GRETA VHDL MODULES

I - TapDelay

This module implements a programmable length free running Tap Delay. The maximum speed at which it can run is 182MHz (internal speed without driving the output outside). The following table summarizes the input/output of this block:

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
LOAD	This synchronous signal indicates when we want to load a new delay length. It resets the status line to zero as well as the internal counters.
STATUS	Goes high when the Tap delay is filled up and ready to have valid input output relation for filtering. It goes low each time the LOAD signal is asserted and goes back up when ready for operation.
LENGTH(8 to 0)	The bus giving the programmable length to be loaded.
DATAin(15 to 0)	Input of the Tap delay X_n
DATAout(15 to 0)	Output of the Tap delay Xn-L

Table 1: Tap Delay I/Os

The basic architecture is as follows:

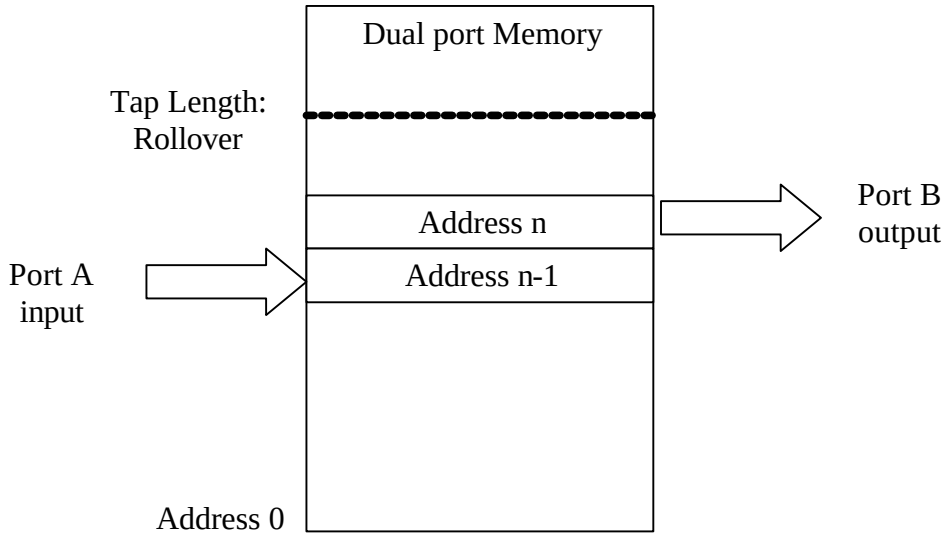

Figure 1: Tap Delay structure

The resources used are as follows:

Slices: 21
Block RAM: 1

II – DiffFilt

This module implements the differentiation filter. It simply does a 2's complement, which is no less than a modified adder (with the first carry set to 1). We must note that performing X_n-X_{n-k} correspond to doing X_n + (not X_{n-k}) + 1. We must however note that the addition is one bit wider since we have to extend the sign to accommodate for the 2's complement sign extension. The speed is around 380MHz.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
Xn(11 to 0)	The sample Xn in 2's complement representation
Xn_k(11 to 0)	The sample Xn-k in 2's complement representation
Yn(12 to 0)	Output of the filter in 2's complement representation

Table 2: Differentiation filter I/Os

The resources used are as follows:

Slices: 13
Block RAM: 0

III – GauFilt1

This module implements a first stage (13 bit input) three point Gaussian filter. It implements the following equation:
$(X_n + X_{n-2}) + X_{n-1}<<1$
The speed is around 220MHz.
The following graph describe the internal architecture:

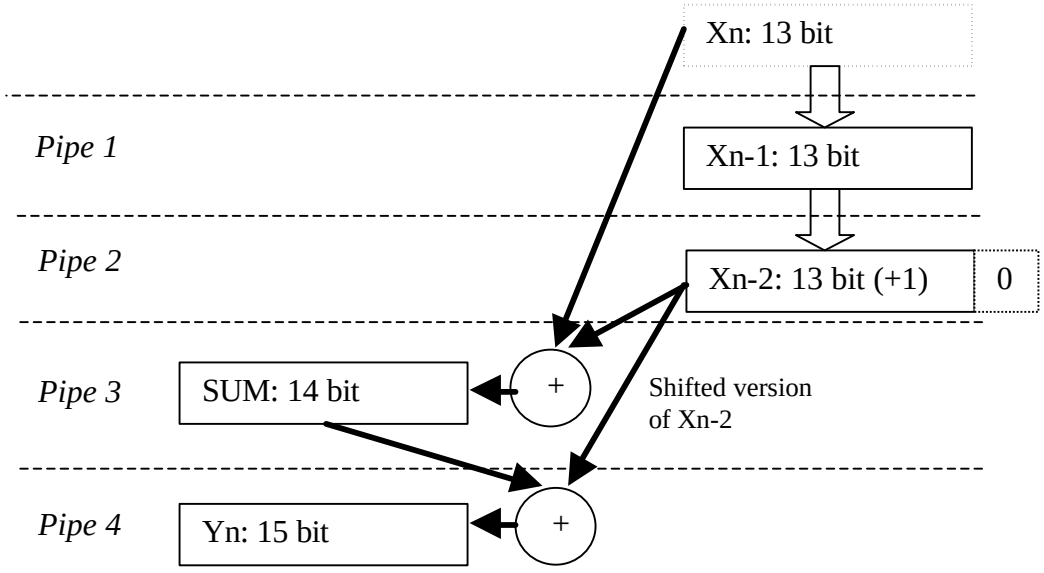

Figure 2: Gaussian Filter first stage architecture

We must note that we loose 1 pipeline delay by doing the addition between ($X_n + X_{n-2}$) but that it saves resources in terms of adder length and intermediary storage register. We do not really care about the delay in the result since it is only one clock cycle delay and we have to take the entire pipeline into account anyways. In addition, if we are required to pipeline the ripple carry adder (look-ahead adder does not seem appropriate due to the extra space required for the small speed gained). In addition since we are operating in 2's complement representation, we cannot use the Carry out as the MSB but we need to extend the operands for a proper MSB. This is due to the fact that in 2's complement the carry out must be combined to the eventual signed bit to have a meaning.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
Xn(12 to 0)	The sample Xn in 2's complement representation
Yn(14 to 0)	Output of the filter in 2's complement representation

Table 3: Gaussian filter stage 1 I/Os

The resources used are as follows:

Slices: 40
Block RAM: 0

IV – GauFilt2

This module implements a second stage (15 bit input) three point Gaussian filter. It implements the following equation:
$(X_n + X_{n-2}) + X_{n-1}<<1$
The speed is around 197MHz (slower than the previous stage). The architecture is the same as the one used for *GauFilt1*.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
Xn(14 to 0)	The sample Xn in 2's complement representation
Yn(16 to 0)	Output of the filter in 2's complement representation

Table 4: Gaussian filter stage 2 I/Os

The resources used are as follows:

Slices: 48
Block RAM: 0

V – LED

This module combine all the blocks needed to generate the Leading Edge Discriminator timing. It is composed of some sub-modules that are not reused anywhere else.

V.1 – VTHCross

This sub-module computes the crossing of a rising edge above a threshold or the crossing of a falling edge under a negative threshold. It is using only the carry of a subtraction and therefore only the carry generation is implemented. For speed reason since we also need to generate a signed version of the threshold a direct implementation using Virtex blocks has been used. The following flow chart shows the implementation:

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
LOADVTH	A synchronous pulse used to load the threshold value
VTH(14 to 0)	The unsigned representation of the threshold
INPUT(15 to 0)	Input sample
SIGN	Indicate if the crossing is positive or negative
CROSSING	The result. Set to one if threshold crossed above VTH or below -VTH

Table 5: Threshold crossing I/Os

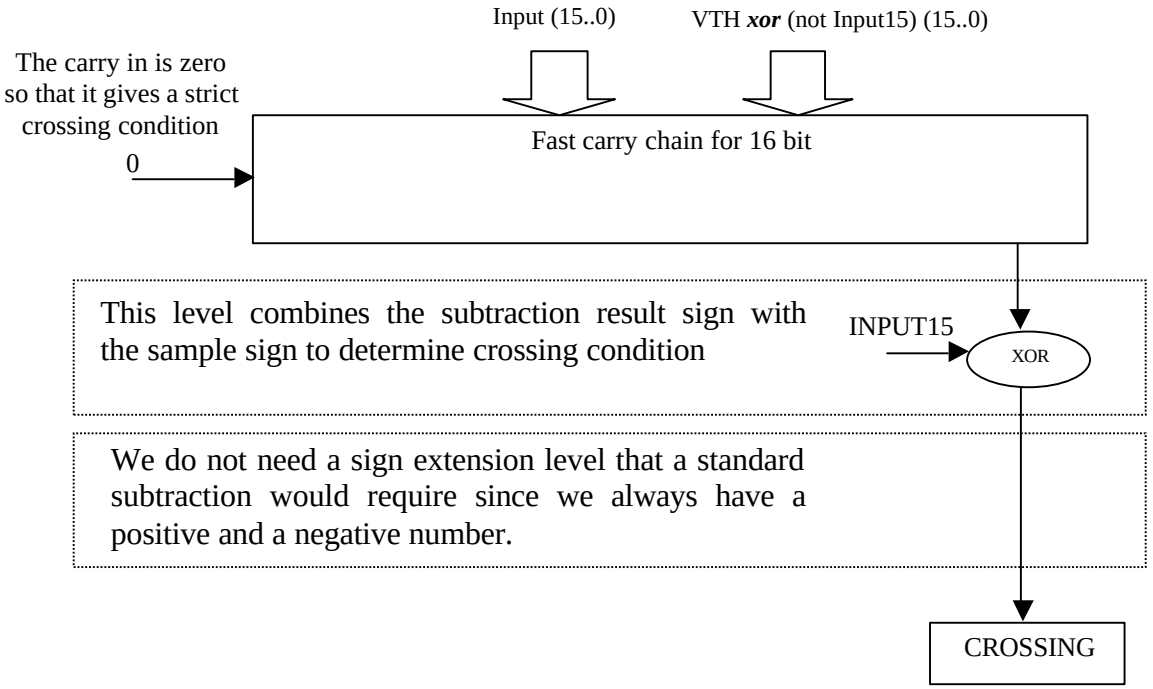

Figure 3: Threshold crossing structure

V.2 – NoiseTimer

This sub-module implements the Noise Timer used as a programmable dead-time. It uses a simple counter that starts counting on a START signal and stops when the programmable length is reached.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
START	The synchronous start pulse (should be valid only one clock cycle)
CLEAR	A synchronous line disabling the counter and bringing any current noise window to ready.
LOADNoise	The synchronous load pulse for storing the window length
WinNoise(6 to 0)	Noise window length
DISABLED	The disable line indicating that the system is still under the noise window condition.

Table 6: Noise Timer I/Os

V.3 –VTHProcess

This sub-module implements the final processing required for the Leading edge Discriminator by combining the different windows with the crossing line to generate the timestamp pulse.

It is implemented as a state machine with two states as described below.

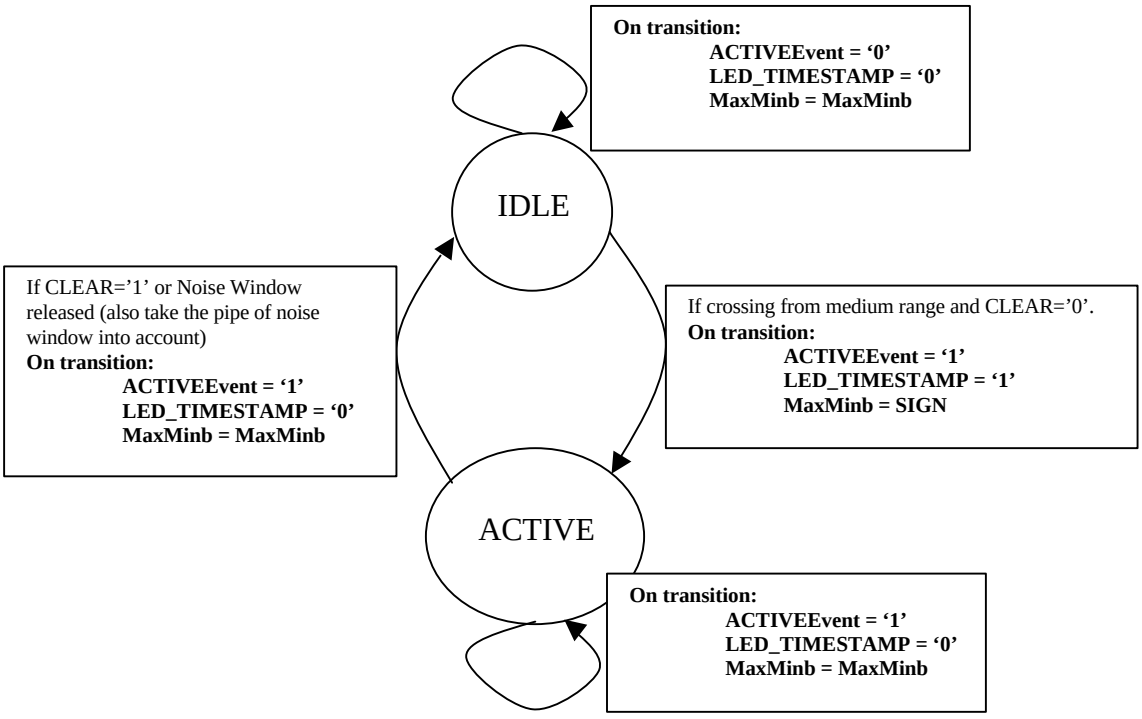

Figure 4: VTHProcess Structure

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
CLEAR	A synchronous line forcing the state into idle
DISABLED	The synchronous signal coming from the Noise Timer module
CROSSING	A synchronous signal coming from VTHCrossing indicating a trigger condition has been detected
SIGN	Input signal indicating the sign of the crossing
MaxMinb	A synchronous signal indicating the sign of the previous crossing.
ACTIVEEvent	Output signal indicating an event is being processed
LED_TIMESTAMP	A synchronous pulse indicating when the timestamp should be recorded. It is also used for starting the Noise Timer

Table 7: VTHProcess I/Os

V.4 –LED

This is the main module. It integrates all the previous modules and various filter modules. The speed is around 214MHz.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
CLEAR	A synchronous line forcing the state into idle and RESETEvent high
Xn (11..0)	The current sample Xn in 2's complement representation
Xn_k (11..0)	The sample delayed by k clock cycles in 2's complement representation
LOADVTH	A synchronous pulse indicting that the VTH input has to be latch into the internal threshold register.
VTH (14..0)	The unsigned value of the threshold. It is one bit less since the MSB is 0 when converted to 2's complement.
LOADNoise	A synchronous pulse indicting that the noise window length has to be latch into the internal register.
WinNoise (6..0)	The noise window length in unsigned representation
ACTIVEEvent	Output status line indicating that an event is being processed.
MaxMinb	A synchronous signal indicating the sign of the previous crossing.
LED_TIMESTAMP	A synchronous pulse indicating when the timestamp should be recorded. It is also used for starting the Noise Timer

Table 8: LED I/Os

The following flow chart describes the full LED algorithm implementation using the previous blocks. For indication the bus sizes have been added.

