER to enable a writing cycle into the Hit List Memory.

HOVRFLW Source: M-LINES ENCODER. Destination: EE REGISTER, VME error register REG0, HIT WRITER. Explanation:"Hit OVeRFLoW" signals that the maximum number of *hits* allowed in the SuperStrip where the current hit should be written has been reached. It will set a flag in both destination registers and will disable the writing of the present *hit* in the Hit List Memory.

LAST2 See section 6.1.

M[0-7] Source: Hit Count. Destination: ADD GEN, M-LINES ENCODER, W GEN. Explanation: "Match" flags from the 8 TAG RAM chips making the unary counter for the current Hit List.

SS[0-15] Source: ML ADD. Destination: ADD GEN, M-LINES ENCODER, W GEN. Explanation: Current "SuperStrip" ID.

TOTHIT[0-2] Source: M-LINES ENCODER. Destination: CMP2 (see Figure 16) Explanation: Binary conversion of the unary counter, indicates the total number of *hits* already stored for the current SuperStrip, but starting from 0. In this way it can be used to compare with the output of COUNTER2, which also starts from 0. Its value is thus zero both for one *hit* and for no *hits* (this is the reason for a separate EMPTY signal).

W_[0-7] Source: W GEN. Destination: Hit Count. Explanation: "Write" signals to the Hit Count TAG RAM.

Block Names

ADD GEN outputs the ADD[0-2] address as indicated in Table 8. and ADD[3-15] as shown in Figure 16.

CMP2 is a comparator. It signals the last Hit List word (LAST2), when HITCN[0-2] from COUNTER2 match TOTHIT[0-2].

HIT WRITER controls writing into the Hit List Memory. The HLM WRITER generates the proper write pulse (HITWR_) as needed by the HLM in the second half of the clock cycle following the assertion of HLE, provided that HOVRFLW is not asserted. It also generates the Output Enable for ML DATA for the corresponding clock cycle (HITWE).

M-LINES ENCODER this is a block of combinatorial logic. It converts the unary counter (M lines) as indicated in Table 4 to the binary value TOTHIT[0-2]. It also generates the EMPTY signal (EMPTY is asserted when all M lines marked with # in the proper rows-columns of Table 4 are zero). Finally, in case all the relevant M lines are already 1, it generates the HOVRFLW signal (Table 4).

W_ GEN this is a block of combinatorial logic followed by a register that produces low active signals (W_ lines) as needed to increment the unary counter according to the rules of Table 5. The rising edge of the W_ pulses is produced before the next clock rising edge using the double frequency clock.

9 Hit Buffer Block Diagram

——FIGURE 16 HAS BEEN UPDATED BUT THE TEXT OF THIS SECTION IS OLD AND NEEDS TO BE UPDATED. PLEASE SKIP THE PARAGRAPH, AND USE THE FIGURE FOR THE OTHER SECTIONS.——-

Figure 16 shows the overall block diagram of the *Hit Buffer* data-flow and control logic. It shows all parts and data paths mentioned above. Control signals to/from the FSM and VME boxes are not indicated. Most of the control signals shown as arrows with one free end, entering or exiting from the various components, connect to either FSM or VME.

Now we describe and explain the block diagram, by giving details of all parts in Figure 16. All the control logic and all latches and registers have been implemented into XILINX XEPLD chips (parts implemented into XEPLD's in Figure 16 are enclosed in bordered boxes and parts which *can not* be integrated into XEPLD are indicated with gray ovals).

Signal Names

AMWR Source: VME base address decoding. Destination: AM Map. Explanation: "AM Map WRite" signals that VME wants to write the AM Map RAM.

CLK Source: Master Clock. Destination: all synchronous logic blocks. Explanation: "CLocK" is the main system clock, it is a symmetric square wave that should cycle at about 30 MHz.

CTRLINES Source: VME INTerface, FSM, FIFO REGISTER. Destination: EE REGISTER, VME error register REG0. Explanation: this name indicates a set of control signals used to generate the parity and error flags to be put in the *End Event* word and in the VME error register.

DCLK Source: Master Clock. Destination: ??? Explanation: "Double frequency CLocK" is a clock with a frequency which is twice that of the main system clock CLK. It is used when multiple operations are required in a single main clock cycle (e.g. read-modify-write of the Hit List Control).

DS Source: DS_ GEN. Destination: Output Line Drivers. Explanation: "Data Strobe" signal for the output data stream.

EE Source: FIFO REGISTER. Destination: FSM. Explanation: "End Event" signal from input streams, it corresponds to MO[22]

EVHOVRFLW Source: M-LINES ENCODER in Hit List Control. Destination: EE REGISTER, VME error register REG0. Explanation:"EVent Hit OVeR-FLoW" signals that the maximum number of *hits* allowed in a SuperStrip has been reached (see Section 8) in the current event. It will set a flag in both destination registers.

HI[0-22] Source: HL transparent latch. Destination: MX1 & SWAP, HI to VME buffers. Explanation: The "Hit Input" bus contains the last word popped out from the input Hit-FIFO and frozen on HL by the Hit-FIFO controller HFCTR.

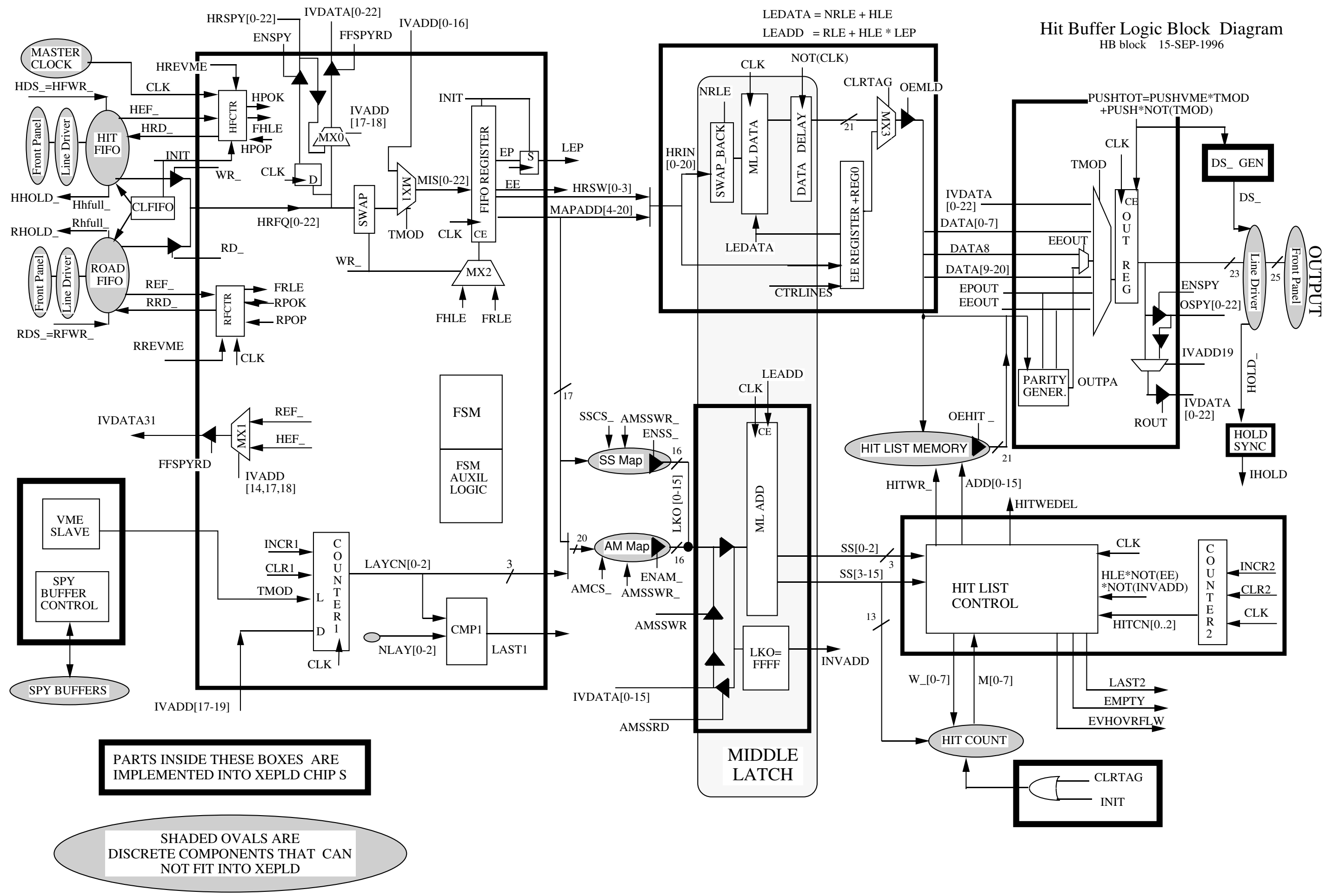

Figure 16: *Hit Buffer* logic block diagram

HITWEDEL Source: Hit Writer in Hit List Control. Destination: Middle Latch, Hit List Memory. Explanation: "HIT Write Enable DELay" enables writing of incoming *hit* words into the Hit List Memory by enabling the output from ML DATA and simultaneously disabling the Hit List Memory output.