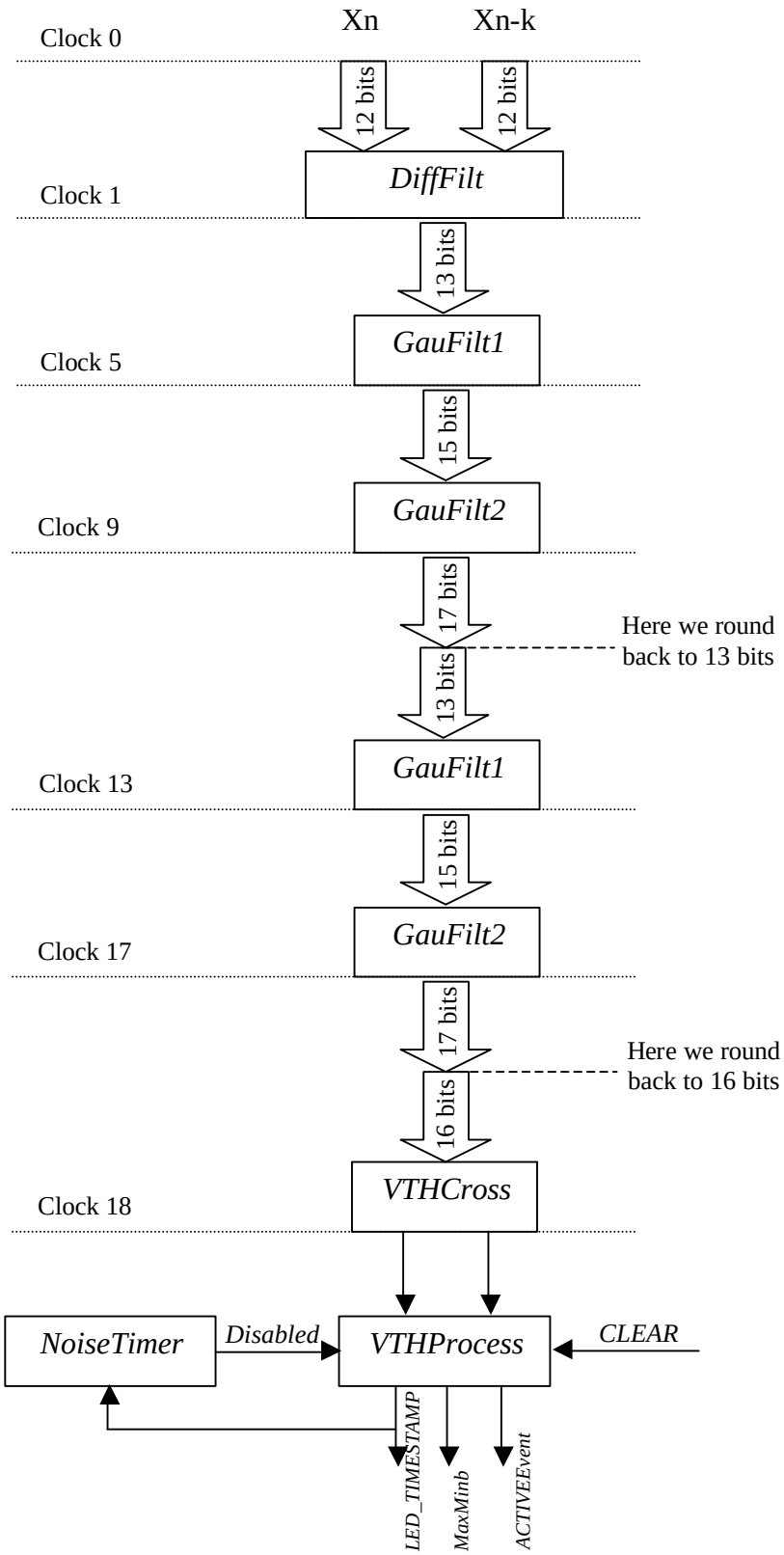

Figure 6: Leading Edge Discriminator (LED) Flow Chart

The resources used are as follows:

Slices: 120
Block RAM: 0

VI – CFD

This module combines all the blocks needed to generate the Constant Fraction Discriminator timing. It is composed of some sub-modules that are not reused anywhere else. It is composed of some sub-modules that are not reused anywhere else.

VI.1 – MULTMinusa

This sub-module implements a multiplication by -1, -2, -4 and -8. It is basically a four-to-one multiplexer depending on the *'a'* value coded with 2 bits. The following table gives the relation between the *'a'* value and the multiplication performed.

***'a'* Value**	**Multiplication**
00	-1
01	-2
10	-4
11	-8

Table 9: Multiplication equivalence

We must note that it adds 1 to the result so that we actually have:

OUTPUT = -a.INPUT + 1.

It is using the dedicated internal multiplexer 4-to-1 between two LUT in a slice. It runs around 366MHz.

Signal Name	**Description**
RESET	Asynchronous reset. It is active high
CLK	Clock.
LOADa	The synchronous load pulse for storing the 'a' value
a(1..0)	The value of 'a' as described in table 9
INPUT(15 to 0)	The input value in 2's complement representation
OUTPUT(18..0)	The output value in 2's complement representation.

Table 10: MULTMinusa I/Os

The resources used are as follows:

Slices: 17
Block RAM: 0

VI.2 – Magnitude

This sub-module implements a magnitude discriminator (similar to a leading edge crossing algorithm). However, the threshold used here does not span the full range since we only need small thresholds. It also allows reducing the logic needed since the carry can be propagated twice as fast for the upper bits. Since the LUT inside the Xilinx have four inputs, it is possible to compute two carry at once when the threshold is sign extended.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
LOADThreshold	The synchronous load pulse for storing the threshold value
Threshold(4..0)	The threshold value in unsigned representation
INPUT(11 to 0)	The input value in 2's complement representation
ENABLE	A synchronous pulse indicating that the magnitude of the input signal is above the threshold.

Table 11: Magnitude I/Os

The following table and figure tries to summarize the two carries generation at once through the fast carry chain.

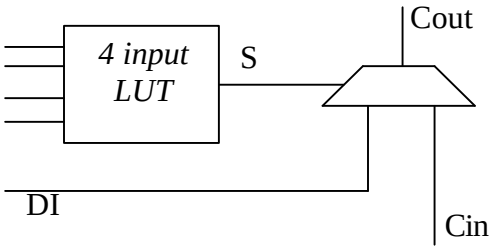

Figure 7: Fast Carry logic VirtexII Xilinx block

Cin	INPUT(i)	not SIGN	*Local carry*	INPUT(i+1)	Cout	Comment
0	0	0	0	0	0	
1	0	0	0	0	0	
0	1	0	0	0	0	
1	1	0	1	0	0	
0	0	1	0	0	0	Spot taken by the Cin decode MUX
1	0	1	1	0	1	Needed to decode Cin
0	1	1	1	0	1	Only represented by SIGN outside the Cin Decode MUX
1	1	1	1	0	1	
0	0	0	0	1	0	
1	0	0	0	1	0	
0	1	0	0	1	0	Spot taken by the Cin decode MUX
1	1	0	1	1	1	Needed to decode Cin
0	0	1	0	1	1	Only represented by "not SIGN" outside the Cin Decode MUX
1	0	1	1	1	1	
0	1	1	1	1	1	
1	1	1	1	1	1	

Table 12: Two carry fast propagation logic

We can therefore have the following equations:

- DI is “not INPUT(11)” for representing “not SIGN”
- S is “((not INPUT(11)) and (not INPUT(i)) and (not INPUT(i+1))) or (INPUT(11) and INPUT(i) and INPUT(i+1))” for decoding Cin

The resources used are as follows:

Slices: 8
Block RAM: 0

VI.3 – TapDelayLoc

This sub-module implements a programmable length free running Tap Delay using local distributed memory. The maximum length for this delay is 64 taps. It is using the same architecture described in figure 1.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
LOAD	This synchronous signal indicates when we want to load a new delay length. It resets the status line to zero as well as the internal counters.
STATUS	Goes high when the Tap delay is filled up and ready to have valid input output relation for filtering. It goes low each time the LOAD signal is asserted and goes back up when ready for operation.
LENGTH(5 to 0)	The bus giving the programmable length to be loaded.
DATAin(15 to 0)	Input of the Tap delay X_n
DATAout(15 to 0)	Output of the Tap delay Xn-L

Table 13: Tap Delay Local I/Os

VI.4 – CFDProcess

This sub-module implements the CFD processing. It detects a zero crossing (both directions are detected to accommodate both polarities). It is implemented as a state machine with two states as described below.
We must note that the data is truncated from 20bits to 16 bits at this level.

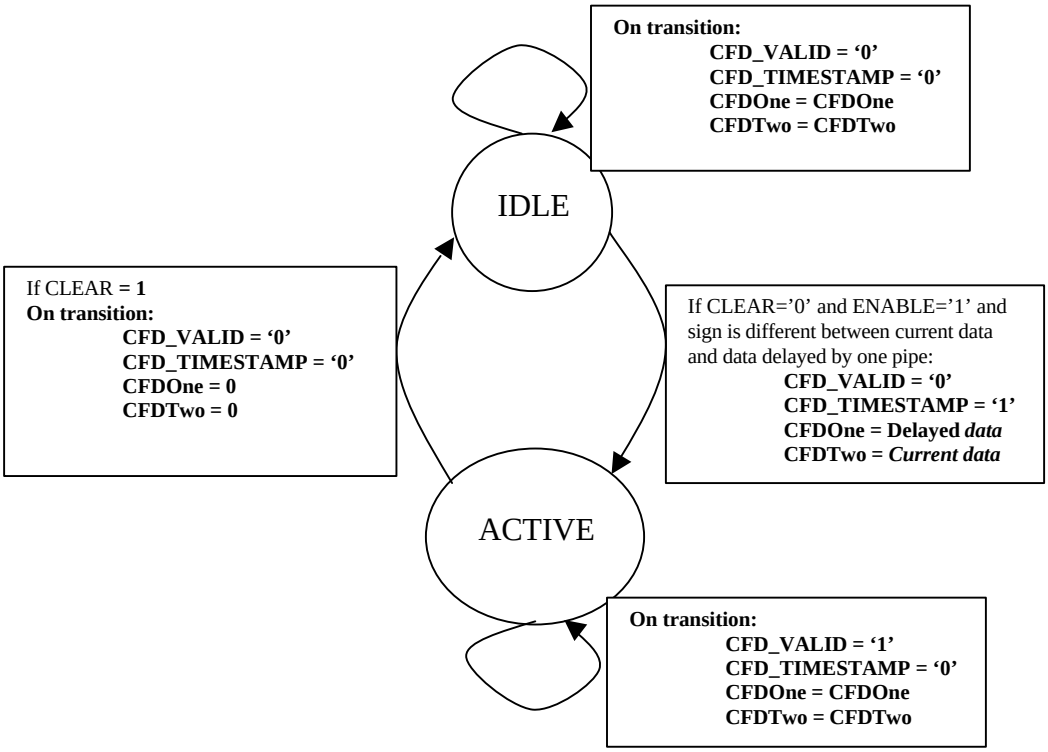

Figure 8: CFDProcess Structure

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
CLEAR	A synchronous pulse indicating that the event has been processed or that the system is disabled and that we can reset all the values corresponding to it
ENABLE	A synchronous pulse indicating that the signal magnitude is above a given threshold.
CFDData(19 to 0)	The input value in 2's complement representation.
CFDOne(15 to 0)	First point of the CFD in 2's complement.
CFDTwo(15 to 0)	Second point of the CFD in 2's complement.
CFD_VALID	A line indicating that the CFD points are valid for the current event
CFD_TIMESTAMP	A synchronous pulse indicating when the timestamp should be recorded.

Table 14: CFD Process I/Os

VI.5 –CFD

This is the main module. It integrates all the previous modules and various filter modules. The speed is around 161MHz (the speed estimation is flawed here because the routing includes the distributed RAM and it will change when the code is compiled for the final design/The speed can be improved if we pipe the RAM output data).

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
CLEAR	A synchronous pulse indicating that the event has been processed and that we can reset all the values corresponding to it
LOADThreshold	The synchronous load pulse for storing the threshold value
Threshold(4..0)	The threshold value in unsigned representation
LOADa	The synchronous load pulse for storing the 'a' value
a(1..0)	The value of 'a' as described in table 9
LOADTAPLength	This synchronous signal indicates when we want to load a new delay length. It resets the status line to zero as well as the internal counters.
TAPLength(5 to 0)	The bus giving the programmable length to be loaded.
Xn(11 to 0)	The sample Xn in 2's complement representation
Xn_k(11 to 0)	The sample Xn-k in 2's complement representation
CFDOne(15 to 0)	First point of the CFD in 2's complement.
CFDTwo(15 to 0)	Second point of the CFD in 2's complement.
TAP_STATUS	Goes high when the Tap delay is filled up and ready to have valid input output relation for filtering. It goes low each time the LOAD signal is asserted and goes back up when ready for operation.
CFD_VALID	A line indicating that the CFD points are valid for the current event
CFD_TIMESTAMP	A synchronous pulse indicating when the timestamp should be recorded.

Table 15: CFD I/Os

The resources used are as follows:

Slices: 228 (distributed RAM included)
Block RAM: 0

The graph given in figure 9 shows the CFD algorithm implementation using the previous blocks.

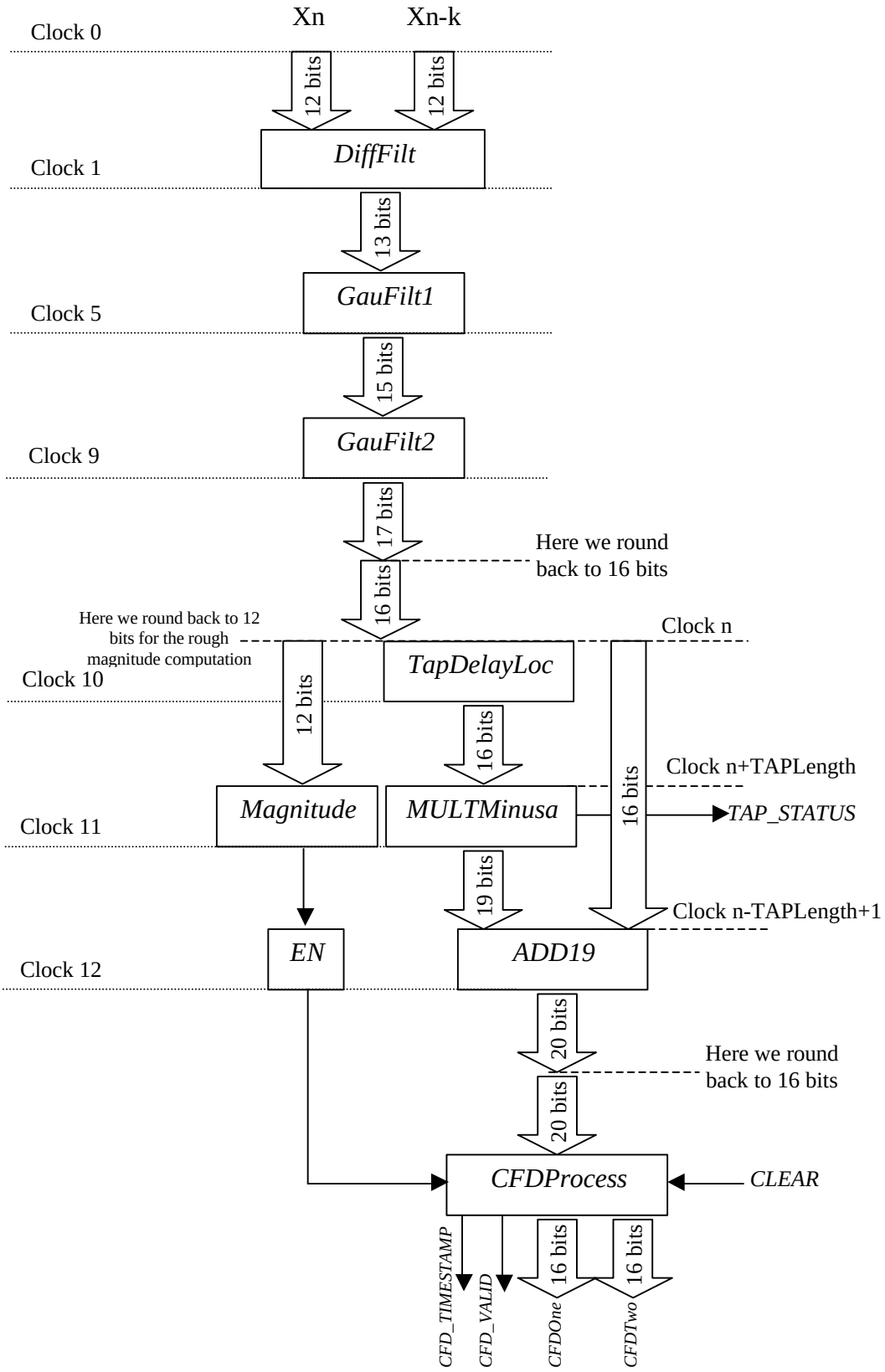

Figure 9: CFD algorithm

VII – TrapzFil

This module implements the computational part of a trapezoidal filter (the tap delay are external to this module). It simply implements the following equation:

$Y_n = Y_{n-1} + ((X_n + X_{n-2m-k}) - (X_{n-m} + X_{n-m-k}))$

It runs around 199MHz.
The following graph describes the internal architecture:

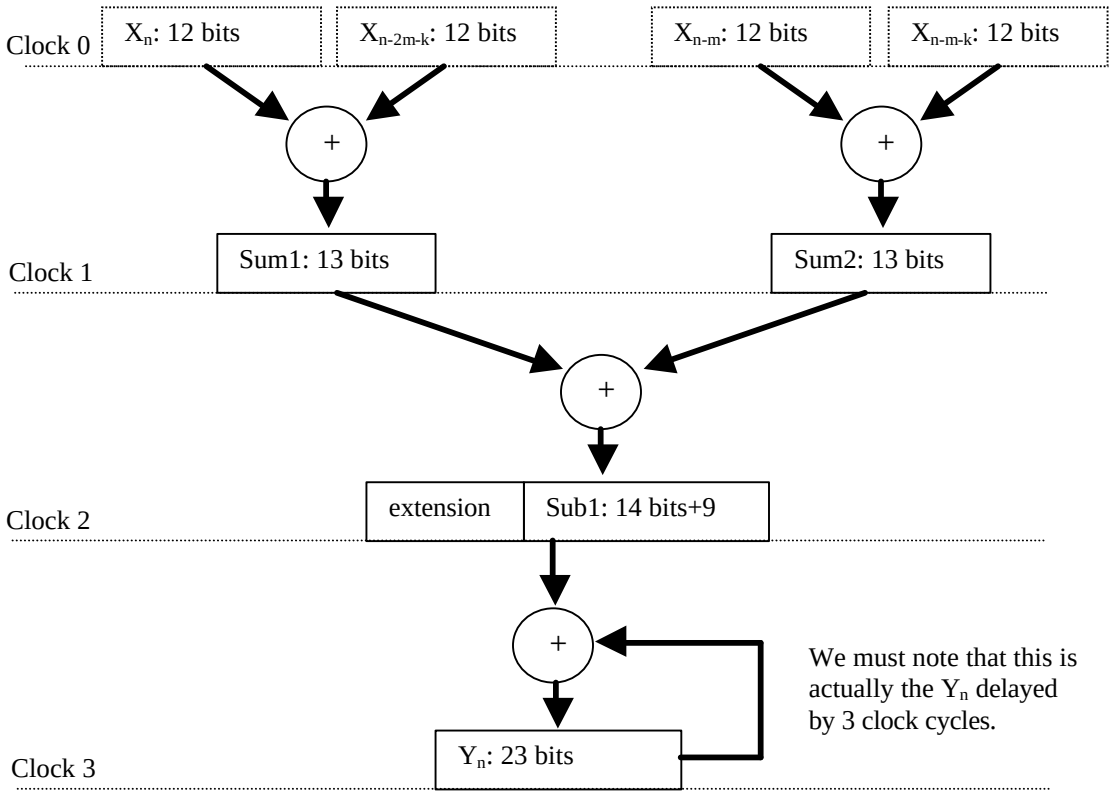

Figure 10: Trapezoidal filter architecture

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
Xn(11 to 0)	The sample X_n in 2's complement representation
Xn_m(11 to 0)	The sample X_{n-m} in 2's complement representation
Xn_m_k(11 to 0)	The sample X_{n-m-k} in 2's complement representation
Xn_2m_k(11 to 0)	The sample X_{n-2m-k} in 2's complement representation
Yn(22 to 0)	The result Y_n in 2's complement representation

Table 16: TrapzFil I/Os

The resources used are as follows:
Slices: 54
Block RAM: 0

VIII – MSearch

This module implements the extremum search given a minimum/maximum flag. It uses several sub-modules described in the following sections.

VIII.1 – COMPARE

This submodule finds the sign of the difference between the two inputs and generates an update signal if the sign line validates the result.
As the sign computation is performed over 23 bits and must be used to multiplex two values, we must be as quick as possible to compute it.

Version 1: generate the update signal using 12 slices.
This version uses the simplest carry chain with a cascade of 23 MUXCY:

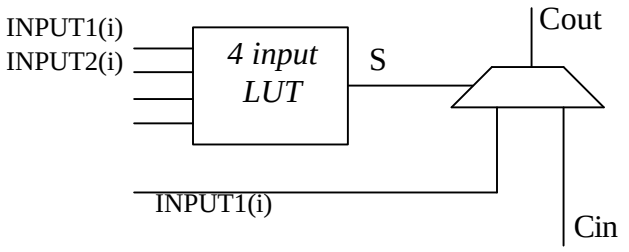

Figure 11: simple COMPARE architecture

Version 2: generate the update signal using 18 slices.
This version uses two LUTs and uses only half the carry chain length since we compute the carry every other one.

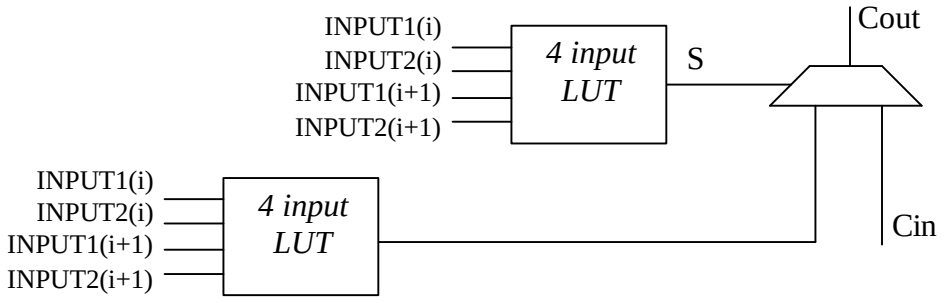

Figure 12: parallel COMPARE architecture

Signal Name	Description
INPUT1(22 to 0)	The new sample in 2's complement
INPUT2(22 to 0)	The sample to compare to in 2's complement
SIGN	A synchronous signal indicating a min or max search
UPDATE	A synchronous pulse indicating that the compare value has to be updated with the new sample.

Table 17: COMPARE I/Os

VIII.2 – MSearch

This module integrates the extremum detection and storage.

With version 2: 153MHz operation
With version 1: 179MHz operation

It seems that even though version 2 looks more promising on paper, the LUT to MUXCY routing takes longer than 11 MUXCY to MUXCy propagation. Since it goes fast enough the simplest implementation was implemented:

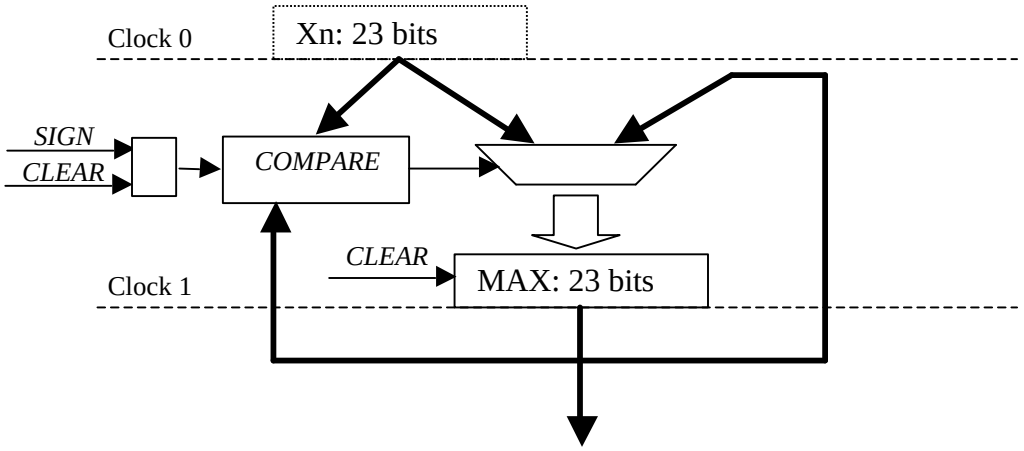

Figure 13: MSearch Structure

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
Xn(22 to 0)	The sample to compare to the current extremum in 2's complement
CLEAR	A synchronous pulse indicating that the event has been processed and that we can reset all the values corresponding to it. It also indicates that a new sign has to be updated for the next tracking.
SIGN	A synchronous signal indicating a min or max search at the time of a CLEAR (for next extremum tracking).
MAX(22 to 0)	The current extremum in 2's complement

Table 18: MSearch I/Os

The resources used are as follows:
Slices: 24
Block RAM: 0

IX – ENERGY

This module integrates the trapezoidal filter with the extremum search in order to create the energy value. It runs at about 180MHz (The routing to the COMPARE block is a little bit more efficient when used in the middle of a design).
The following figure summarizes the overall pipeline for the energy computation:

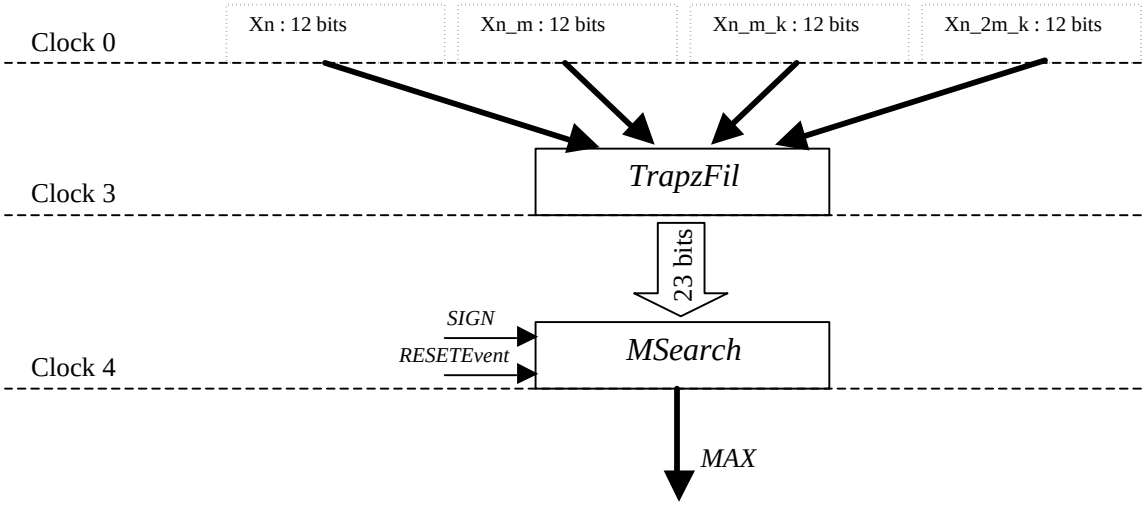

Figure 14: Energy computation structure

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
Xn(11 to 0)	The sample X_n in 2's complement representation
Xn_m(11 to 0)	The sample X_{n-m} in 2's complement representation
Xn_m_k(11 to 0)	The sample X_{n-m-k} in 2's complement representation
Xn_2m_k(11 to 0)	The sample X_{n-2m-k} in 2's complement representation
CLEAR	A synchronous pulse indicating that the event has been processed and that we can reset all the values corresponding to it.
SIGN	A synchronous signal indicating a min or max search (valid when CLEAR is active)
MAX(22 to 0)	The current extremum in 2's complement

Table 19: Energy I/Os

The resources used are as follows:
Slices: 79
Block RAM: 0

X – TIMER

This module is the main timer with 48bit precision (up to 78 hours at 100MHz). It has a main synch signal so that all channels can have the same reference. The circuit operates around 197MHz.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
SYNCH	A synchronous pulse use to reset the timer (used to synchronize all channels)
TIME(47 to 0)	The current time

Table 20: TIMER I/Os

The resources used are as follows:

Slices: 48 (the average number will be reduced when combined with other logic).
Block RAM: 0

XI – ProcCORE

This module is the main processing core for one channel. It handles all events and is responsible for creating the packet.
The available status lines from the processing units are:

- TAPDelay1_STATUS
- TAPDelay2_STATUS
- TAPDelay3_STATUS
- TAPDelay4_STATUS
- TAPDelayLoc1_STATUS
- ACTIVEEvent
- CFD_VALID

It must generate the following control line for the processing units:

- CLEAR
- LOADTap1
- LOADTap2
- LOADTap3
- LOADTap4
- LOADTapLoc1
- **LOAD_LEDTH**
- LOAD_LEDTimer
- **LOAD_CFDTH**
- LOAD_CFDa

Note: for all the 'LOADxxx' lines, the data must also be presented at the right time.

It receives the following data oriented line/busses:

- LED_TIMESTAMP
- LED_SIGN
- CFD_TIMESTAMP
- CFD_ONE
- CFD_TWO
- MAX

It must accommodate the following external parameters:

- PileUp Window (10bits for 10us)
- External Validation Window (11bits for 20us after)
- **External Sliding length (11 bits for 20us)**
- **Raw Data Sliding length (11 bits for 20us)**
- **Raw Data window length (10 bits for up to 10us)**
- **LED Threshold (16 bits)**
- **CFD Threshold (5 bits)**
- CFD fraction (2 bits)
- BoardID (13 bits: three are generated internally for the 8 channels)
- Trigger mode (Internal, External, Internal with validation)
- **Start/Stop acquisition**
- **Debug mode**
- External trigger
- Timer Synch.
- External FIFO busy

The bold items in the previous lists may be different (at the parameter level) for all the channels. This increases the number of VME-based configuration values. The programming register cannot be read back as it would involve too much multiplexing to send it back out through one bus especially since those value are stored close to their processing unit for speed reasons.
The following configuration register are used for setting up the system:

Register Name	Address (Word aligned)	Description
Board ID	0x00	Value is given by the switch configuration. Only 8 bits will be used. *Read-Only*
Programming Done	0x01	Bit 7 to 0: Gives the current programming status on all channels Bit 8: Gives the programming status on the debug module Bit 9: Gives the programming status on the DACs Bit 10: and FIFO 0 EF flag. Bit 11: and FIFO 1 EF flag.