HITWR_ Source: Hit Writer in Hit List Control. Destination: Hit List Memory. Explanation: "HIT WRite" is the write pulse to the Hit List Memory.

HRIN[0-20] Source: SWAP. Destination: ML DATA, *End Event* Register. Explanation: "Hit/Road IN" corresponds to HI[0-20] (Write Mode) or RI[0-20] (Read Mode). Bit 21, i.e. the End Packet bit from the input streams is only used by FSM and it is not propagated to Middle Latch.

HRSW[0-3 Source: FIFO REGISTER. Destination: SWAP. Explanation: "Hit/Road SWapped bus" contains the 4 bits of the FIFO data that need to be propagated to the output, but that are not used to address the Hit List memory. They correspond to HI[0-3] during the Write Mode and to RI[17-20] during the Read Mode.

HVME Source: VME interface. Destination: HI to VME buffers. Explanation: "Hit stream to VME bus" enables output from the *hit* transparent Latch (HL) to the VME data bus.

HREVME Source: VME interface. Destination: Hit-FIFO Controller HFCTR. Explanation: "Hit fifo Read Enable from VME" generates a HR pulse to read one word from the Hit-FIFO during the next clock cycle. The VME interface has a simple FIFO controller that generates HREVME. HREVME is logically ORed with the action of the Hit-FIFO Controller HFCTR to generate the HR pulse to the Hit-FIFO (see Section ??? and Figure ???). The result is that the HPOP-HPOK protocol is bypassed when the input FIFO is read by VME. In this case the FIFO is read independently of the empty flag status (HFE), which is also read by VME.

IHOLD Source: HOLD SYNC. Destination: FSM. Explanation: "Internal HOLD" is the synchronized line corresponding to the external asynchronous HOLD_, signaling an Almost Full from downstream FIFO.

INIT Source: VME interface. Destination: FSM, HFCTR, RFCTR, TAG RAM, FIFO REGISTER, EE REGISTER. Explanation: "INITialize" initiates an Init cycle. As long as INIT is high the FSM's will stay in the "ready" state. When INIT is dropped the *Hit Buffer* starts performing the normal operation flow.

IVADD Source: VME interface. Destination: MAPADD multiplexer, LAY multiplexer. Explanation: "Internal Vme ADDress" is the internal address bus from VME.

IVDATA Source: VME interface, HI to VME buffers, RI to VME buffers, LKO to VME buffers, OUT REGISTER multiplexer, OUT REGISTER to VME buffers. Destination: VME interface, SS Map, AM Map. Explanation: "Internal VME DATA" is the internal data bus to/from VME.

LEADD Source: AUXILIARY LOGIC Destination: ML ADD. Explanation: "Latch Enable ADDress" causes the present output of the look-up table (SS Map while in Write mode, AM Map while in Read mode) to be latched at the next clock rising edge.

LEDATA Source: AUXILIARY LOGIC Destination: ML DATA. Explanation: "Latch Enable DATA" causes the present output of the FIFO REGISTER to be latched at the next clock rising edge.

M[0-7] Source: Hit Count. Destination: Hit List Control. Explanation: "Match" flags from the 8 TAG RAM chips making the unary counter for the current Hit List.

MIS[0-22] Source: MX1 & SWAP. Destination: FIFO REGISTER. Explanation: "Multiplexed Input Streams" bus. In Write Mode, it corresponds to the *hit* input stream. In Read Mode, it corresponds to the *road* input stream. On the MIS lines, the *road* number and the ϕ sector are coded in bit fields which are different from those of the *road* input stream (Section 3.1). In Read Mode the *road* number is coded by MIS[4-20] (MIS[4-20] = RI[0-16]), while the ϕ sector is coded by MIS[0-3] (MIS[0-3] = RI[17-20]). This field swap allows a single bus to address both the AM Map and the SS Map, compatible with a simple mapping into consecutive VME addresses.

NLAY[0-2] Source: Dip-switch. Destination: Layers comparator CMP1 Explanation: "Number of LAYers" codes for the total number of layers.

OEHIT Source: ???. Destination: Hit List Memory. Explanation: "Output Enable HIT" enables output from Hit List Memory onto the data bus.

PUSH Source: FSM. Destination: OUT REGISTER, DS_ GEN. Explanation: It latches current DATA[0-20], EPOUT, and EEOUT into OUT REGISTER at next clock rising edge. It also makes Data Strobe signal to be sent on output cable during next clock cycle.

PUSHVME Source: VME INTERFACE. Destination: OUT REGISTER. Explanation: this signal from VME commands to send one word on the output stream.

RREVME Source: VME interface.. Destination: ???. Explanation: "Road fifo Read Enable from VME" generates a RR pulse to read one word from the Road-FIFO during the next clock cycle. The VME interface has a simple FIFO controller that generates RREVME. RREVME is logically ORed with the action of the Road-FIFO Controller RFCTR to generate the RR pulse to the Road-FIFO (see Section ??? and Figure ???). The result is that the HPOP-HPOK protocol is bypassed when the input FIFO is read by VME. In this case the FIFO is read independently of the empty flag status (RFE), which is also read by VME.

RVME Source: VME interface. Destination: RI to VME buffers. Explanation: "Road stream to VME bus" enables output from the *road* transparent Latch (HL) to the VME data bus.

SSWR Source: VME base address decoding. Destination: SS Map. Explanation: "SS Map WRite" signals that VME wants to write the SS Map RAM.

TMOD Source: VME interface. Destination: FIFO controllers, VME MUXs. Explanation: "Test MODe" puts the *Hit Buffer* in Test Mode. The *Hit Buffer* is in Test Mode as long as TMOD is TRUE.

W_[0-7] Source: W GEN in Hit List Control. Destination: Hit Count. Explanation: "Write" signals to the Hit Count TAG RAM.

Block Names

AM Map 1M x 16bit fast RAM. The 17 LSb of the address come from the FIFO REGISTER???, the 3 MSb from the output of COUNTER1. Provides in output the SuperStrip address for the current *road*-Layer combination. The output FFFF is reserved to flag the "invalid *road*" condition.

CLFIFO this box generates the proper signals to clear the two FIFO, it is activated by INIT.

COUNTER1, COUNTER2 these are 3 bit synchronous counters, they increase by 1 at each clock rising edge if INCR is high, and are zeroed at clock rising edge if CLR is high. Only used in Read mode. COUNTER1 keeps track of which layer is being processed, and COUNTER2 keeps track of which *hit* word in that layer is being sent to the output.

EE REGISTER + REG0 This box contains a 21-bit register where the *End Event* word is built before sending it in output???. It contains logic to generate the parity bit for the output stream, check the input streams parity, and generate the parity error bit and flag, as described in section 10.1. REG0 is a VME register, at base address 0, containing the error bits.

FIFO REGISTER 23-bit positive edge triggered register with high-active Clock Enable.

FSM this box indicates the HB finite state machine, thoroughly described in the following of this document. It has been put in the diagram to remind that it is part of the XEPLD logic.

HFCTR, RFCTR are the two FIFO Controllers. They contain FSM's and logic to handle FIFO reading as described in SVT note 34 ("Input FIFO reading for SVT modules"). A **special note** is needed about the handling of the two input streams. The following of this paragraph assumes familiarity with the content of SVT note 34. That document assigns a transparent latch and a register to the reading of one FIFO. Since here there are two FIFO's which are only read in two separate phases of the *Hit Buffer* operation, it is possible to partially combine the data paths, and use only one register multiplexing the transparent latch outputs. The transparent latches are implemented using the input latch of the XEPLD and the two latch outputs are multiplexed using the XEPLD internal logic. This makes it impossible to start both

FIFO Controllers from state S3, since they may both attempt to write into the common register at the same time. For the *Hit Buffer* case, this conflict can be avoided by having the Hit-FIFO Controller start in state S3 and the Road-FIFO Controller start in state S0.

HIT LIST CONTROL (HLC) this box represents a fairly complex logic, it is described in detail in the last section of this document.

HIT LIST MEMORY (HLM) 64k x 21bit fast RAM with high-active output enable (OEHIT). The 13 MSb of the address come from the ADD MIDDLE LATCH, the 3 LSb from the HLC. The 21 data bits come from the DATA MIDDLE LATCH.