Register Name	Address (Word aligned)	Description
		Bit 12: and FIFO 0 PAE flag. Bit 13: and FIFO 0 HF flag. Bit 14: and FIFO 0 PAF flag. Bit 15: and FIFO 0 FF flag **and** FIFO 1 FF flag. *Read-Only*
External Window	0x02	External validation window length (only bit 10 to 0 are used) Value at reset is 0x07FF *Write-only*
Pileup Window	0x03	Pileup-window length (only bit 10 to 0 are used) Value at reset 0x0400 (10us) *Write-only*
Noise Window	0x04	Noise window length (only bit 6 to 0 are used) Value at reset 0x0040 (640ns) *Write-only*
External trigger sliding length	0x05	Length before we read the energy when we operate in external trigger. (only bit 10 to 0 are used) Value at reset 0x01C2 (4.5us) *Write-only*
Collection time	0x06	Collection time maximum length (length of the flat top in the trapezoid). Only bits 8 to 0 are used. Value at reset 0x01C2 (4.5us) *Write-only*
Integration time	0x07	Integration time length (length of one side of the trapezoid). Only bits 8 to 0 are used. Value at reset 0x01C2 (4.5us) *Write-only*
Control/Status: Channel 0 Channel 1 Channel 2 Channel 3 Channel 4 Channel 5 Channel 6 Channel 7	 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F	**This register address is the only one that is read/write.** Bit 15: Pre-buffer Ready flag Bit 14-12: Unused for now. Bit 11-10: Polarity validation. Value at reset 11. (*Read/Write).* 01: only positive trigger considered 10: only negative trigger considered 11: both triggers considered 00: no trigger considered (the LED still fires externally though, as opposed to START/STOP)

Register Name	Address (Word aligned)	Description
		Bit 9: Tap delay 4 set-up (*Read-only)*. If set to 0, the tap delay is filling-up and no event can be acquired. Bit 8: Tap delay 3 set-up (*Read-only)*. If set to 0, the tap delay is filling-up and no event can be acquired. Bit 7: Tap delay 2 set-up (*Read-only)*. If set to 0, the tap delay is filling-up and no event can be acquired. Bit 6: Tap delay 1 set-up (*Read-only)*. If set to 0, the tap delay is filling-up and no event can be acquired. Bit 5: CFD Tap delay setup. (*Read-only)*. If set to 0, the tap delay is filling-up and no event can be acquired. Bit 4-3: Trigger mode. We have the following configuration: 00/11: Internal mode 01: External mode 10: Validation Value a reset: 00. (*Read/Write)* Bit 2: Pile-up drop-out enable. When this bit is set to one, the channel drops any event that occurs in a pileup window else it just notifies pileup through a flag in the event header. Value at reset 1. *Read/Write.* Bit 1: Debug Mode. This bit when set to 1 puts the channel in debug mode operation using internal data. Value at reset 0. (*Read/Write)* Bit 0: START/STOP. This bit must be set to one for the channel to operate. Value at reset is 0. (*Read/write)*
LED Threshold: Channel 0 Channel 1 Channel 2 Channel 3 Channel 4 Channel 5 Channel 6 Channel 7	 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17	This is the LED threshold. Only bits 15 to 0 are used. This is an unsigned value and the three lower bits are extra precision bit. (The dot is between bit 3 and 2). The value at reset is 0x7FFF (full range so that no event should trigger). It is internally converted to a signed value depending on how the sign of the current sample is. *Write-only.*
CFD parameters: Channel 0 Channel 1 Channel 2 Channel 3 Channel 4 Channel 5 Channel 6 Channel 7	 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F	Bit 12-7: CFD delay. Value at reset is 0x3F giving a delay of 630ns. *Write-only* Bit 6-5: CFD fraction. It is a value indicating what fraction is used (see table 9 for values). Value at reset is "00" giving a fraction of 0.5. *Write-only* Bit 4-0: CFD threshold. It is an unsigned value of the CFD threshold. The dot is after bit 0. Value at reset is 0x10 (160kev). *Write-only*

Register Name	Address (Word aligned)	Description
Raw Data sliding length: Channel 0 Channel 1 Channel 2 Channel 3 Channel 4 Channel 5 Channel 6 Channel 7	 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27	Delay before we start retrieving raw data. Only bits 10 to 0 are used. Value at reset is 0x01C2 (4.5us). *Write-only*
Raw Data window length: Channel 0 Channel 1 Channel 2 Channel 3 Channel 4 Channel 5 Channel 6 Channel 7	0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F	Length of raw data retrieved. Only bits 9 to 0 are used. Value at reset is 0x32 (500ns). *Write-only*
Debug data buffer address	0x30	Used by debug module (see section XIII)
Debug data buffer data	0x31	Used by debug module (see section XIII)

Table 21: Configuration Registers

We therefore need 6 bit of address for register configuration.
The event data is organized as a packet that contains a header and the raw data. The packet structure is described below in table 22:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Board ID																Packet length (header included)															
LED/external Timestamp bit 0-15																LED/external Timestamp bit 16-31															
LED/external Timestamp bit 32 to 47																Energy bit 0-15															
Energy bit 16-22							x	x	x	x	x	Sx	E	C	P	CFD Timestamp bit 0-15															
CFD Timestamp bit 16-31																CFD Timestamp bit 32-47															
CFD point 1																CFD point 2															
Raw data point 0												x	x	x	x	Raw data point 1												x	x	x	x
Raw data point 2																Raw data point 3												x	x	x	x
. . . .																. . . .															

Table 22: Event packet structure

Note: P is for the pileup flag indicating that the energy is corrupted
C is for CFD valid indicating that a CFD crossing occurred.
E is for external trigger flag (the timestamp is external time and not LED, CFD is not valid)
S is for the sign of the LED crossing. (1 negative)
x *is for 'not used'*

This module contains several sub-modules described in the following sub-sections:

XI.1 – MUX16_16to1

This is a multiplexer 16bits wide, 16 input to 1 output with enable. It is using built-in features of the VirtexII FPGA so that it can operate at full speed (around 141MHz) with the less resource requirement (64 slices). The following diagram shows how the multiplexer is implemented for one bit

Signal Name	**Description**
RESET	Asynchronous reset. It is active high
CLK	Clock.
ENABLE	The multiplexer outputs non-zero value when set to 1
S(3 to 0)	The selection bus
INPUT0(15 to 0)	The input selected by "0000"
INPUT1(15 to 0)	The input selected by "0001"
INPUT2(15 to 0)	The input selected by "0010"
INPUT3(15 to 0)	The input selected by "0011"
INPUT4(15 to 0)	The input selected by "0100"
INPUT5(15 to 0)	The input selected by "0101"
INPUT6(15 to 0)	The input selected by "0110"
INPUT7(15 to 0)	The input selected by "0111"
INPUT8(15 to 0)	The input selected by "1000"
INPUT9(15 to 0)	The input selected by "1001"
INPUT10(15 to 0)	The input selected by "1010"
INPUT11(15 to 0)	The input selected by "1011"
INPUT12(15 to 0)	The input selected by "1100"
INPUT13(15 to 0)	The input selected by "1101"
INPUT14(15 to 0)	The input selected by "1110"
INPUT15(15 to 0)	The input selected by "1111"
OUTPUT(15 to 0)	The output

Table 23: MUX16_16to1 I/Os

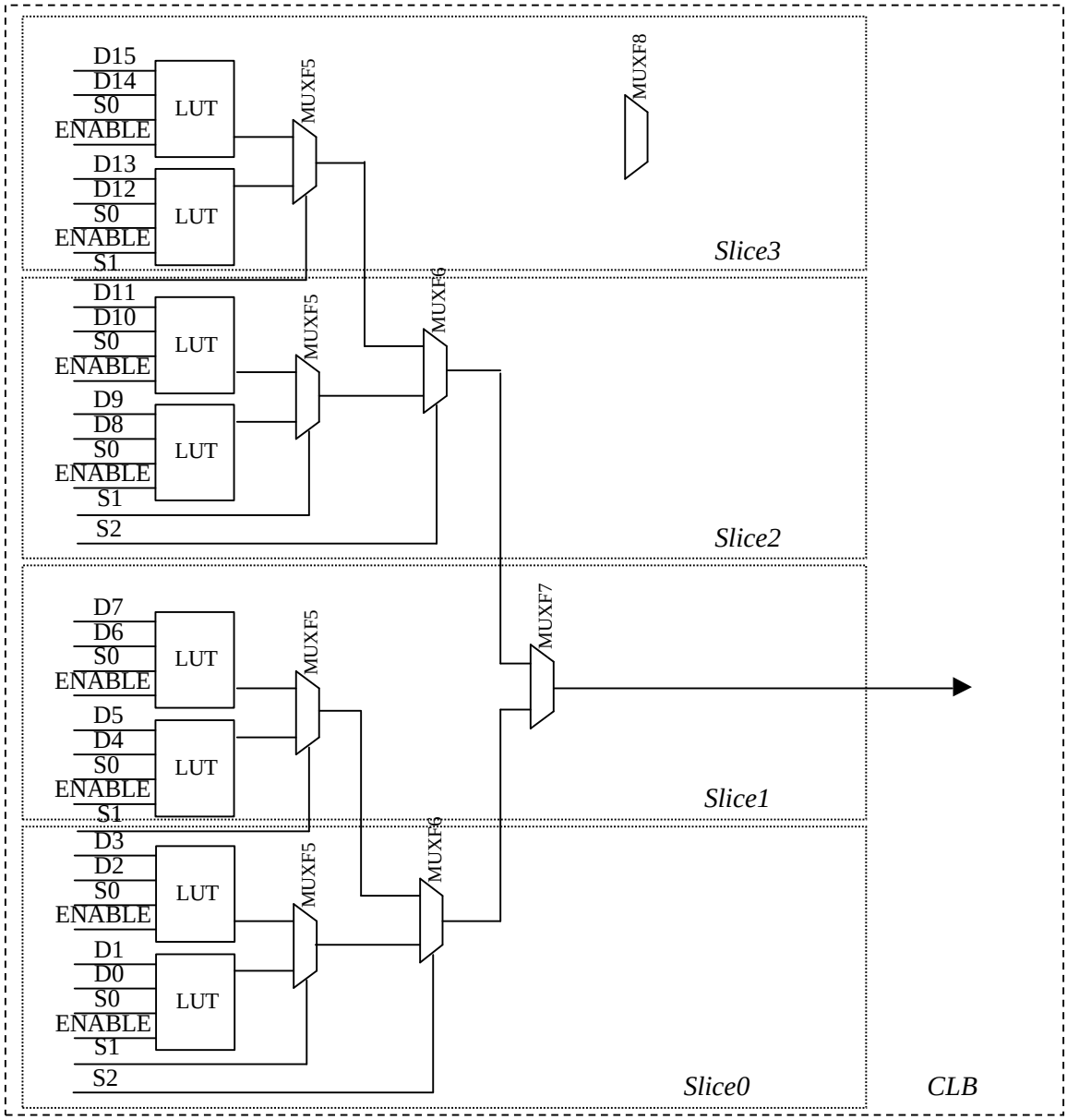

Figure 15: Mux 16 to 1 structure

XI.2 – WaitCounter

This module implements three different counters:

- a counter that can start on two different window-lengths. It accommodates the internal delay window and the external delay window. (We then save one counter by using only one). It is possible to combine them, as we use either one or the other. It is gated with a dead time window computed from the sliding and the raw data length.
- a counter with enable used to generate the pre-buffer memory address as well as generate the proper number of raw data point retrieval.
- a counter used to wait until the raw data is ready.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
LOADValidationWait	The external validation length load pulse
ValidationWait(10 to 0)	The external validation length
LOADExternalWait	The external trigger sliding length load pulse
ExternalWait(10 to 0)	The external trigger sliding length
LOADm	The integration length load pulse
m(8 to 0)	The integration length
LOADk	The collection length load pulse
k(8 to 0)	The collection length
LOADRawLength	The raw data length load pulse
RawLength(9 to 0)	The raw data length
LOADSlidingWait	The raw data sliding length load pulse
SlidingWait(10 to 0)	The raw data sliding length
STARTExternalWait	The synchronous start pulse for external waiting window
STARTInternalWait	The synchronous start pulse for internal waiting window
STARTSlidingWait	The synchronous start pulse for sliding raw data window
STARTAddWait	The synchronous start pulse for address/raw length window
AddEnable	The enable line for the address increment
Address(9 to 0)	The Address where the data has to be written in the packet
RawAdd	A line that indicates when the raw data has to be sent.
RawSlidingDone	A pulse that indicates when the sliding raw data length is done
ExtValidation	A line that indicates when we are waiting for external validation
MinDeadtimeDone	Indicates that the minimum dead-time has been reached
ComputedDone	A synchronous pulse that indicates the end of the energy computation window

Table 24: WaitCounter I/Os

XI.3 – StateMachine

This module implements the main channel housekeeping state-machines. It has two parallel state-machines
The system works around two state-machines in order to decrease the dead time. The time flow chart described in figure 16 (next page describe the use of resources and in particular the pre-buffer memory). Also is shown the time when the data can be read-out and sent to the outside world.

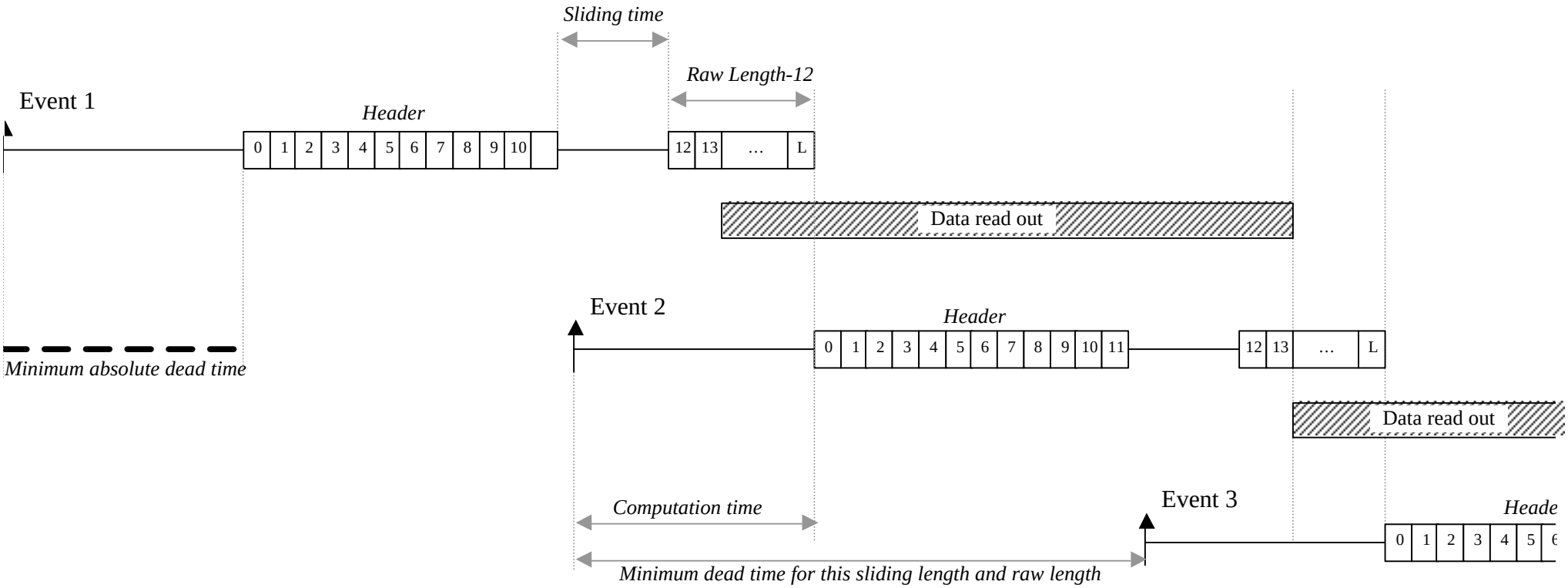

Figure 16: Pre-buffer use by the state machine and the outside multiplexer

The minimum dead-time for a given configuration can be computed as follows:

Max(computation[internal trigger or external trigger] , Raw length + sliding time)

The dead-time is increased if the pre-buffer is not being emptied when a new event occurs.