HL,RL the two small rectangular boxes in the top left corner called HL and RL are the transparent latches used in the FIFO reading respectively for the *hit* stream, and the *road* stream. They are controlled by HR and RR signals (latches are frozen when these signals are low, transparent otherwise). These are simply the inverse of the HR_ and RR_ signals that provide the R_ reading pulse (low-active) to the Hit-FIFO and Road-FIFO respectively.

ML ADD Middle Latch for address, 16-bit positive edge triggered register with high-active clock enable (LEADD). On output the 13 MSb are sent to the HLM (ADD[3-15]), the 3 LSb to the HLC (SS[0-2]).

ML DATA Middle Latch for data, 21-bit positive edge triggered register with high-active clock enable (LEDATA) and high-active output enable (OEDATA). When in Write mode, the *hits* are written here before being loaded into the Hit List RAM. In Read mode, the *road* number is written here to be included in the End Packet word.

MX1 and SWAP 23-bit multiplexer. When WR is high (Write mode), output MO (Multiplexer Out) is equal to HI (HitIn). When WR is low (Read mode), MO output is RI (RoadIn).

MX2 1-bit multiplexer. Output =FHLE (LE signal from Hit-FIFO controller) when WR=1. Output=FRLE (LE from Road-FIFO controller) when WR=0.

OUT REGISTER 23-bit positive edge triggered register with high-active clock enable (PUSH).

SS Map 128k x 16bit fast RAM. Provides in output the SuperStrip address corresponding to each *hit*. The output FFFF is reserved to flag the "invalid Hit" condition.

SWAP this block swaps back the two fields ϕ sector and *road* number of a word from the *road* stream, so that during Read Mode HRIN[0-16] correspond to the *road* number and HRIN[17-20] to the ϕ sector.

TAG RAM 8 chips of 8k x 8bit fast Cache TAG RAM connected as shown in Figure 13. Note that the chips are used as if they were 8k x 1bit. These chips continuously compare the input data (kept fixed to 1 in our case) to the content pointed by the address lines and set the M (match) line to 1 or 0 according to successful/unsuccessful comparison. IDT71B74S10 chips are used. The CLRTAG signal resets all TAG RAM content to zero. Probably the CLRTAG signal will have to be asserted for two consecutive clock cycles, to make it long enough to satisfy chip specifications. Each location of the TAG RAM correspond to a SuperStrip and keeps track of the number of *hits* written in the HLM for that SuperStrip.

DS_ GEN this box indicates the logic that generates the proper Data Strobe signal on the output cable. Assertion of this signal will be controlled by PUSH.OR.PUSHVME.

HOLD_ SYNC this box indicates the logic that generates the IHOLD signal from the HOLD_ signal incoming on the output cable. This signal originates from the downstream FIFO Half_Full flag. It will have to be suitably latched (twice ?) to make it synchronous and avoid metastable states, and inverted if needed. The internal IHOLD signal is high-active.

10 Checking for Integrity and Consistency

The last word of each event has the same format on both input streams.

It contains a 1-bit parity code that needs to be checked on the fly for every event in order to promptly detect the occurrence of data corruption. The parity code is computed by taking the bitwise XOR of all data words in the event (22 bits/word, excluding the EE bit) excluding the *End Event* word.

The HB must compute the parity code of incoming data in both input streams and compare it with the code stored in each *End Event* word.

The HB must also compute the parity code for the output stream and include it in the *End Event* word.

The EE word also contains an 8-bit event tag. The event tags in the two input data streams (*hits* and *roads*) must be compared to check that they actually belong to the same event. If tags are found to be different, then we know that the system is somehow out of sync and therefore output data has no meaning. This error condition is flagged by an error bit in the output stream (see Section 10.1).

The HB must include the event tag in each EE word in the output stream.

If the event tags of the two input streams are different, than the tag received in the *hit* stream is included.

10.1 Exception Handling

A number of error conditions are detected within the HB.

1. Fifo Overflow: the "hold" mechanism in the communication protocol should prevent the input Fifo's from becoming full. If the "Fifo Full" signal is detected by the input control logic, it is a symptom that something is going wrong in the data transfer and that part of the information has probably been lost.

2. Internal Overflow: when the *Hit Buffer* encounters a situation which causes data to be lost inside the *Hit Buffer* itself. In the *Hit Buffer* only one such instance is foreseen:

    (a) Buffer Overflow: the amount of memory allocated to a particular SuperStrip in the Hit List Memory for storing *hits* is insufficient and some *hits* of that SuperStrip have been lost.

3. Invalid data: when the *Hit Buffer* detects a problem with input data. Two possible errors are flagged:

    (a) Invalid Hit: a *hit* received on the input stream does not correspond to a valid SuperStrip address.

    (b) Invalid Road: a *road* received on the input stream does not correspond to a valid SuperStrip address.

4. Parity Error: the parity code in the EE word in one (at least) of the input streams does not check. Input data is probably corrupted.

5. Lost Sync: the event tags from the two input streams (*hits* and *roads*) do not match.

When an error condition is detected the following actions are taken:

1. A corresponding error flag is set in the Error Flag register. This register is readable and clearable through the VME interface. Bit assignments are specified in the section dealing with VME registers.

2. An error flag is set in the EE word of the output event. There is one such bit for each of the 5 error classes described above. The error flag bits in the output event are obtained by ORing the corresponding bits in the input events from the two input streams with the possible error conditions detected within the HB while processing that event.

    The *End Event* word format is:

    [0-7] Event tag
    [8] Parity
    [9] Parity Error
    [10] Lost Sync
    [11] FIFO Overflow
    [12] Invalid Data
    [13] Internal Overflow
    [14] Truncated output (not used inside HB only propagated) [15-18] Spare error flags
    [19-20] Level 2 Buffer [21] End Packet bit: set to 1
    [22] End Event bit: set to 1

Figures 17 and 18 show how the *End Event* REGISTER of Figure 16 is implemented. The *End Event* word is built in the EE REGISTER as follows:

END EVENT REGISTER - Part I

15-SEP-96

	20	8	7 0
EE WORD:	EVENT ERRORS	P	EV TAG

Bit [8-0] : Event Tag & Parity Generator

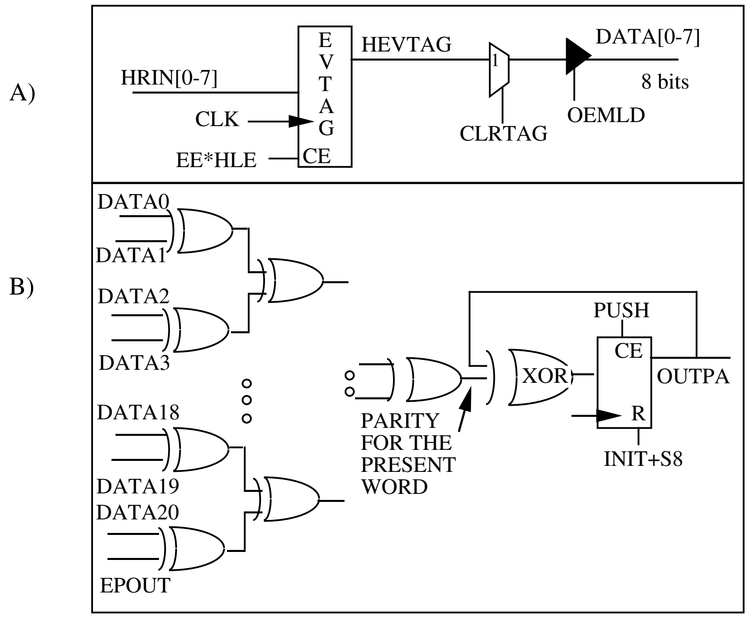

Figure 17: Parity block and *End Event* register

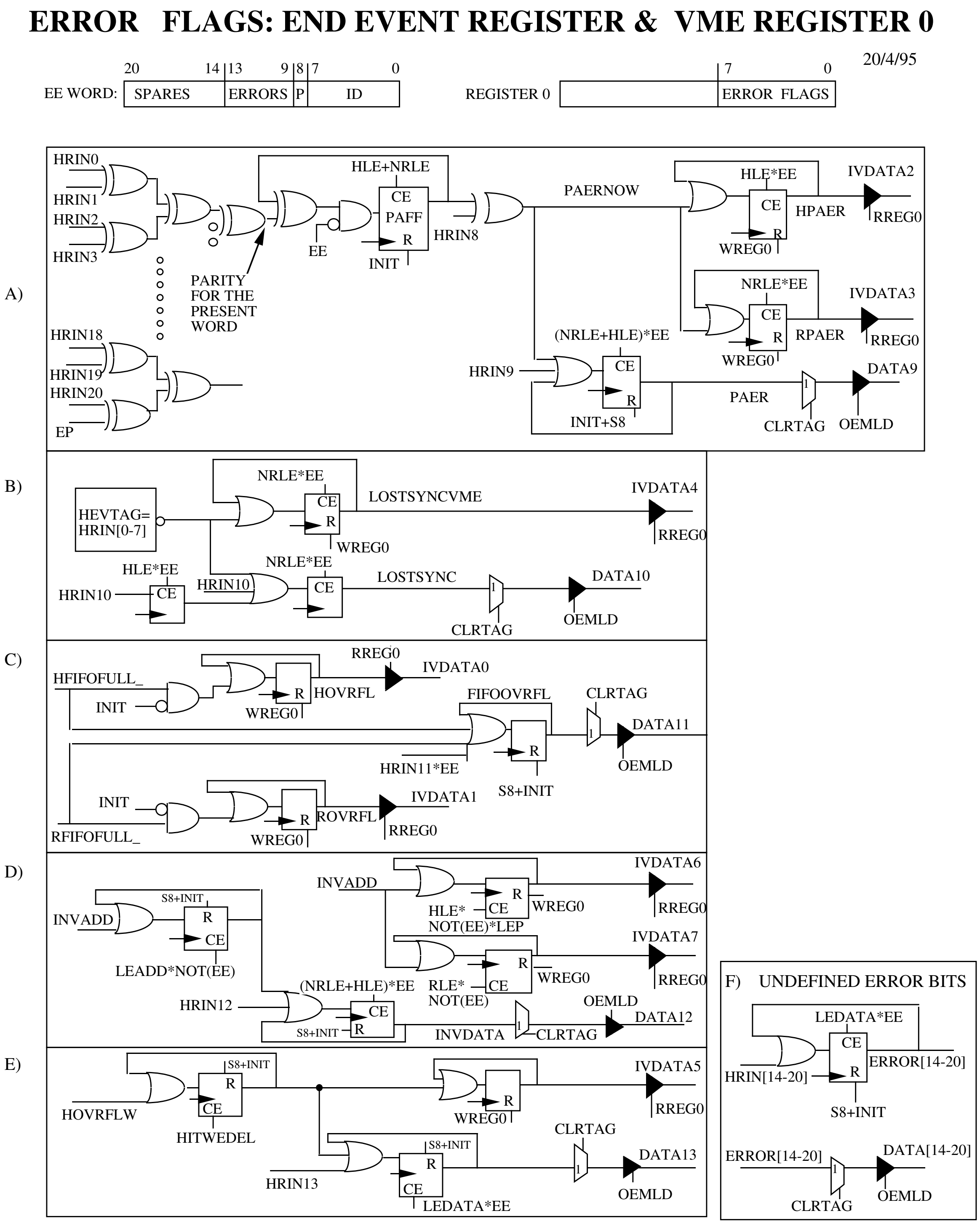

Figure 18: Parity block and *End Event* register

- Bits 0-7 are the Event Tag used to identify the event (see Figure 17A). The Event Tag is included in the *End Event* word of each stream and is used to verify the consistency of data on different streams. The information in the EVTAG register is copied from the *hit* stream (HRIN bus during the Write Mode) enabling the clock with the HLE signal (HLE means a word is available from the *hit* stream) ANDed with the EE signal that ensure the *End Event* word is present on the HRIN bus.

- Bit 8 is the parity computed on all words (22 bits/word since the EE bit is excluded) sent in output to the downstream board, excluding the EE word (see Figure 17B). The parity of each word on the DATA bus is added to the parity of all previous words in the XOR and the feed back flip-flop. The result is stored for each new word in the OUTPA flip-flop gated by the PUSH signal. The register OUTPA is reset once per event (S8 signal means that the FSM is in state 8) and when the HB is initialized (INIT).

- Bits 9-20 are dedicated to error flags. Only the bits 9-13 have meaning for the HB, but all of them must be correctly propagated to the downstream boards because they could have been seen by other modules upstream. Figure 18A-F shows the hardware implementation of this part of the *End Event* register. Since the VME register 0 (see the VME section) also stores information on error conditions, its hardware description is included in the same figure. The integration time for the EE register error flags is one event so the reset of the corresponding flip-flops is sent once per event and when the board is initialized (S8+INIT). The integration time for the VME register is defined only by the VME writing cycle on that register, so the reset of the corresponding flip-flops is done by WREG0. In APPENDIX 18 follows the detailed descriptions of the error flag implementations.

11 Spy Buffers

Data flowing through each input and output stream are copied into spy buffers. There is one such buffer for each input and output stream (a total of 3 for the *Hit Buffer*). The purpose of these buffers is to spy all the data going by without causing any interference to the functioning of the *Hit Buffer*. Similar buffers will be implemented in all modules of SVT. They act as built in logic state analyzers and will help system monitoring and diagnosis. The contents of all buffers can be freeze at any time (e.g. on error detection) to take a snapshot of all data that went through every SVT module.

The buffers are implemented as RAM banks where a pointer is incremented each time a data word is popped from the input FIFO or pushed to the output stream. The RAM is 23 bit wide to hold 21 bit data plus EP and EE. The pointer is the address where the next data will be written. When the pointer overflows, it simply wraps around: incoming data will overwrite the buffer contents over and over in a circular fashion. At POWER ON the pointer is reset to zero, an overflow flag is set the first time the pointer wraps around so that we know how much valid data the buffer contains. The buffers can be in one of two modes: SPY or FREEZE.

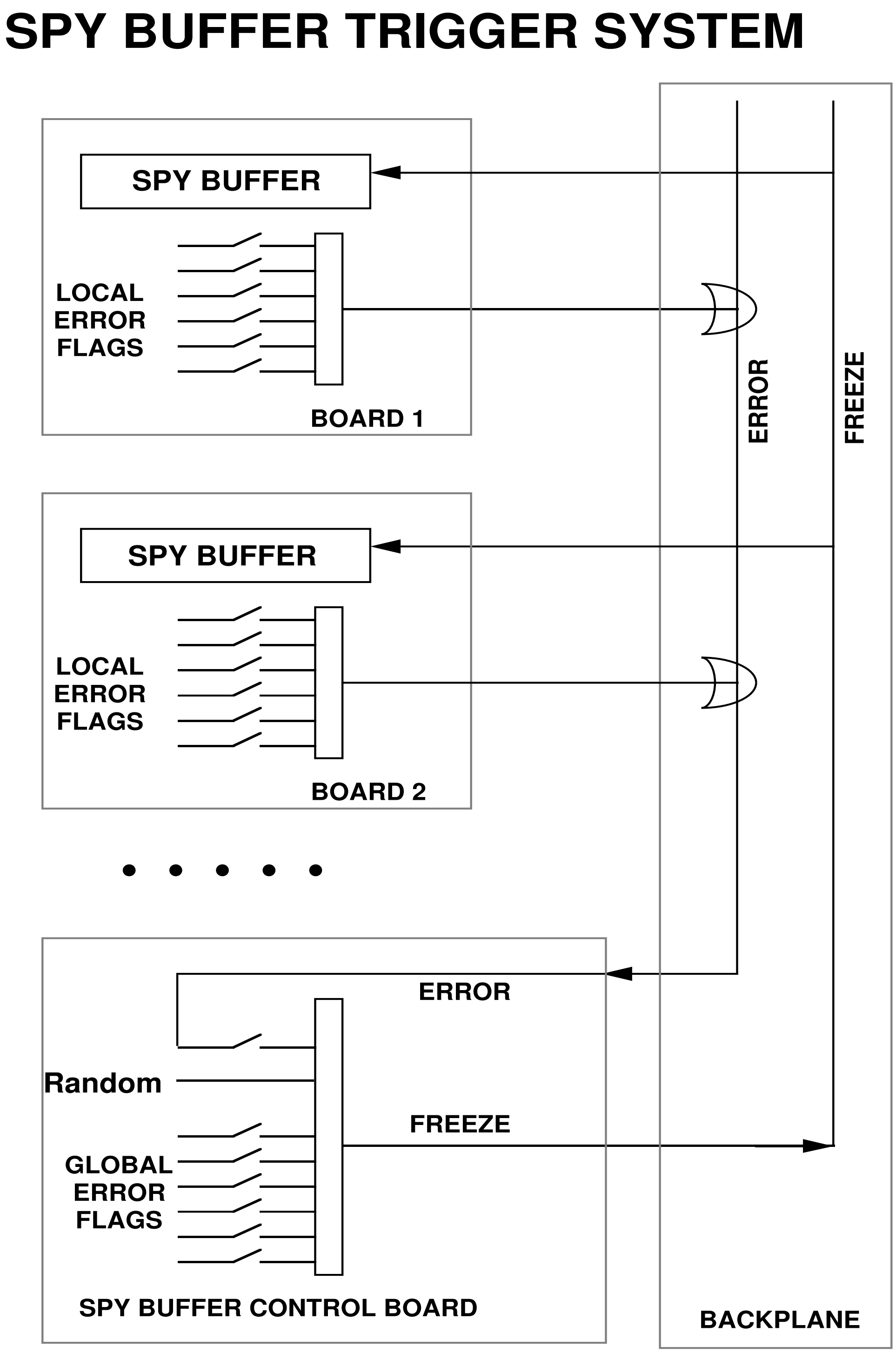

Figure 19: Spy buffer trigger system

When in SPY mode data are continuously copied into the RAM, when in FREEZE mode copying is suspended and the contents of all the buffers can be read through the VME interface without causing any interference to the data flow. The current value of the pointer and the overflow flag can also be read. Both the pointer and the overflow flag can be cleared through VME.