XI.3.a – TimerMachine

One state machine is responsible for getting the timing and allowing for programming values. The other one is responsible for the packet creation and data retrieval.

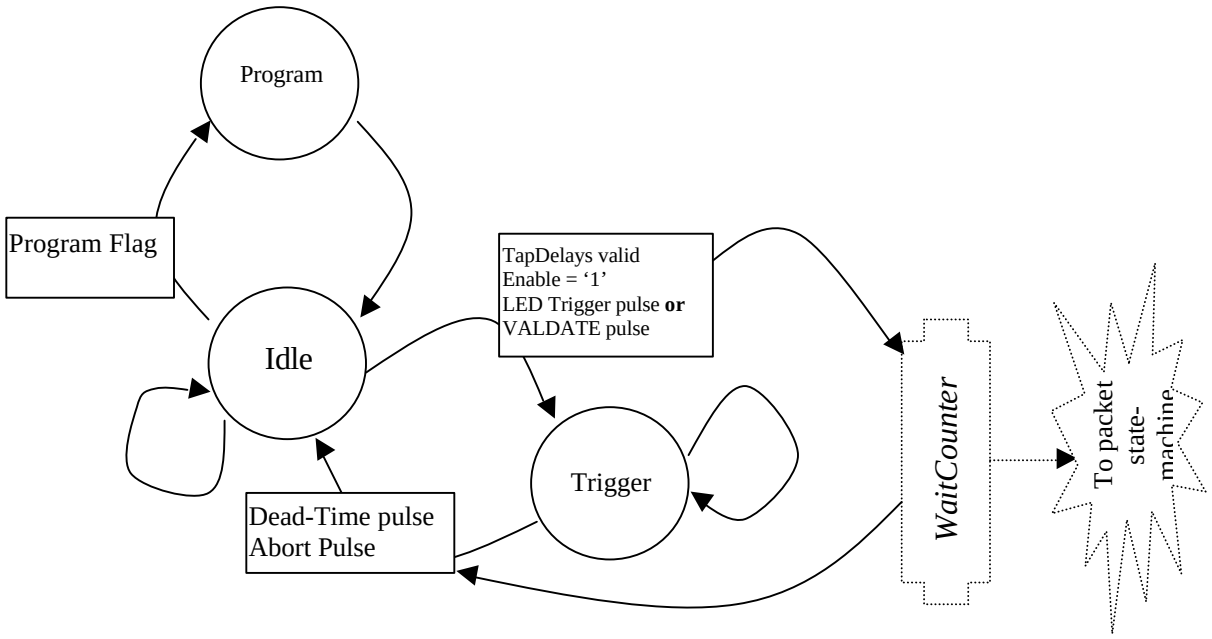

Figure 17: timing/program state-machine

This state machine filters the various LED triggers or External triggers and when one trigger can be taken into account, it starts the dead time/processing timer. Any event arriving during this time will be discarded for packet creation.

Signal Name	**Description**
RESET	Asynchronous reset. It is active high
CLK	Clock.
CHANNEL_ID	The channel ID
PROGData(15 to 0)	The data retrieved by the VME module
PROGAdd(5 to 0)	The address retrieved by the VME module
PROGFlag	A flag indicating new data has been retrieved
PROG_ACK	A pulse acknowledging the programming
TapDelay_Valid(4 to 0)	Status line from the tap delays
LED_TIMESTAMP	Timestamp pulse from the LED module
VALIDATE	A pulse indicating external trigger
LED_SIGN	Sign of the LED crossing
LATCH_TIMESTAMP	A pulse getting the current timestamp
LOADValidationWait	The external validation length load pulse
LOADPileupWait	The pileup length load pulse
LOADExternalWait	The external trigger sliding length load pulse
LOADm	The integration length load pulse

Signal Name	Description
LOADk	The collection length load pulse
LOADRawLength	The raw data length load pulse
LOADSlidingWait	The raw data sliding length load pulse
LOAD_CFDa	The CFD fraction load pulse
LOAD_CFDTH	The CFD Threshold load pulse
LOAD_CFDTap	The CFD delay length load pulse
LOAD_LEDTH	The LED Threshold load pulse
LOAD_LEDTimer	The LED noise window length load pulse
StatusReg(15 to 0)	The status register that can be read back
STARTExternalWait	The synchronous start pulse for external waiting window
STARTInternalWait	The synchronous start pulse for internal waiting window
STARTValidation	The synchronous start pulse for external validation window
MinDeadtimeDone	Indicates that the minimum dead-time has been reached
Abort	Indicates that the packet state-machine aborts (pileup or pre-buffer full)
CLEAR	The clear line for the LED module

Table 25: TimerMachine I/Os

XI.3.b – PacketMachine

This state machine is responsible for generating the cycles where we access the pre-buffer.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
TRIG_MODE(1 to 0)	Trigger mode as defined in register (table 21)
ENABLE	START/STOP mode as defined in register (table 21)
PILEUP	PILEUP mode as defined in register (table 21)
EVENT_PILEUP	Pileup flag on the current event
PREBUFFER_ACK	Acknowledge from the VME interface module that the pre-buffer is being read enough so that it can start being reused
RawSlidingDone	A pulse that indicates when the sliding raw data length is done
ExtValidation	A line that indicates when we are waiting for external validation
ComputedDone	A synchronous pulse that indicates the end of the energy computation window
AddEnable	The enable line for the address increment
RawAdd	A line that indicates when the raw data has to be sent.
VALIDATE	A pulse indicating external validation
LATCH_TIMESTAMP	A pulse getting the current timestamp
STARTSlidingWait	The synchronous start pulse for sliding raw data window
STARTAddWait	The synchronous start pulse for address/raw length window
PREBUFFER_READY	A line indicating that the pre-buffer has been filled enough so that it can start being read on the VME side.
MUX_SELECT	The selection bus

Signal Name	Description
MUX_ENABLE	The multiplexer to pre-buffer data enable line
RESETEvent	A pulse indicating all the data from the current event can be discarded.
ABORT	A priority condition or pileup prevents the data from being sent outside on the VME and therefore we can rearm the event filter.

Table 26: PacketMachine I/Os

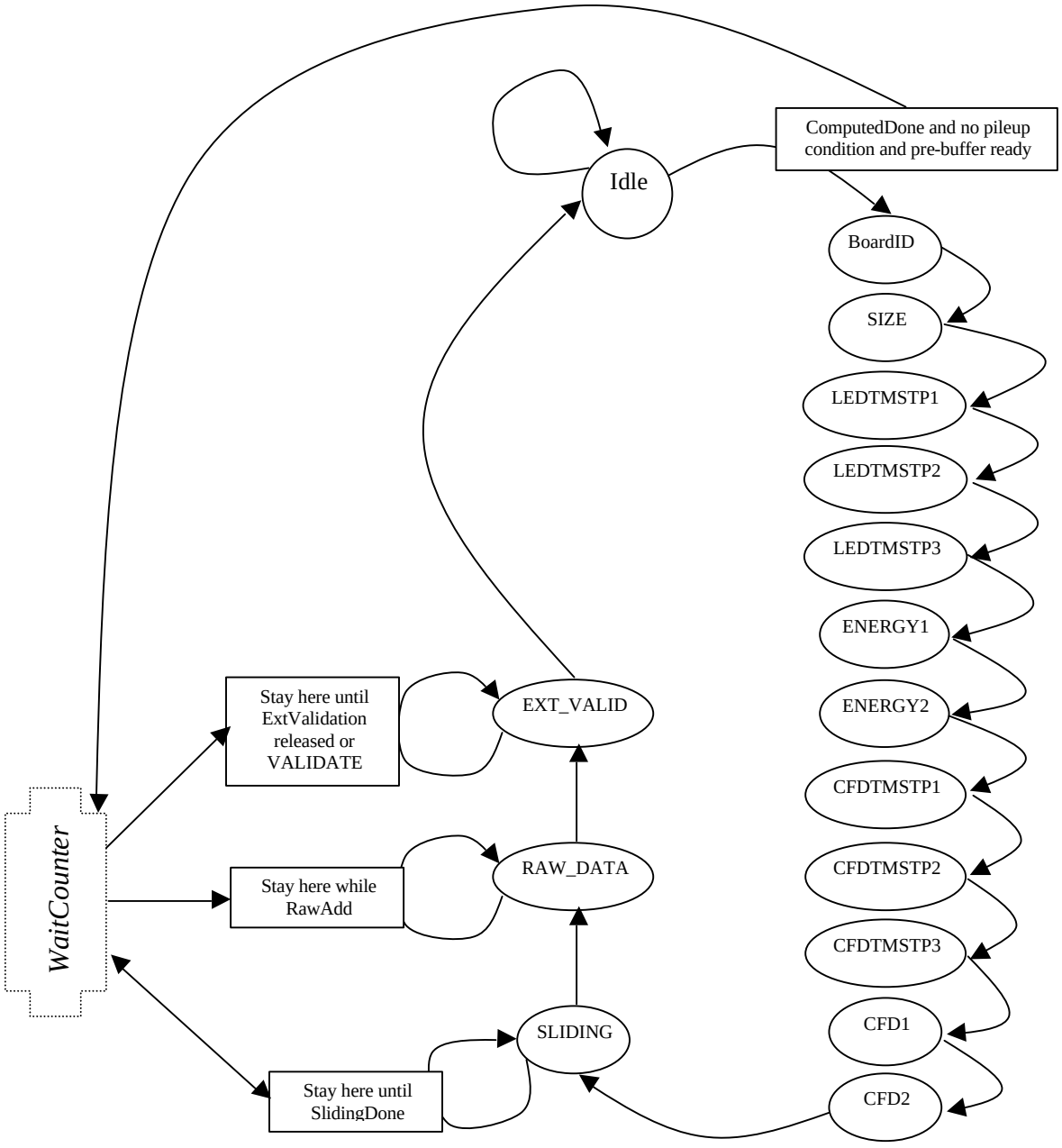

Figure 18: Packet State-machine

XI.4 – PileupCounter

This module implements the pileup detection. It detects the following configurations:

- An event occurred less than the pileup length before when STARTEvent is launched.
- An event occurs less than the pileup length after the STARTEvent is lauched. (obviously a pileup length cannot be more than the minimum time between two STARTEvent)
- If two non considered events pileup but after the pileup length and before a new STARTEvent, the flag is not set.

The following timing diagrams show the various configuration and behaviors:

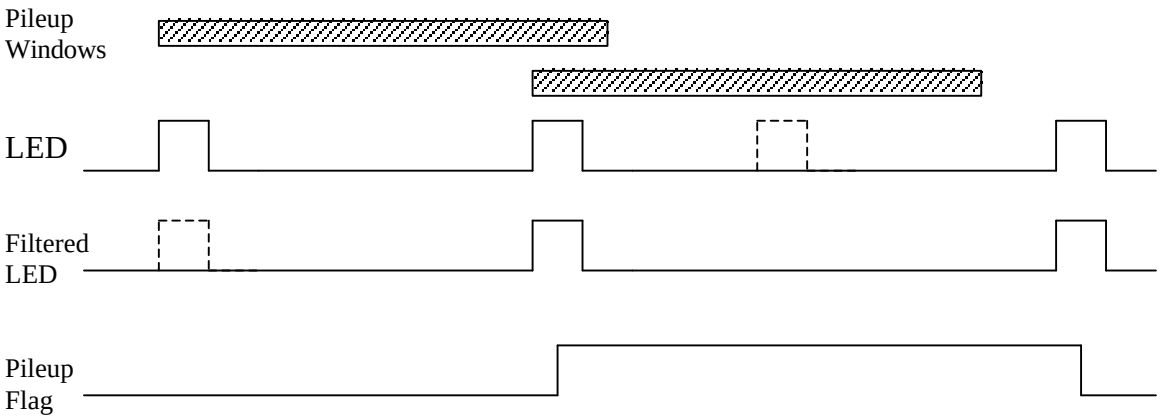

Figure 19: Pre-pileup case

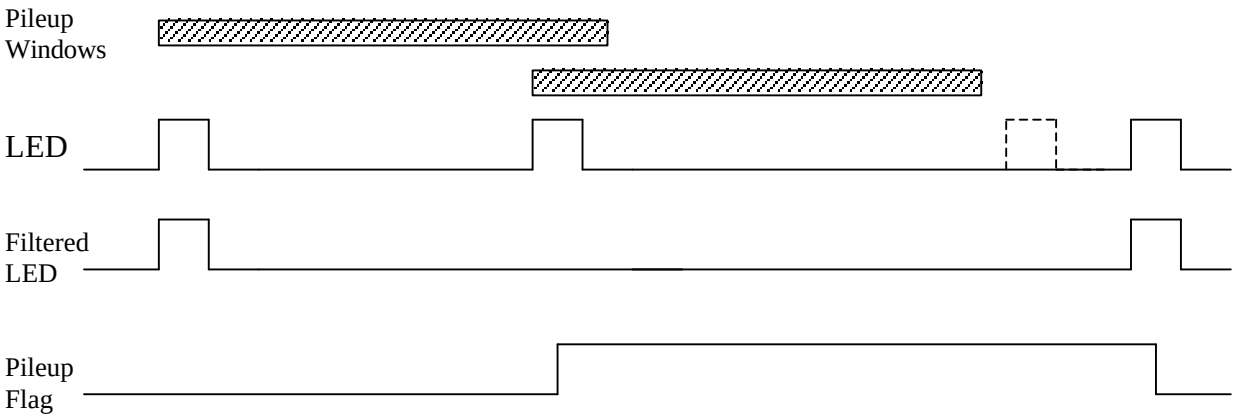

Figure 20: Post-pileup case one

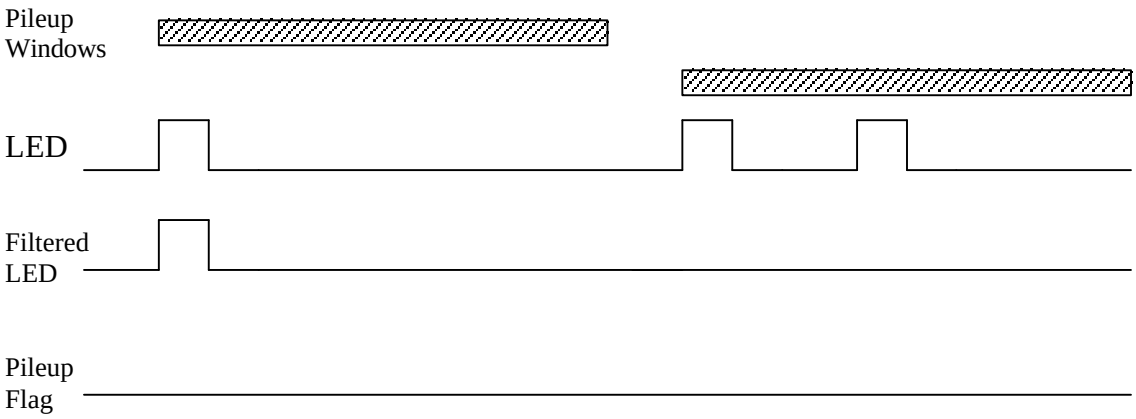

Figure 21: Post-pileup case two

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
LOADPileupWait	The pileup length load pulse
PileupWait	The pileup length
STARTPileup	Any trigger from LED regardless of event filtering
STARTEvent	The trigger pulse for the event we consider
PILEUP_EVENT	Line indicating if the current event being processed has a pileup condition. (It is reset each time a new STARTEvent is sent).

Table 27: PileupCounter I/Os

XI.5 – ProcCore

This module combines all the previous ones and implements the core processing and packet generation. It operates around 129MHz.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
SYNCH	The timer synchronization pulse
BOARD_ID (15 to 3)	The Board ID
CHANNEL_ID (2 to 0)	The channel ID
PROGData(15 to 0)	The data retrieved by the VME module
PROGAdd(5 to 0)	The address retrieved by the VME module
PROGFlag	A flag indicating new data has been retrieved
PROG_ACK	A pulse acknowledging the programming
TapDelay_Valid(4 to 0)	Status line from the tap delays
RAW_DATA:(15 to 0)	The Raw data from the last Tapdelay
ENERGY (22 to 0)	The energy from the peak detector

Signal Name	Description
CFD1 (15 to 0)	CFD Y point 1
CFD2 (15 to 0)	CFD Y point 2
CFD_TIMESTAMP	The timestamp pulse for CFD
CFD_VALID	The flag indicating the CFD point are valid or not
LED_TIMESTAMP	Timestamp pulse from the LED module
LED_SIGN	Sign of the LED crossing
VALIDATE	Pulse validating an event in external validation trigger mode
LOADm	The integration length load pulse
LOADk	The collection length load pulse
LOAD_CFDa	The CFD fraction load pulse
LOAD_CFDTH	The CFD Threshold load pulse
LOAD_CFDTap	The CFD delay length load pulse
LOAD_LEDTH	The LED Threshold load pulse
LOAD_LEDTimer	The LED noise window length load pulse
StatusReg(15 to 0)	The status register that can be read back
RESETEvent	A pulse indicating all the data from the current event can be discarded.
SIGNEvent	A line indicating the sign of the event that has been filtered by the states-machines. (When a LATCH_TIMESTAMP is initiated).
PREBUFFER_ACK	Acknowledge from the VME interface module that the pre-buffer is being read enough so that it can start being reused
PREBUFFER_READY	A line indicating that the pre-buffer has been filled enough so that it can start being read on the VME side.
PREBUFFER_ADDRESS(9 to 0)	The Address where the data has to be written in the packet
PREBUFFER_DATA(15 to 0)	The data that has to be written in the packet
Size (8 to 0)	Size of the packet in long words
PREBUFFER_EN	Enable for the packet memory
CLEAR	The clear line for the LED module

Table 28: WaitCounter I/Os

The resources used are as follows:

Slices: 371
Block RAM: 0

The following figure shows the general interconnect between the sub-modules.

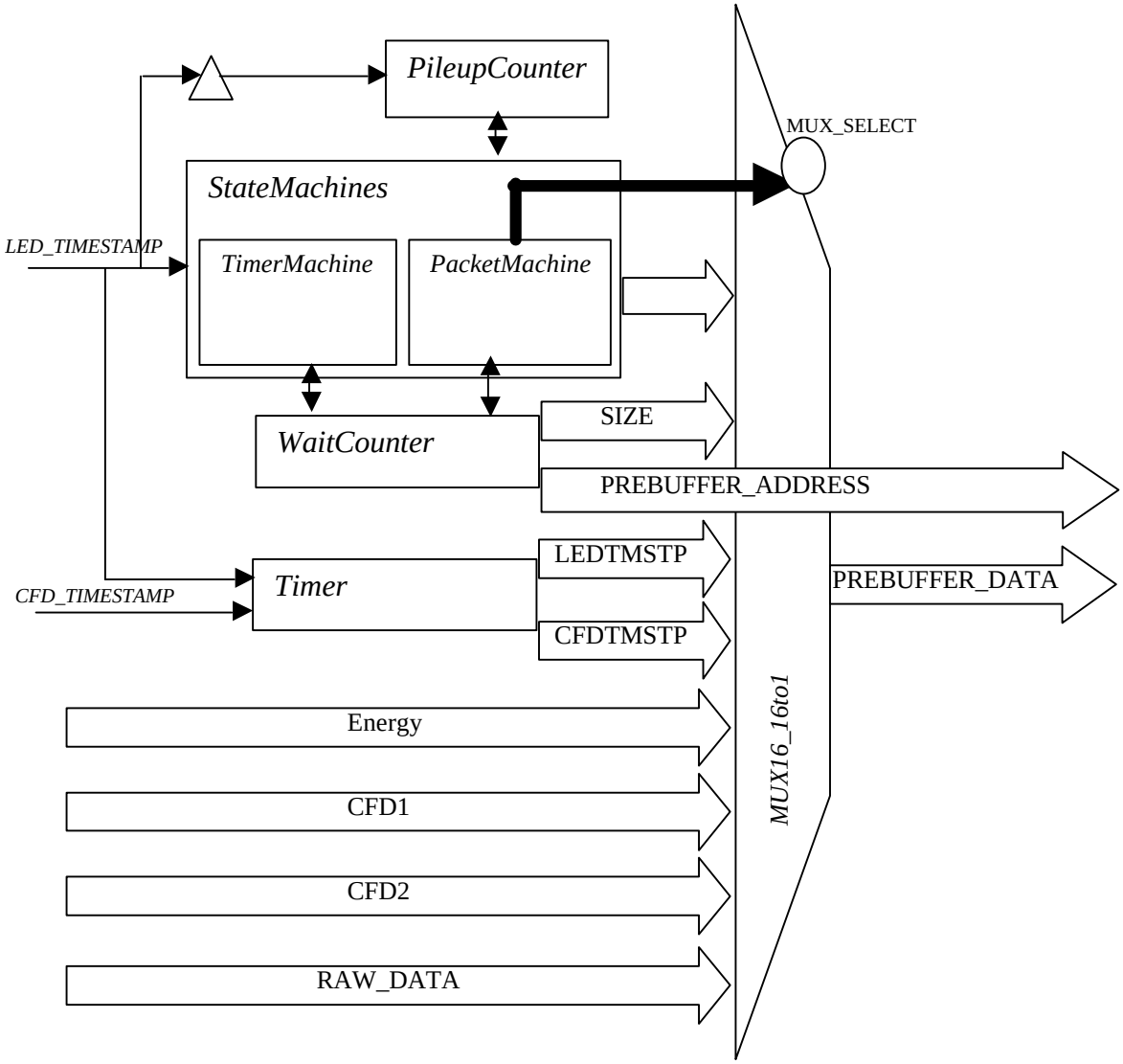

Figure 22: ProcCore simple interconnect (not all control lines are drawn)

XII – CHANNEL

This module implements the full digital processing for one channel. It includes the multiplexer for ADC data and debug data as well as the pre-buffer on the other end. It does not contains the VME load engine and the FIFO write engine as they are shared by all the channels.

The resources used are as follows:

Slices: 925
Block RAM: 4

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
SYNCH	The timer synchronization pulse
BOARD_ID (15 to 3)	The Board ID
CHANNEL_ID (2 to 0)	The channel ID
PROGData(15 to 0)	The data retrieved by the VME module
PROGAdd(5 to 0)	The address retrieved by the VME module
PROGFlag	A flag indicating new data has been retrieved
PROG_ACK	A pulse acknowledging the programming
ADCData(11 to 0)	The Raw data from the ADC
DEBUGData(15 to 0)	Data from the debug module
ADDRESSout (9 to 0)	Pre-buffer address
DATAout(15 to 0)	Pre-buffer Data
CLKout	Pre-buffer clock
ENABLEout	Pre-buffer enable
VALIDATE	Pulse validating an event in external validation trigger mode
LED_TRIGGER	Leading edge trigger
BUSY	Line indicating we are busy processing
StatusReg(15 to 0)	The status register that can be read back
PREBUFFER_ACK	Acknowledge from the VME interface module that the pre-buffer is being read enough so that it can start being reused
PREBUFFER_READY	A line indicating that the pre-buffer has been filled enough so that it can start being read on the VME side.
Size (8 to 0)	Size of the packet in long words

Table 29: Channel I/Os

The diagram shown next page describes the data flow for one channel

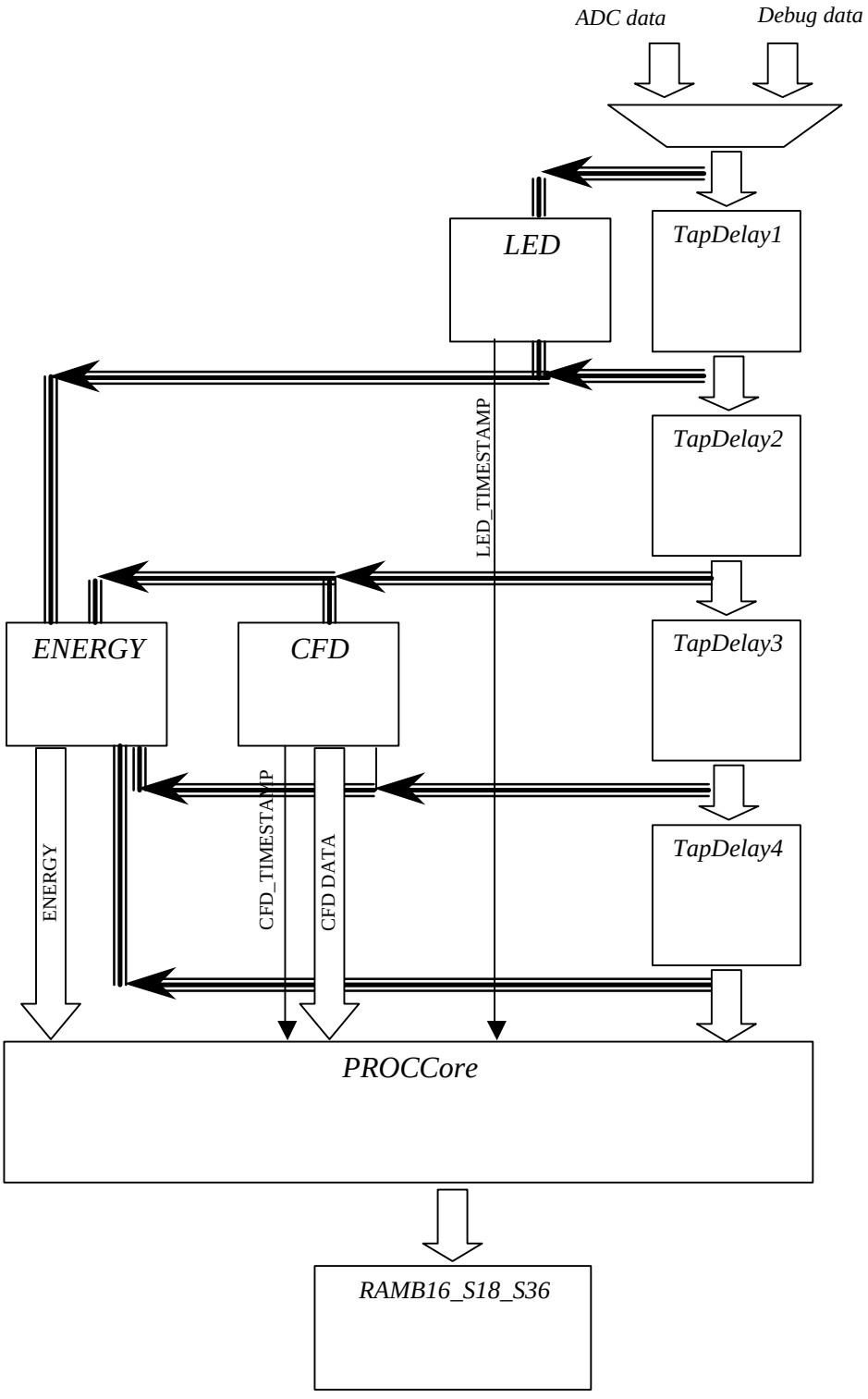

Figure 23: Channel Data Flow

XIII – DEBUG

This module implements a simple dual port memory that can be loaded from the VME and that can output a thousand raw points to all the channels. As long as at least one channel is in debug mode, the debug data is clocked out.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
DEBUGFlags (7 to 0)	Debug flag: one per channel
PROGData(15 to 0)	The data retrieved by the VME module
PROGAdd(5 to 0)	The address retrieved by the VME module
PROGFlag	A flag indicating new data has been retrieved
PROG_ACK	A pulse acknowledging the programming
DEBUGData (15 to 0)	The debug data bus

Table 30: DEBUG I/Os

This module uses two addresses from the Processing block VME register address space (see table 21) in order to load the address and data to the debug memory.

XIV – FIFOInterface

This module multiplex the eight channel packet data and generates the proper control signals for the FIFO transfer (write path). The operating frequency of this module is half the one used for normal processing since we de-multiplex the data from 2 byte wide to 4 byte wide.
This module contains some sub-modules described in the following sub-sections.

XIV.1 – FIFOMachine

This module implements a state machine with counters in order to perform a non priority multiplexing of the 8 input data path.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
PREBUFFER_ADDRESS (8 to 0)	The address for the pre-buffers
PREBUFFER_ENABLE (7 to 0)	The enable for the 8 channel pre-buffers
PREBUFFER_ACK (7 to 0)	Pre-buffer acknowledgement for the 8 channels
PREBUFFER_READY (7 to 0)	Pre-buffer ready line for the 8 channels
FIFO_PAFneg	FIFO almost full flag (should be set so that we have at least more than one full size packet left): full FIFO size minus 2048 would be fine even though 512 is the minimum required.