The buffer status is controlled by a single line (FREEZE) in the backplane as shown in Figure 19, all buffers being controlled by the same line. The FREEZE line is driven by the Spy Buffer Control Board (not part of the *Hit Buffer*).

The ERROR line in the backplane (BERRORLINE_) is an active low signal set by the OR of a number of error conditions (ERROR LINE in Figure 20). Each error condition can be enabled or disabled through VME. The BERRORLINE_ is an open collector line that acts as a wired OR of all enabled error conditions of all the modules in a crate: it is received by the Spy Buffer Control Board (not part of the *Hit Buffer*) and can be used to trigger the FREEZE line.

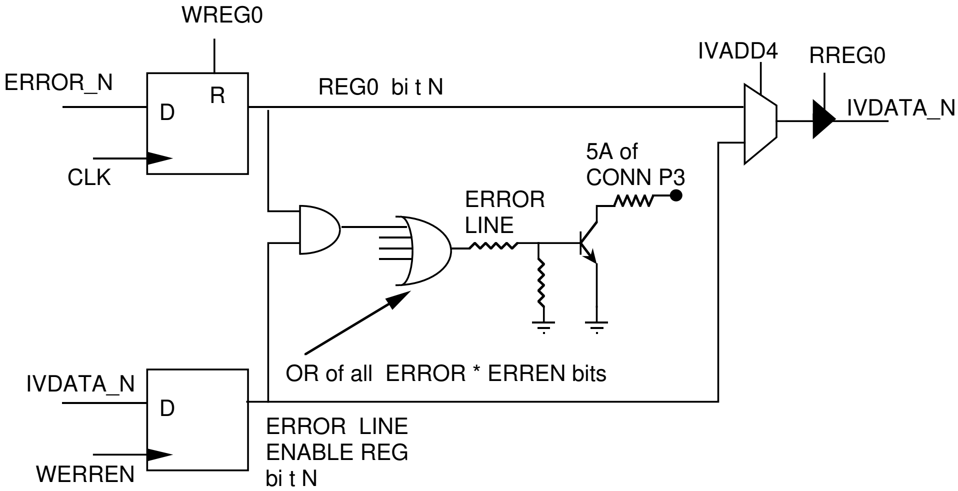

Figure 20: ERROR line generation

Figure 20 shows how the ERROR line is implemented.

11.1 Spy Buffer Implementation

The Spy Buffer control logic performs these main functions:

1. Generates the addresses (see block ADD GEN in Figure 21) for the Spy Buffer memories. The two input stream Spy Buffers (Hit and Road Spy Buffers) share both the address and the data bus since they are used (written or read) at different times. The address is provided by VME (IVADD[0-16] in Figure 21) or by internal counters, one per each stream, depending on which status is the board, FREEZE or SPY mode.

2. Generates the control signals to the memories (see block RAM CNTR in Figure 21). Chip selects are provided to reduce power dissipation on the

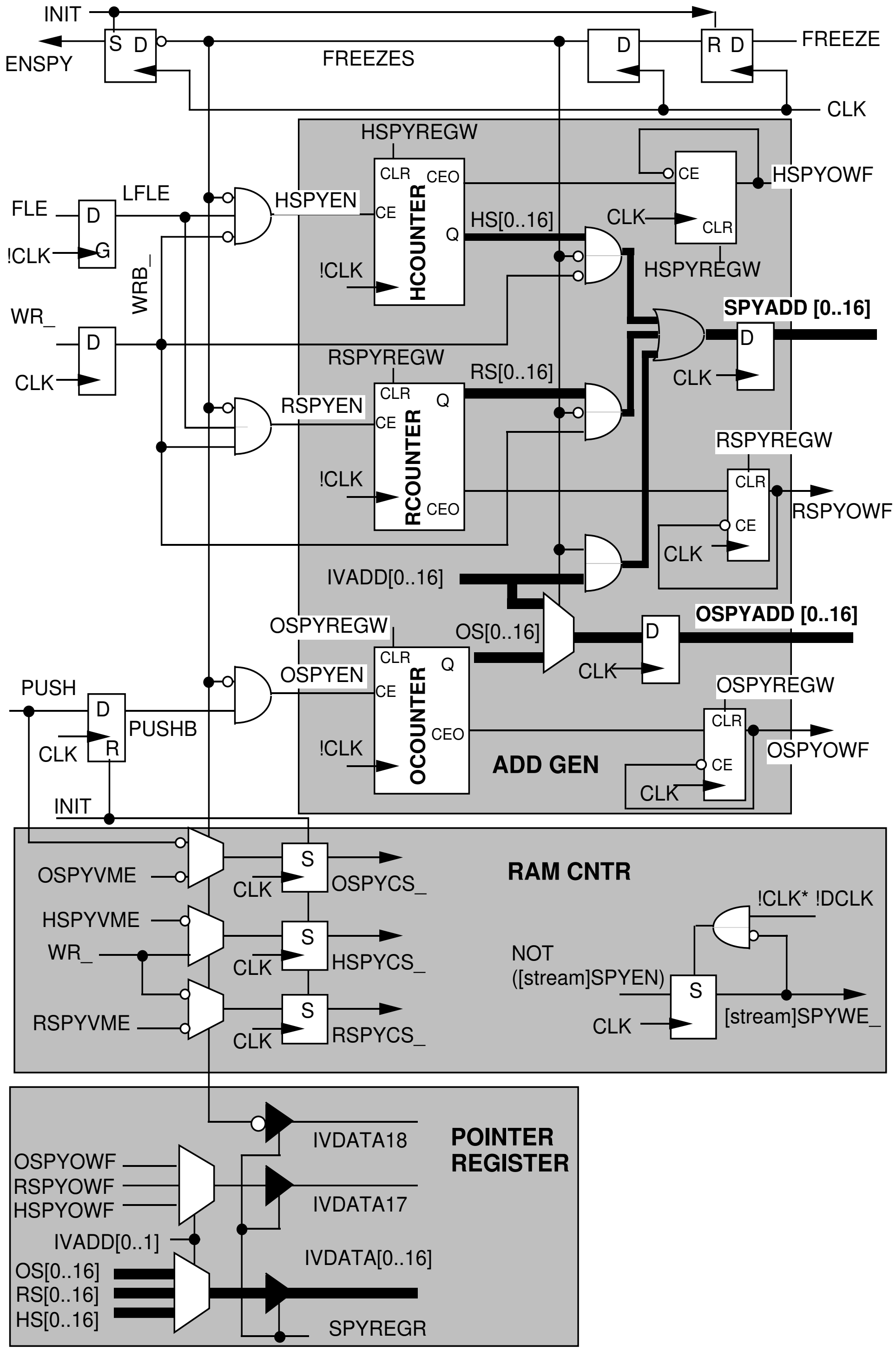

Figure 21: Spy buffer control logic

board. Signals to write on the memories ([stream]SPYWE_ in Figure 21) are generated so that each word received in the input streams is written in the Spy Buffer after being latched in the FIFO REGISTER (FLE is the latch enable for the FIFO REGISTER) and each word to be sent on the output stream is stored in the Spy Buffer after being latched in the OUTPUT REGISTER (PUSH is the latch enable to the OUTPUT REGISTER). The output control of the memories (Output Enable generation) is totally dependent by the VME logic and is described in section 14.2.

3. Generates the bidirectional data buses to the Spy Buffer memories (see in Figure 16 HRSPY[0-22] for the Hit and Road Spy Buffers and OSPY[0-22] for the Output Spy Buffers) so that words flowing in the streams can be written in the Buffers and can be read by VME (IVDATA bus in Figure 16).