Signal Name	Description
FIFO_WENneg	FIFO write enable
Size (8 to 0)	Packet size for current channel
CHANNEL_SELECT (2 to 0)	This indicates which channel has to be used
ENABLE	This is the enable for the external multiplexers

Table 31: FIFOMachine I/Os

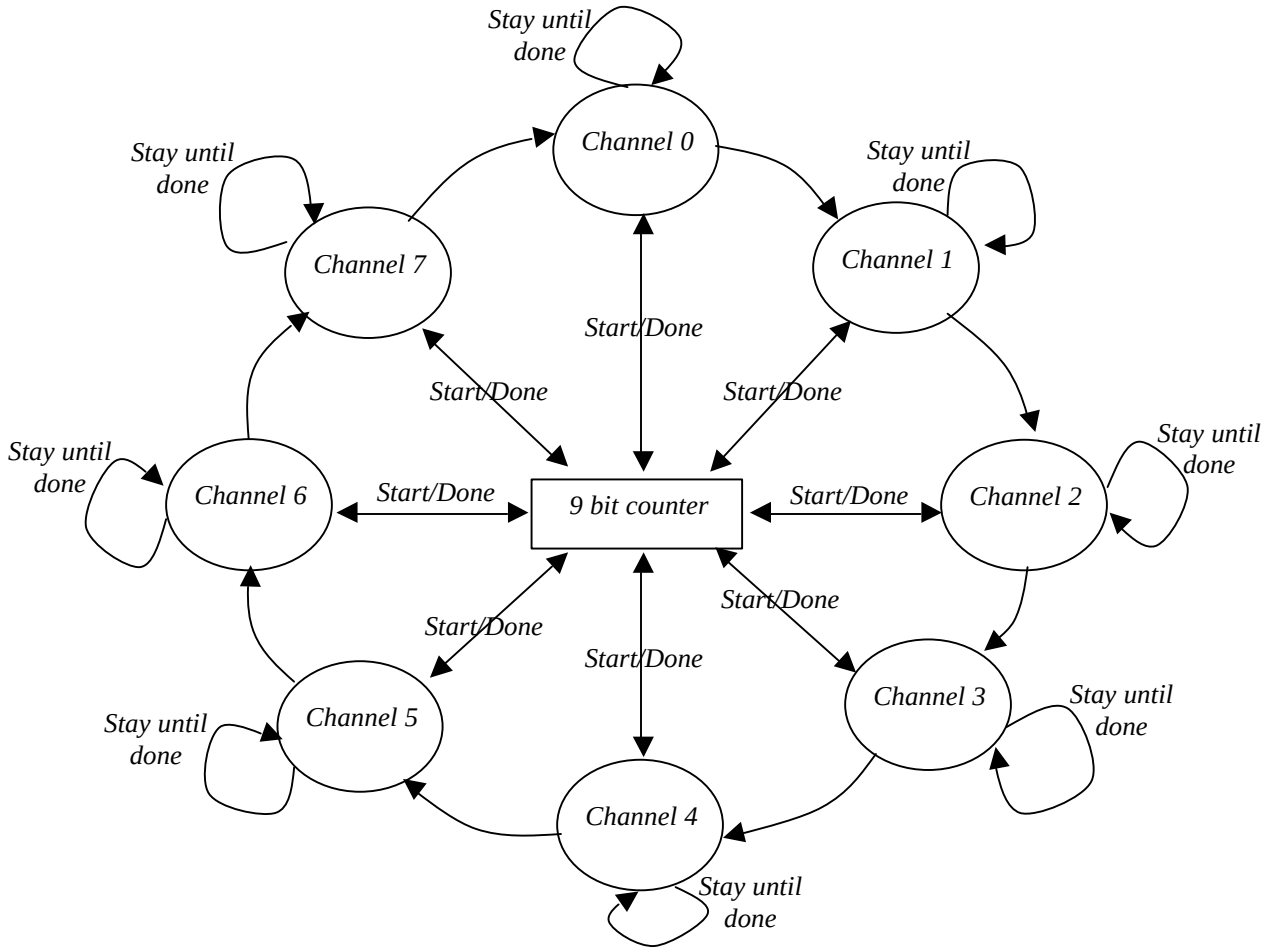

Figure 24: FIFO State machine with counter

If the pre-buffer is not ready when the 'token' is passing through a channel, it is then propagated directly to the next channel. If the same channel is ready again as soon as we are done (which cannot happen in the full system but just in case) the token is propagated to the next channel anyway to avoid lock-up on a particular channel. If the same channel is the only one active, we have to scan the entire loop (8 clock cycle) before it can be serviced again. The states of the state-machine are directly used to select which channel is used so that we save logic. However, there is no real speed concerns for this module as it goes at half the processing speed. The 9 bit counter is loaded with the packet size when a pre-buffer is detected as having to be serviced. The output data of the counter is used as addresses for the pre-buffers.

XIV.2 – FIFOInterface

This module uses the previous sub-module and several multiplexer modules in order to implement the full data multiplexing to the FIFO.
The following table lists the I/Os and the following diagram shows the architecture.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
DataCh0 (31 to 0)	Packet data for channel 0
DataCh1 (31 to 0)	Packet data for channel 1
DataCh2 (31 to 0)	Packet data for channel 2
DataCh3 (31 to 0)	Packet data for channel 3
DataCh4 (31 to 0)	Packet data for channel 4
DataCh5 (31 to 0)	Packet data for channel 5
DataCh6 (31 to 0)	Packet data for channel 6
DataCh7 (31 to 0)	Packet data for channel 7
Size0 (8 to 0)	Packet size for channel 0
Size1 (8 to 0)	Packet size for channel 1
Size2 (8 to 0)	Packet size for channel 2
Size3 (8 to 0)	Packet size for channel 3
Size4 (8 to 0)	Packet size for channel 4
Size5 (8 to 0)	Packet size for channel 5
Size6 (8 to 0)	Packet size for channel 6
Size7 (8 to 0)	Packet size for channel 7
PREBUFFER_ADDRESS (8 to 0)	The address for the pre-buffers
PREBUFFER_ENABLE (7 to 0)	The enable for the 8 channel pre-buffers
PREBUFFER_ACK (7 to 0)	Pre-buffer acknowledgement for the 8 channels
PREBUFFER_READY (7 to 0)	Pre-buffer ready line for the 8 channels
FIFO_PAFneg	FIFO almost full flag (should be set so that we have at least more than one full size packet left): full FIFO size minus 2048 would be fine even though 512 is the minimum required.
FIFOData (31 to 0)	Data lines to the FIFO
FIFO_WENneg	FIFO write enable

Table 32: FIFOInterface I/Os

The state-machine basically controls the multiplexing and provides the prebuffers addresses. They are then selected by an enable. The data multiplexer has an external enable that outputs a pattern when not enable. This pattern is used as a packet separator.
It is currently set to 0xAAAAAAAA.

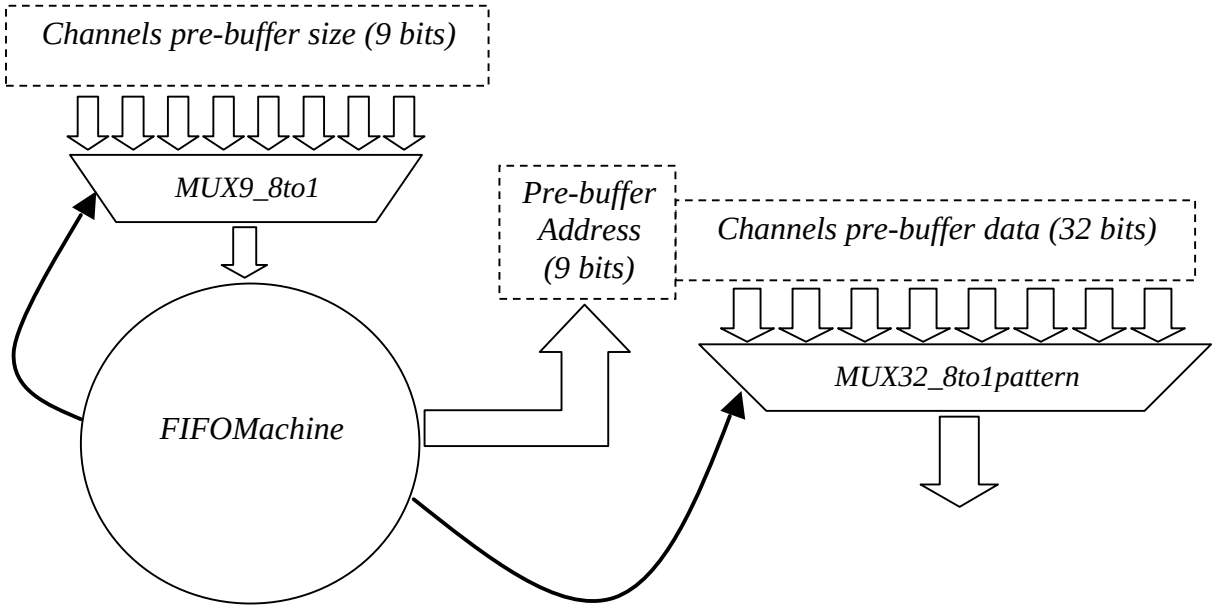

Figure 25: FIFO Interface architecture

XV – EightChannel

This module can be seen as an independent full processing block. It contains the DEBUG module, eight channels processing chain and the FIFO interface module. This block can be reused independently from the BUS interface or the various electrical level (LVCMOS, LVPECL, …) used.

Signal Name	**Description**
RESET	Asynchronous reset. It is active high
CLK	Clock.
ADCData0 (11 to 0)	Input ADC Data for channel 0
ADCData1 (11 to 0)	Input ADC Data for channel 1
ADCData2 (11 to 0)	Input ADC Data for channel 2
ADCData3 (11 to 0)	Input ADC Data for channel 3
ADCData4 (11 to 0)	Input ADC Data for channel 4
ADCData5 (11 to 0)	Input ADC Data for channel 5
ADCData6 (11 to 0)	Input ADC Data for channel 6
ADCData7 (11 to 0)	Input ADC Data for channel 7
StatusReg0 (15 to 0)	Status register for channel 0
StatusReg1 (15 to 0)	Status register for channel 1
StatusReg2 (15 to 0)	Status register for channel 2
StatusReg3 (15 to 0)	Status register for channel 3
StatusReg4 (15 to 0)	Status register for channel 4
StatusReg5 (15 to 0)	Status register for channel 5

Signal Name	Description
StatusReg6 (15 to 0)	Status register for channel 6
StatusReg7 (15 to 0)	Status register for channel 7
PROGData(15 to 0)	The data retrieved by the VME module
PROGAdd(5 to 0)	The address retrieved by the VME module
PROGFlag	A flag indicating new data has been retrieved
PROG_ACK	A pulse acknowledging the programming
SYNCH	The timer synchronization pulse
BOARD_ID (15 to 3)	The Board ID
VALIDATE (7 to 0)	External trigger (one per channel)
GLOBAL_VALIDATE	Global External trigger
FIFO_PAFneg	FIFO almost full flag (should be set so that we have at least more than one full size packet left): full FIFO size minus 2048 would be fine even though 512 is the minimum required.
FIFOData (31 to 0)	Data lines to the FIFO
FIFO_WENneg	FIFO write enable
LED_TRIGGER (7 to 0)	Output trigger (one per channel)
GLOBAL_TRIGGER	Global output trigger
EVENT_READ (7 to 0)	Event is being read into FIFO (one per channel)
BUSY	The board is busy processing (global busy)
CLKout	Output FIFO clock.

Table 33: Eight_Channel I/Os

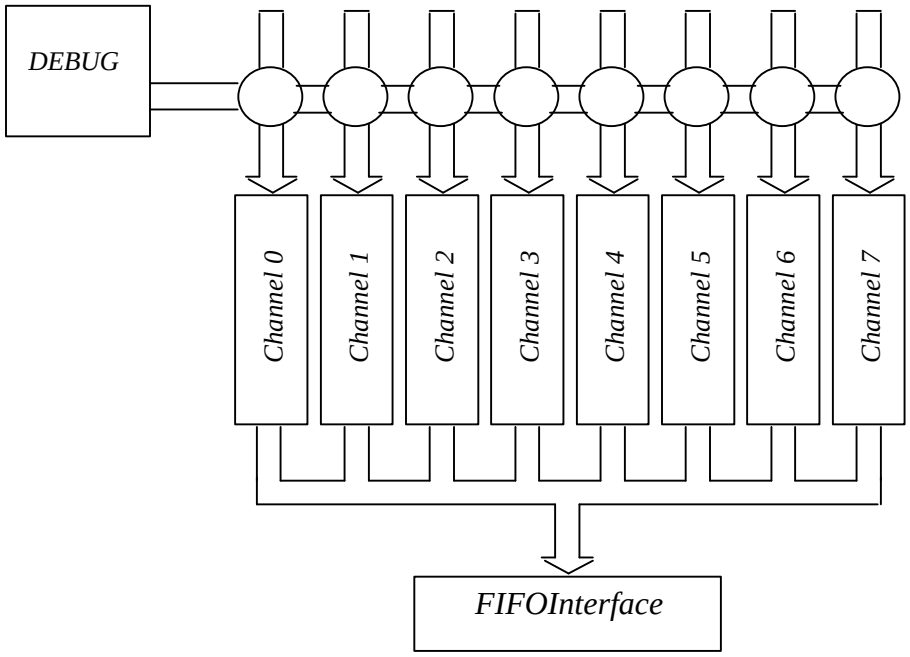

Figure 26: Eight_Channel architecture

XVI – DACControl

This module controls the two LTC1660 used to adjust the analogue section. It has been rewritten so that it does not clock all the time, hence further reducing eventual noise on the board. It uses a clock of 3.125MHz (50MHz divided by 16) and programs the DACs with a gated clock of 1.5625MHz, which is well inside the allowed programming frequency.

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
PROGData(15 to 0)	The data retrieved by the VME module
PROGAdd(5 to 0)	The address retrieved by the VME module
PROGFlag	A flag indicating new data has been retrieved
PROG_ACK	A pulse acknowledging the programming
DAC_Din	DACs serial data in
DAC_CSLD0	DAC 0 chip select
DAC_CSLD1	DAC 1 chip select
DAC_SCK	DACs serial clock

Table 34: DACControl I/Os

The following address space is used for the DAC programming:

- 0XXXXXXb is reserved for the processing core programming (see table 21). Only 5 bit are provided for the decoding because it allows easy module instantiation for reuse with different analogue section.
- 100XXXXb is used for programming the DACs as follows:

Address Range	Description
1000XXXb	DAC 0 programming (the 3 LSBs are directly mapped to the DAC programming addresses for address 1 to 7 corresponding to line A to G, and address 0 is remapped to line H)
1001XXXb	DAC 1 programming (the 3 LSBs are directly mapped to the DAC programming addresses for address 1 to 7 corresponding to line A to G, and address 0 is remapped to line H)

Table 35: DAC programming address space

One DAC is used to adjust the Gain of the analog chain and the other is used to adjust an offset on the analogue chain.

XVII – VMEControl

This module handles the VME transactions for registers programming and for FIFO readout. It is added outside the processing block as it accommodates functions external to the main processing such as launching the DAC programming.

The following rules will be followed:

- The IRQ interrupt vector is the 8 LSBs of the Board ID (13 bits)
- The registers are accessed in A16/D16 mode with A15 to A8 for decoding what board is being accessed, based on the 8 LSBs of the Board ID. It gives a register region size of 128 D16 words.
- The FIFO is accessed in A24/D32 mode with A23 to A16 for decoding the address space. It gives a block size of 16384 D32 words for DMA readout.