4. Spy Buffer pointer information, the counter Overflow bit and the FREEZE bit are provided in output under the VME control to implement the Pointer Registers described in the Section 14.1 (see the block POINTER REGISTER in Figure 21).

12 Init

An Init cycle is initiated through VME by writing into the appropriate address, or when the INIT line is asserted high on the backplane input.

The module shall not respond to the backplane init signal if it lasts less than 100 nsec.

Init performs the following functions:

1. Resets all FSM's to their initial state;

2. Clears the TAG RAM;

3. Clears the input Fifo's;

4. Clears the Parity bit and the Error Flags in the *End Event* register;

5. If the HB was in the "Test Mode" state (see below) prior to the Init, then this state is exited;

6. Following an Init the HB must be ready to accept the next event within 1 microsecond or less;

7. The contents of the look-up table RAM's are not changed by the Init cycle.

In Figure 22 Init is implemented as the most significant bit ($Q2$) of a 3-bit counter that is set to the initial value [1,0,0] by the signal Active-Init, the WINIT signal (from VME or from the backplane) synchronized by the following flip-flop. As a consequence INIT will be active all the time the Active-Init signal will be on, plus 4 clock cycles following the Active-Init trailing edge. This implementation guarantees that INIT is long enough to reset the TAG RAM.

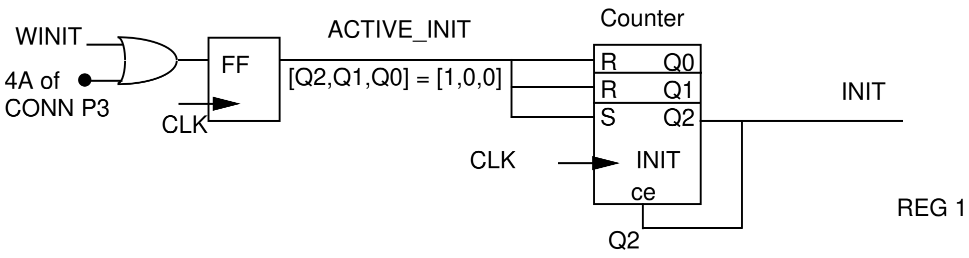

Figure 22: INIT as the MSB bit of a 3-bit counter

13 Test Mode

The HB can be set in "Test Mode" by writing into the appropriate VME address (see below).

While the HB is in Test Mode:

1. all the finite state machines are reset to the initial state and halted there
2. the input FIFO's are available for reading through VME
3. the output register is available for writing through VME
4. the SS Map and AM Map RAM's can be read/written through VME.

The Test Mode is exited only following an Init.

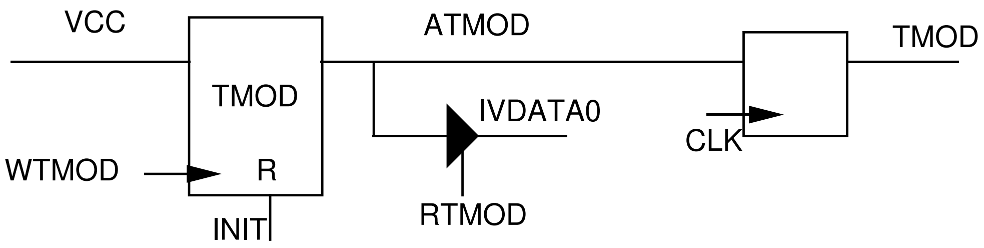

Figure 23: TMOD

Figure 23 shows how TMOD is generated. An asynchronous signal (ATMOD) is activated by a VME writing cycle to the register 2. ATMOD is synchronized by the following flip-flop to generate the TMOD signal.

Section 7 shows the implementation of the main functions enabled by TMOD.

14 VME Interface

The upper 8 bits in the address is used as "module address" while the lower 24 bits are used as "internal address".

The module address is set by dip switches. Each RAM is mapped to a contiguous internal address range and is accessible through block transfer. FIFO's are mapped to a contiguous address range in order to facilitate the use of block transfer. Address modifier codes that are supported are:

09:	Extended non-privileged data access
0B:	Extended non privileged block transfer

The following operations can be performed asynchronously by VME only if the HB is in Test Mode:

1. reading or writing the output register;
2. reading the Hit or Road Fifos;
3. reading or writing the AM or SS maps;

All other VME operations can be performed regardless of whether the HB is in Test Mode or not.

14.1 VME Internal Address Map

Unless otherwise specified, the following rules hold:

- registers are read/write
- status and control signals are in positive logic (1=TRUE, 0=FALSE)

XX000000: Error Flags
[0] Hit FIFO overflow
[1] Road FIFO overflow
[2] Hit Parity Error
[3] Road Parity Error
[4] Lost Sync
[5] Buffer Overflow
[6] Invalid Hit
[7] Invalid Road
A write to this address (data is irrelevant) will clear all error flags.

XX000004: Init
This is a write-only register.
A write to this address (data is irrelevant) will cause an Init cycle.

XX000008: Test Mode
A write to this address (data is irrelevant) will put the HB into Test mode.

XX00000C: Hit-FIFO Status
This is a read-only register.
[0] FIFO Empty_ (active low)

[1] FIFO Half Full_ (active low)
[2] FIFO Full_ (active low)

XX000010: Road-FIFO Status
This is a read-only register.
[0] FIFO Empty_ (active low)
[1] FIFO Half Full_ (active low)
[2] FIFO Full_ (active low)

XX000014: Output Status (Hold)
This is a read-only register.
[0] syncronized HOLD_ (active low)

XX000018: FSM State
This is a read-only register: it contains the current state number of the finite state machine that controls the HB

XX000020: FSMH State
This is a read-only register: it contains the current state number of the finite state machine that controls the input Hit-FIFO.

XX000024: FSMR State
This is a read-only register: it contains the current state number of the finite state machine that controls the input Road-FIFO.

XX000028: Output Register
A write to this address when the HB is in Test mode will cause a data transfer to occur immediately on the output stream. Data will be transferred regardless of the status of the IHOLD signal.

XX000030: Hit Spy Buffer Pointer
[0-16] Pointer
[17] Overflow flag
[18] Spy Status (0 = FREEZE, 1 = SPY)
A write to this address (data is irrelevant) will reset the pointer to zero and clear the overflow flag. The status cannot be changed: it is controlled only by the status of the FREEZE line on the backplane.

XX000034: Road Spy Buffer Pointer
[0-16] Pointer
[17] Overflow flag
[18] Spy Status (0 = FREEZE, 1 = SPY)
A write to this address (data is irrelevant) will reset the pointer to zero and clear the overflow flag. The status cannot be changed: it is controlled only by the status of the FREEZE line on the backplane.

XX000038: Output Spy Buffer Pointer

[0-16] Pointer
[17] Overflow flag
[18] Spy Status (0 = FREEZE, 1 = SPY)
A write to this address (data is irrelevant) will reset the pointer to zero and clear the overflow flag. The status cannot be changed: it is controlled only by the status of the FREEZE line on the backplane.

XX000040: Error Line Enable
Each bit, if set, enables the corresponding error flag to drive the ERROR line on the backplane.
[0] Hit FIFO overflow
[1] Road FIFO overflow
[2] Hit Parity Error
[3] Road Parity Error
[4] Lost Sync
[5] Buffer Overflow
[6] Invalid Hit
[7] Invalid Road

XX100000 - XX1FFFFF: ID Prom (256K x 8)

This is a 256k bites EPROM that contains ASCII text. It is read only.

XX200000 - XX20FFFF: Hit-FIFO

This read-only address range is used for mapping the Hit-FIFO so that it can be read easily with a VME block transfer. The HB must be in Test mode for this operation to yield meaningful results. The size of the FIFO in the current implementation is 4K, the remaining address space is reserved for possible future expansion. The FIFO will be read once for every address, regardless of its Empty Flag status. The Empty Flag will be included in the read data as most significant bit:
[0-22] 23-bit data word from FIFO
[31] 0 if data have been read from an empty FIFO or the last word in the FIFO has been read, 1 if more than one word was in the FIFO.

XX210000 - XX21FFFF: Road-FIFO

This read-only address range is used for mapping the Road-FIFO so that it can be read easily with a VME block transfer. The HB must be in Test mode for this operation to yield meaningful results. The size of the FIFO in the current implementation is 4K, the remaining address space is reserved for possible future expansion. The FIFO will be read once for every address, regardless of its Empty Flag status. The Empty Flag will be included in the read data as most significant bit:
[0-22] 23-bit data word from FIFO
[31] 0 if data have been read from an empty FIFO or the last word in the FIFO has been read, 1 if more than one word was in the FIFO.

XX280000-XX2FFFFF: Hit Spy Buffer
128k words 23 bit each
[0-20] Data
[21] End Packet
[22] End Event

XX300000-XX37FFFF: Road Spy Buffer
128k words 23 bit each
[0-20] Data
[21] End Packet
[22] End Event

XX380000-XX3FFFFF: Output Spy Buffer
128k words 23 bit each
[0-20] Data
[21] End Packet
[22] End Event

XX400000 - XX47FFFF: SS Map

This is a 128K times 16 bit RAM that holds the look-up table for the mapping from *hit* coordinate to SuperStrip as described above. The "invalid *hit*" condition is flagged by the look-up output being FFFF.

XX800000 - XXBFFFFF: AM Map

This is a 1M times 16 bit RAM that holds the look-up table for the mapping from *road* and layer to SuperStrip as described above. The "invalid *road*" condition is flagged by the look-up output being FFFF.

14.2 VME Hardware Implementation

————Sorry this is missing—————-

15 Design Partitioning

The *Hit Buffer* board makes use of large scale programmable integrated devices for all parts except RAM's and FIFO's.

All of the *Hit Buffer* control logic is implemented in the XILINX 7300 XEPLD chips.

The XEPLD programming is in ABEL, with few exceptions where the ORCAD schematics are used together with ABEL files.

Figure 24 shows how the project is divided into XILINX chips.

———— Sorry the description of the Figure is still missing —————

In Figure ?? is the list of used chips with important informations:

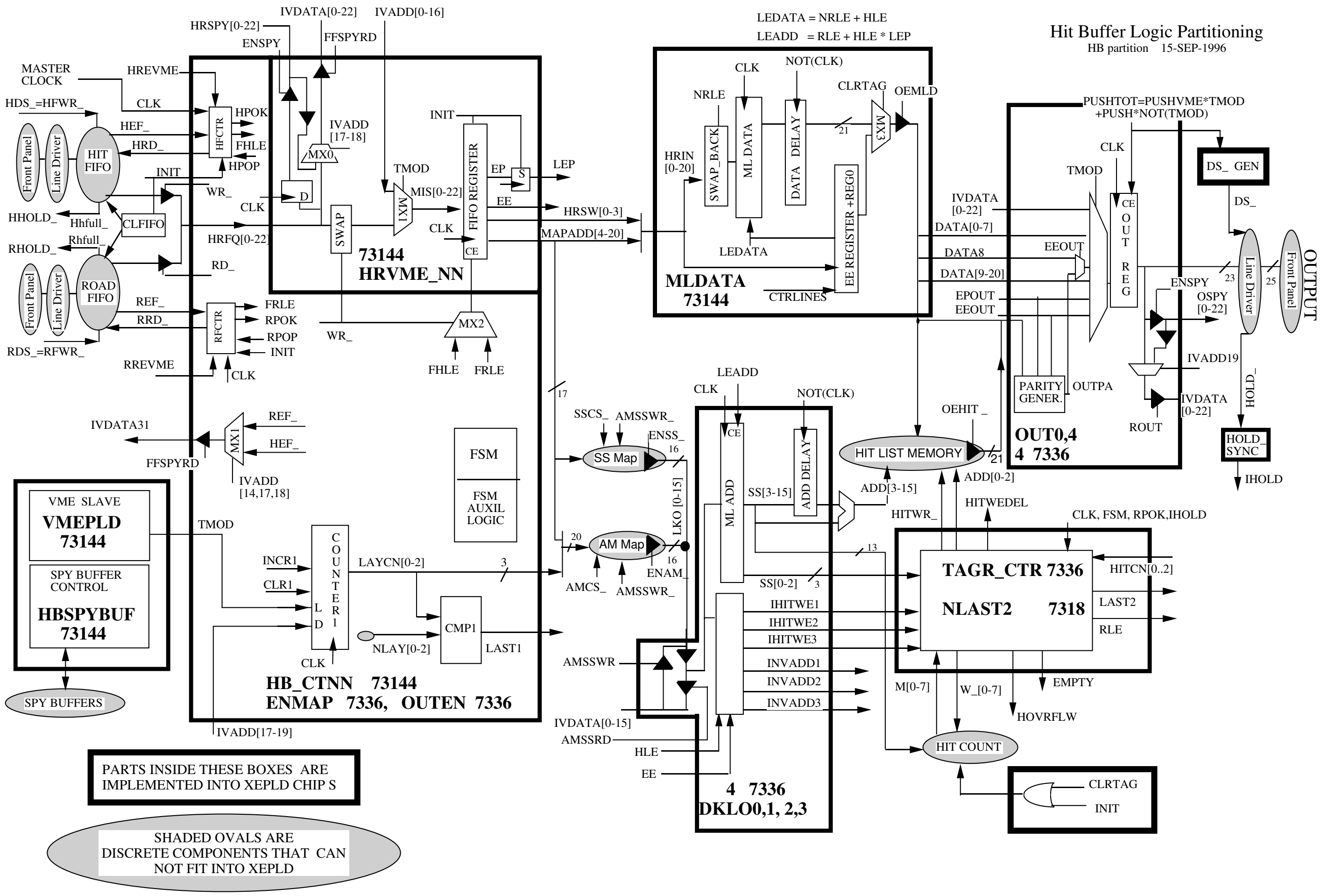

Figure 24: *Hit Buffer* logic block diagram partitioned into XEPLD chips

1. a measure of the amount of used resource (SIZE FACTOR, to be compared to the chip macrocell number, that is 144 for 73144 and 36 for 7336);
2. the number of still available free pins (FREE INPUTS, OUTPUTS and I/Os);
3. the number of used over the number of available fast inputs (USED FI/AV FI).

16 Input/Output Interface

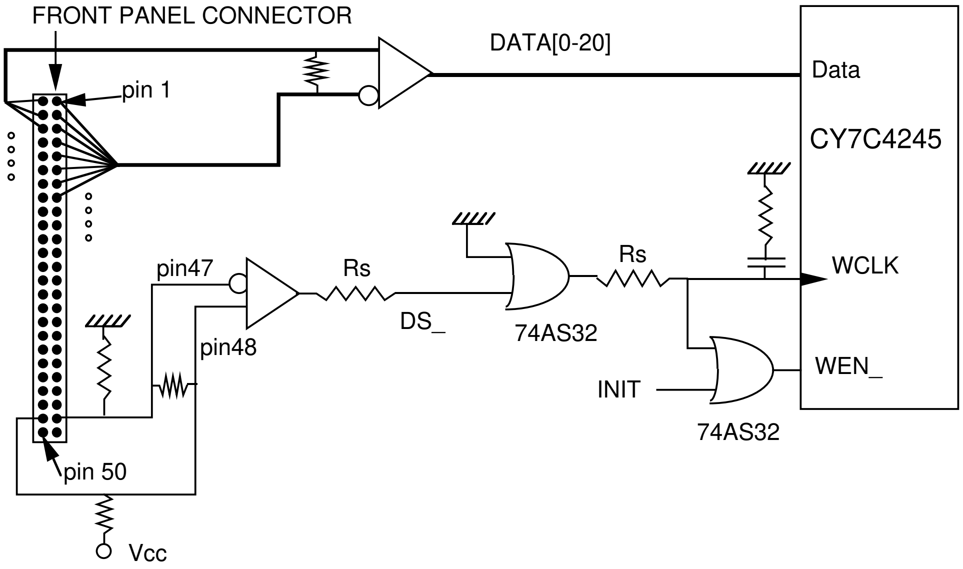

Figure 25: The DS_ signal and DATA bus from the input connector to the CY7C4245-10 fifo.

The input FIFO is CY7C4245, 4Kx18 synchronous fifo from CYPRESS. Data are sampled on the rising edge of the strobe; Figure 25 shows how the DS_ (data strobe) is received from the connector and used to write in the fifo (this is exactly the same of what it is done on the Merger board). The differential lines are sent to the AT&T receivers 1141LF. Since the fifo is very fast and sensitive to spikes the gate 74AS32 has been inserted to filter glitches shorter than few nanoseconds; it adds a 5.8 ns maximum delay on the DS_ line compared to the data lines that go directly from the receivers to the fifo. The DS_ line to be sent to write clock WCLK is also provided with both serial and parallel terminations. The DS_ itself is used to produce the active low write enable line (WEN_) to the fifo. Since the fifo is very fast, it is important to enable it only for a short time before writing. A small delay between the WCLK and WEN_ lines is due to the second gate 74AS32.