The module is supposed to interface with the VME chipset CY7C960/CY7C974 from Cypress. The following configuration has been used:
The four CY7C964's decode the addresses as follows:

CY7C964 #0 (A7 to A0-D7 to D0): not used
CY7C964 #1 (A15 to A8-D15 to D8): decode the BoardID value
CY7C964 #2 (A24 to A16-D24 to D15): decode the BoardID value
CY7C964 #4 (none-D7 to D0): not used

Only two region bits are used since we split the VME address space as described above. The following region mapping is used:

- "00": unused – Idle selection
- "01": Register space decoded with CY7C964 #1 VCOMP and AM selecting A16 mode (0x28, 0x29, 0x2C, 0x2D)
- "10": Register space decoded with CY7C964 #2 VCOMP and AM selecting A24 mode (0x32, 0x33, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F)
- "11": unused

The chipset is to be programmed by an EEPROM and the companion CY7C964 programmed by the module at power-up. Here is a summary of the option required:

- Serial Prom programming
- DBE polarity high
- AM/LA multiplexing
- Lack response is supported. Can be tied to '0' on the chipset too.
- No Bus hold-off
- No Master interleave
- I/O configuration
- CS0, CS1 polarity high
- Region 0: CS0=0, CS1=0, DBE assert time at least 3
- Region 1: CS0=1, CS1=0, DBE assert time at least 3
- Region 2: CS0=0, CS1=1, DBE assert time at least 3
- Region 3: CS0=1, CS1=1, DBE assert time at least 3
- Region 0: All I/O modes disable, no special code
- Region 1: A16 mode enable, all the others disable, no special code
- Region 2: A24 mode enable, all the others disable, no special code
- Region 3: All I/O modes disable, no special code

Signal Name	Description
RESET	Asynchronous reset. It is active high
BOARD_ID (15 to 3)	The Board ID
PROGData(15 to 0)	The data retrieved by the VME module
PROGAdd(5 to 0)	The address retrieved by the VME module
PROGFlag	A flag indicating new data has been retrieved
PROG_ACK	A pulse acknowledging the programming
FIFO_PAEneg	FIFO almost empty flag
FIFO_PAFneg	FIFO almost full flag (should be set so that we have at least more than one full size packet left): full FIFO size minus 2048 would be fine even though 512 is the minimum required.
FIFO_EFneg	FIFO empty flag
FIFO_HFneg	FIFO half full flag
FIFO_FFneg	FIFO full flag
FIFO_RENneg	FIFO read enable synchronous line
FIFO_OEneg	FIFO output enable asynchronous line
FIFO_ACCESSED	FIFO being read flag used for LED
SVIC_CLK	VME chipset clock
SVIC_STROBE	VME strobe requesting reconfiguration
SVIC_LDS	VME signal for data multiplex transactions
SVIC_CS (1 to 0)	VME chip select output (define which region is active)
SVIC_REGION (1 to 0)	VME region select input
SVIC_DBE (3 downto 0)	VME read/write strobe
SVIC_LIRQneg	VME interrupt request
SVIC_R_W	VME read/write line
SVIC_LADI	VME Address modifier latch
SVIC_LACKneg	VME local acknowledgement
SVIC_PRENneg	VME prom enable
SVIC_LDENneg	VME control line for interrupt
SVIC_LDS	VME control line for multiplexing
XCVR_STROBE	VME Strobe used to configure the companion chips
XCVR_LDS	VME control line used to configure the companion chips
XCVR_MWB	VME control line used to configure the companion chips
LDATAin (15 downto 0)	VME module 16 bit data in
LDATAout (15 downto 0)	VME module 16 bit data out
LDATA_Tristate	VME module data tri-state select line
XCVR_VCOMP (1 downto 0)	VME address match lines
XCVR_CONFIG (7 downto 0)	VME module 8 bit data line used only to configure the companion chip #2
LADDR (7 downto 1)	VME module address/address modifier
StatusReg0 (15 to 0)	Status register for channel 0
StatusReg1 (15 to 0)	Status register for channel 1
StatusReg2 (15 to 0)	Status register for channel 2
StatusReg3 (15 to 0)	Status register for channel 3
StatusReg4 (15 to 0)	Status register for channel 4
StatusReg5 (15 to 0)	Status register for channel 5

Signal Name	Description
StatusReg6 (15 to 0)	Status register for channel 6
StatusReg7 (15 to 0)	Status register for channel 7

Table 36: VMEControl I/Os

XVIII – CHIP

This module is the top module before I/O pad configuration. It contains the previous top modules:

Eight_Channel
DACControl
VMEControl

Signal Name	Description
RESET	Asynchronous reset. It is active high
CLK	Clock.
SYNCH	The timer synchronization pulse
BOARD_ID (15 to 3)	The Board ID
ADCData0 (11 to 0)	Input ADC Data for channel 0
CLKCh0	Individually adjusted positive clock for channel 0
CLKCh0neg	Individually adjusted negative clock for channel 0
ADCData1 (11 to 0)	Input ADC Data for channel 1
CLKCh1	Individually adjusted positive clock for channel 1
CLKCh1neg	Individually adjusted negative clock for channel 1
ADCData2 (11 to 0)	Input ADC Data for channel 2
CLKCh2	Individually adjusted positive clock for channel 2
CLKCh2neg	Individually adjusted negative clock for channel 2
ADCData3 (11 to 0)	Input ADC Data for channel 3
CLKCh3	Individually adjusted positive clock for channel 3
CLKCh3neg	Individually adjusted negative clock for channel 3
ADCData4 (11 to 0)	Input ADC Data for channel 4
CLKCh4	Individually adjusted positive clock for channel 4
CLKCh4neg	Individually adjusted negative clock for channel 4
ADCData5 (11 to 0)	Input ADC Data for channel 5
CLKCh5	Individually adjusted positive clock for channel 5
CLKCh5neg	Individually adjusted negative clock for channel 5
ADCData6 (11 to 0)	Input ADC Data for channel 6
CLKCh6	Individually adjusted positive clock for channel 6
CLKCh6neg	Individually adjusted negative clock for channel 6
ADCData7 (11 to 0)	Input ADC Data for channel 7
CLKCh7	Individually adjusted positive clock for channel 7
CLKCh7neg	Individually adjusted negative clock for channel 7
FIFO_PAEneg	FIFO almost empty flag
FIFO_PAFneg	FIFO almost full flag (should be set so that we have at least more than one full size packet left): full FIFO size minus 2048

Signal Name	Description
	would be fine even though 512 is the minimum required.
FIFO_EFneg	FIFO empty flag
FIFO_HFneg	FIFO half full flag
FIFO_FFneg	FIFO full flag
FIFO_RENneg	FIFO read enable synchronous line
FIFO_OEneg	FIFO output enable asynchronous line
FIFOData (31 to 0)	FIFO input data
FIFO_WENneg	FIFO write enable synchronous line
FIFO_WCLK	FIFO write clock
VALIDATE (7 downto 0)	External individual trigger/validate
GLOBAL_VALIDATE	Global trigger/validate
LED_TRIGGER (7 to 0)	Channel Leading edge trigger (10ns wide)
EVENT_READ (7 to 0);	Event being written to FIFO
ENABLE_STATUS (7 to 0)	Channel enable status lines
BUSY	Board Busy with events
FIFO_ACCESSED	FIFO being read flag used for LED
DAC_Din	DACs serial data in
DAC_CSLD0	DAC 0 chip select
DAC_CSLD1	DAC 1 chip select
DAC_SCK	DACs serial clock
SVIC_CLK	VME chipset clock
SVIC_STROBE	VME strobe requesting reconfiguration
SVIC_LDS	VME signal for data multiplex transactions
SVIC_CS (1 to 0)	VME chip select output (define which region is active)
SVIC_REGION (1 to 0)	VME region select input
SVIC_DBE (3 downto 0)	VME read/write strobe
SVIC_LIRQneg	VME interrupt request
SVIC_R_W	VME read/write line
SVIC_LADI	VME Address modifier latch
SVIC_LACKneg	VME local acknowledgement
SVIC_PRENneg	VME prom enable
SVIC_LDENneg	VME control line for interrupt
SVIC_LDS	VME control line for multiplexing
XCVR_STROBE	VME Strobe used to configure the companion chips
XCVR_LDS	VME control line used to configure the companion chips
XCVR_MWB	VME control line used to configure the companion chips
LDATAin (15 downto 0)	VME module 16 bit data in
LDATAout (15 downto 0)	VME module 16 bit data out
LDATA_Tristate	VME module data tri-state select line
XCVR_VCOMP (1 downto 0)	VME address match lines
XCVR_CONFIG (7 downto 0)	VME module 8 bit data line used only to configure the companion chip #2
LADDR (7 downto 1)	VME module address/address modifier

Table 37: Chip I/Os

XIX – Packaging the design

In order to have a VHDL definition of the output (not through the UCF file), several VHDL modules have been created containing only the I/O/tri-state/bi-directional pads. They do not contain any logic and can be modified independently from the actual processing so that it is easy to customize the output signal characteristics. The ADC differential clocks lines are LVPECL, the status lines are LVTTL and all the other signals are LVCMOS.

XX – Performances, Area of concern

After synthesis, mapping, placement and routing, we have the following results in terms of resources used and speed issues.

Device utilization summary:

Number of External DIFFMs 8 out of 258 3%
(PECL differential pair)
Number of External DIFFSs 8 out of 258 3%
(PECL differential pair)
Number of External IOBs 243 out of 516 46%
(Standard I/Os)

Number of RAMB16s 41 out of 96 42%
(4/channel for tap delays, 1/channel for pre-buffer, one global for the debug module)
Number of SLICEs 7973 out of 14336 55%
(~1000/channel)
Number of BUFGMUXs 15 out of 16 93%
(Main clock fan out)
Number of DCMs 11 out of 12 91%
(Digital Clock Manager: 8 for the 8 ADCs, two for the FIFO clock with divide by 2 and phase adjustment, one for the DACs control)

The number of Slices used is reasonable for further modification enhancement. However, the main clock routing fan-out is almost at its limit especially because of the use of so many different clocks.

Timing summary:

Minimum period for main clock is 8.764ns.
(~114MHz)
Minimum period for FIFO clock is 14.358ns.
(FIFO clock is main clock divided by 2)
Minimum period for VME clock is 9.249ns.
(~108MHz)
Minimum period for DAC Clock is 11.690ns.
(~DAC Clock is main clock divided by 32)

Power summary:

	I(mA)	P(mW)
Total estimated power consumption:		2076
Vccint 1.5V:	1264	1895
Vcco 3.3V:	55	180
Nets:	721	1082
Logic:	343	514
Outputs:	120	180
Quiescent:	200	300

Thermal summary:
Estimated junction temperature: 53C (a small heat sink is advisable)
Ambient temp: 25C
Theta J-A: 14C/W

Several areas may be of concern during actual hardware debug time:

- Metastability between the VME clock and the main clock when write program commands are sent. Since those two clocks are asynchronous one to another, there is a tiny probability that when data is transferred from one to another the setup time is not respected. Xilinx guarantees that a delay of 3ns before latching again after a metastable event provides one error every 10,000 years in average. In the case of errors a method can be devised to guarantee a better operation at the cost of speed.
- I/O block delay between the clock and the synchronous FIFO controls. The delay introduced by the pad routing might prove to be too important for synchronous operation at the FIFO level (the read-out of registers to VME is of no importance as we can increase at will the number of clock cycle for a read sequence). If proven to be the case, a method involving packing registers in I/O block can be used for the critical signals, shortening the output delay time but slowing eventually the entire design.
- ADC/FIFO clock phase adjustment. The Clock adjustment can only be done independently by 25% of the period. Special care will have to be used for the final adjustment once the board is laid out and the analog section is tested.

After Auto routing with carry-chains constraints and period constraints we have roughly the following allocation inside the FPGA:

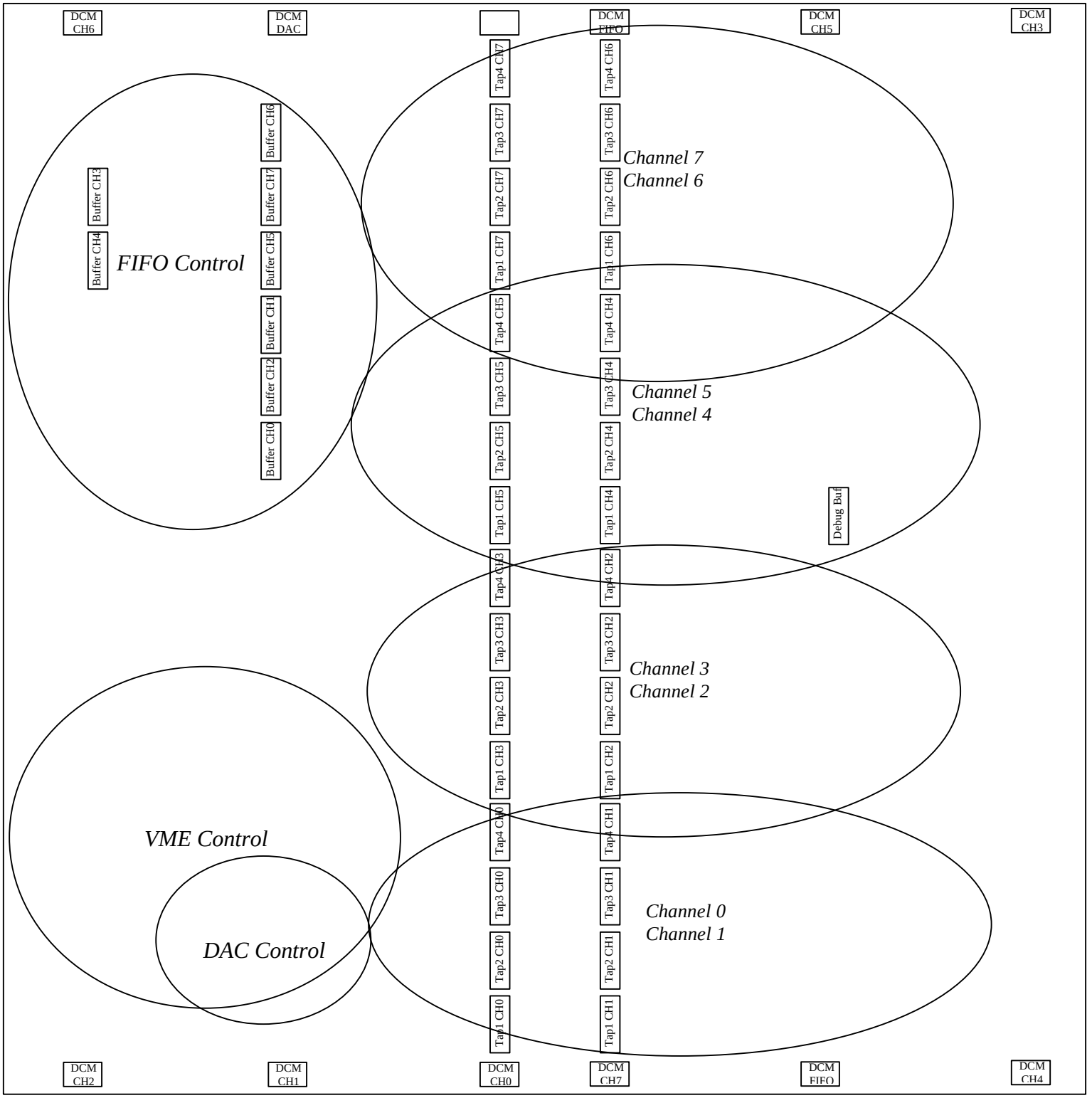

Figure 27: FPGA Placement