Figure 26 shows the timing with which the DS_ signal must be generated by an upstream board to be compatible with the writing function into the CY7C4245-10 fifo. To use the same signal DS_ to generate both WCLK and WEN_, an active low

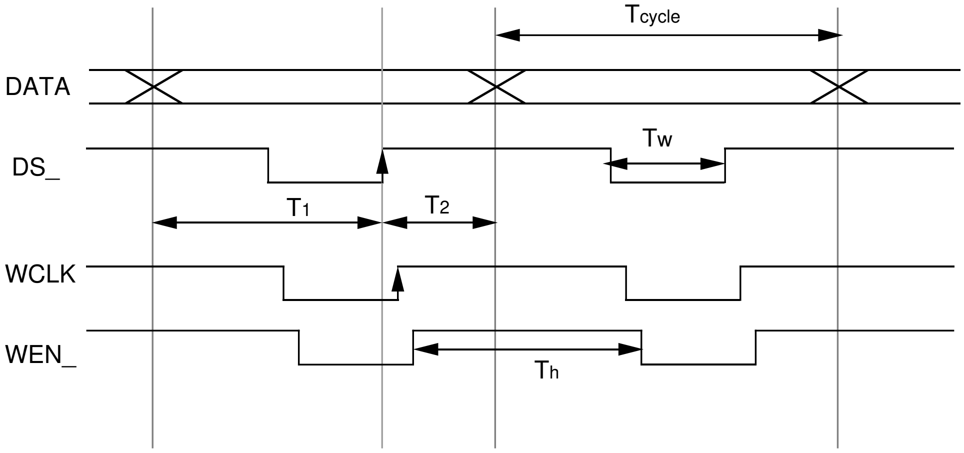

Figure 26: The DS_ signal timing relative to the DATA bus timing.

signal is necessary. This signal will enable the clock with the correct set-up time and will strobe data on the rising edge, at the end of the pulse. To understand the limits for T1, T2 and Tw in the Figure, we need also to consider some of the fifo switching characteristics shown in table 9.

Description	Min.	Max.	Unit
Clock Cycle Frequency		100	MHz
Data Access time	2	8	ns
Clock Cycle time	10		ns
Clock HIGH time	4.5		ns
Clock LOW time	4.5		ns
Data Set-up time	3		ns
Data Hold time	0.5		ns
Enable Set-up time	3		ns
Enable Hold time	0.5		ns

Table 9: Important switching characteristics of the Fifo CY7C4245-10

The minimum clock low time is 4.5 ns, but the width of the signal needs to be at least Tw_min=8.8 ns since WEN_ has a 3 ns Set-up time and can be 5.8 ns delayed compared to the WCLK (gate 74AS32 in Figure 25). However the small delay of WEN_ (the minimum 73AS32 delay is 1 ns) with respect to WCLK guarantees the Enable Hold time (0.5 ns) is satisfied.

The minimum clock high time the fifo accept is 4.5 ns, but in our case the clock high time Th will be longer since the period is roughly Tcycle=33-40 ns and Th=Tcycle-Tw.

The minimum value for T1 is given by the Data Set-up time (3ns), while the minimum value for T2 is given by the Data Hold time (0.5 ns) added to the 74AS32 maximum delay (5.8 ns). We add to these numbers few nanoseconds to guarantee that small routing differences won't prevent the writing function. In conclusion we recommend the minimum values for T1 and T2:

$$T1\_min = 6 \text{ ns}; \quad T2\_min = 10 \text{ ns};$$

As a result for a Tcycle=40 ns, the DS_ rising edge can have a delay T1 between 6 and 30 ns.

The electrical implementation of the input/output drivers for the front panel is such that these requirements are followed:

1. The DS_ sent on the output cable is adjusted in time and width with respect to the data word being sent such that it can be succesfully used as input to the logic shown in Figure 25 located into a downstream board. The *Hit Buffer* generates the DS_ signal for the output connector as described in Section 7 and shown in Figure 11. The DS_ timing for the *Hit Buffer* is described for a clock cycle of 39 ns by the parameters values (see Figure 25):

   $$T1 = 29 \text{ ns}; \quad T2 = 10 \text{ ns}; \quad Tw = 19 \text{ ns};$$

   Since the DS_ generation is implemented in a programmable chip, the timing can be adjusted if necessary. The drivers used on the *Hit Buffer* upstream the output connector are the AT&T 26C31CD.

2. If an input cable is disconnected, the corresponding DS_ line will be set to high (pullup and pulldown are applied to the input differential lines, see Figure 25).

3. If the output cable is disconnected, the input HOLD_ signal will be set to TRUE (that is the HOLD_ line after the receiver will be low; pullup and pulldown are applied to the input differential lines).

4. A dip switch is implemented on the board which allows to disregard the status of HOLD_ (data will be pushed out regardless of HOLD_).

17 Physical Implementation

The *Hit Buffer* is implemented as a nine unit Eurocard board 400mm deep (9Ux400mm). Multiwire technology has been used for electrical connections. The standard 32 bit VME interface is implemented using two connectors on the backplane (P1 and P2). Connector P3 is used for the following signals: ERROR, FREEZE, INIT. Board thickness is 2.36 ± 0.18 mm. The front panel includes:

- 3 Flat cable connectors, 50 contacts each for input and output streams. Connector type 3M-3433D202 or equivalent.
- 1 Green LED for DTACK activity
- 3 Green LED's for Data Strobe activity (one for each I/O stream)
- 3 Red LED's for HOLD_ activity (one for each I/O stream)

Table 10 shows the pinout of the front panel connectors. The last 4 couples of pins are assigned to HOLD_ (Pins [50,49]), DS_ (Pins [48,47]), EE (Pins [46,45]) and EP (Pins [44,43]). The other 21 couples of pins are used for the data signals DATA[0-20] defined and described for the different streams in Section 3.

pin	use	pin	use
1	D_0^*	2	D_0
3	D_1^*	4	D_1
⋮	⋮	⋮	⋮
41	D_{20}^*	42	D_{20}
43	EP*	44	EP
45	EE*	46	EE
47	DS_*	48	DS_
49	HOLD_*	50	HOLD_

Table 10: Pinout for the front panel connectors. * indicates the line connected to the inverted pin of the driver or receiver.

18 APPENDIX

1. Bits [14-20] (see Figure 18F) are simply implemented as the OR of the corresponding bits in the *hit* and *road End Event* words since they are not generated by the HB. The flip-flop clocked when the *End Event* word for *roads* or *hits* is present on the HRIN bus (EE*LEDATA=1 where LEDATA=HLE+NRLE) is set if an error flag is present on one of the two input streams. This is the simplest implementation: all other error bits add an internal error code to the corresponding error bit of the two input streams.

2. Bit 9 is the result of the parity check on both input streams (see Figure 18A). In the PAFF Flip Flop the parity is calculated and added on all event words received in input, two times for each event: the first time is for data coming from the *hit* stream (during WRITE MODE) and the second time is for data coming from the *road* stream (during READ MODE). The parity calculated in PAFF for one stream is compared to what it is expected to be (bit 8, HRIN8, of the corresponding *End Event* word) and the result is latched in the following Flip Flop when the *End Event* word is identified (HLE*EE=1 and RLE*EE=1 for the *hit* and *road* stream respectively). On the same clock edge when EE bit is active, the PAFF Flip Flop is synchronously reset, since data from that stream are finished and a new parity calculation is going to start for the other stream. The *Hit Buffer* parity check is added to the upstream board parity check bit (HRIN9), before being latched to generate the parity check bit (see PAER in Figure 18A). While in the EE register a single bit is available for the parity check (PAER) and it is the OR of the

errors on the two input streams, in the REG0 there is space for two separate bits: the Hit PArity ERror and the Road PArity ERror.

3. Bit 10 is the result of the two input stream consistency check (Lost Sync error, see Figure 18B). The EVTAG register containing the event code of the *hit* stream (see Figure 17A) is compared to the the bits 0-7 of the HRIN bus and the result is latched when the *End Event* word of the *road* stream is present on that bus (EE*RLE=1). The values of HRIN10 at the two times HLE*EE=1 and RLE*EE=1 are ORed to the internal error flag to include in the EE register the error flags from upstream boards. The description of the implementation of the remaining error bits will be less detailed because it is very similar to the previous ones.

4. Bit 11 reports if one of the two FIFOs has ever been Full during the event (FIFOOVRFL see Figure 18C).

5. Bit 12 reports if any of the addresses to the HLM was invalid during the event (invalid DATA) (see Figure 18D). The INValid ADDress is generated when all lines of the LKO bus are high (see Figure 16).

6. Finally bit 13 reports if any SuperStrip had too many *hits* so that some of them could not be written in the HLM (HOVRFLW, see Figure 18E).