EICROC Digital Readout Architecture Implementation Specification

March 30, 2026

Version	Author	Date	Description
0.1	N. KACHKACHI	19/09/2025	1st version
0.2	N. KACHKACHI	12/11/2025	2nd version. Addition of the digital periphery. Update of the pixel and cluster arbiter.
0.3	N. KACHKACHI	1/12/2025	3rd version. Corrections.
0.4	N. KACHKACHI	13/02/2026	4th version. Updates.
0.5	N. KACHKACHI	30/03/2026	5th version. Updates: Signals naming, whishbone IF, general updates according to synthesis requirements .

Contents

1 EICROC Digital Readout Brief Description **3**
1.1 General Description of EICROC Digital Readout 4

2 Cluster Digital Implementation **6**
2.1 Cluster digital top 7
2.1.1 Cluster digital top pinout 8
2.2 Pixel sub-bloc 11
2.2.1 Pixel Sub-Block Pinout 11
2.2.2 Ready/Valid handshake protocol 13
2.2.3 Pixel sub-block Control Unit 15
2.2.4 Pixel sub-block Data Path 18
2.3 Cluster arbiter sub-bloc 20
2.3.1 Cluster arbiter Pinout 20
2.3.2 Cluster arbiter control unit 22
2.3.3 Cluster arbiter data path 25
2.4 Pipelining 30

3 Digital Periphery **32**
3.1 Digital Periphery 33
3.1.1 Digital periphery pinout 34
3.2 Hub sub-block 36
3.3 fifos periphery sub-block 36
3.3.1 Fifos periphery Control Unit 36
3.3.2 FIFOS periphery Data Path 39
3.4 Read Matrix Handler sub-block 40
3.4.1 Read matrix handler control unit 41
3.4.2 Read matrix handler data path 43

Chapter 1

EICROC Digital Readout Brief Description

1.1 General Description of EICROC Digital Readout

The EICROC digital readout manages the recording, the arbitration, the sending and the formatting of data generated by the ADC and TDC corresponding to active pixels. The ASIC pixels matrix consists of 32 × 32 identical pixels, arranged into groups of four pixels called clusters. The pixels matrix is thus composed of 256 clusters, organized in 16 columns of 16 clusters each, that communicate with an End-of-Column (EOC). The addressing of the clusters columns is controlled by the digital periphery, which handles the clusters readout as well.The figure 1.1 shows a block diagram of the digital readout.

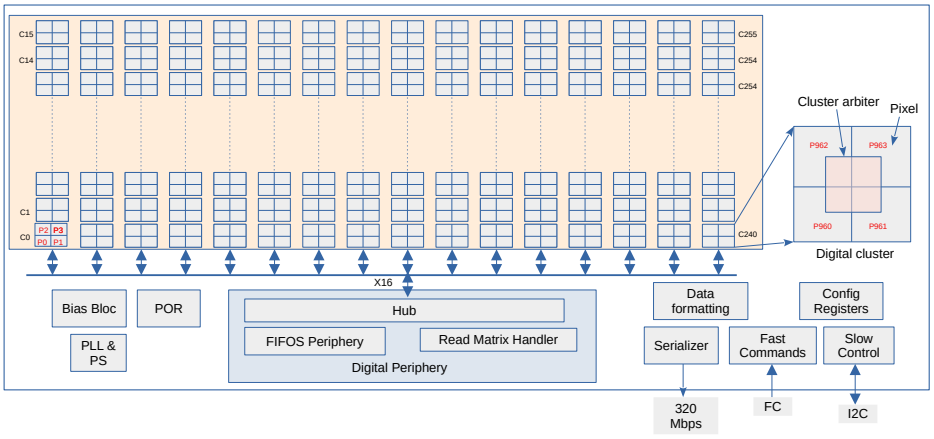


Figure 1.1: Block diagram of the EICROC digital readout

The digital readout contains 1024 identical single pixel readout channels organized in 256 clusters of 4 pixels each. The data of each pixel are first transferred to the cluster arbiter, and then sent to the End of Column before being sent to the digital periphery and subsequent formatting blocks where they are formatted and serialized, and later sent to the post-processing. Within a cluster, in each of the four pixels, when the useful data is available, it is stored in a register before being sent to the cluster arbiter. When

the data of the four pixels is available, and the cluster is selected for reading, the active pixels are sent sequentially to the digital periphery. Each hit event is indexed by a BCID(Bunch crossing ID) counter value. When all the clusters related to an event are read, the digital periphery jumps to the next BCID.
The digital readout contains the following specific functional blocks:

- The matrix of the pixels clusters: It is composed of 256 clusters of 4 pixels each organized in 16 columns each containing 16 clusters and ending in EOC termination. The cluster arbiter reads the active pixels information and sends it to the digital periphery when it is addressed for reading. After each bunch crossing, a 256-bits busy word is generated by the clusters matrix and sent directly to the digital periphery in order to be associated with the corresponding Bunch Crossing ID (BCID).

- The digital periphery: It manages the busy word–BCID association, generates the read addresses for the active clusters, retrieves the pixels information, and packages the readout matrix data into a frame. It is composed of three sub modules : A hub, a periphery FIFOs sub-block and a read matrix handler sub-module.

- A slow control block receives the slow control parameters sent by I^2C and sends them into a configuration register.

- A Fast Command Unit receives and decodes the fast commands that are sent by the LpGBT along with the 320 MHz clock.

- A phase shifter and a PLL provide the time measurement clock (40 MHz).

- Specific blocks are used to provide the bias voltages, the monitoring information and a Power On Reset signal.

Chapter 2

Cluster Digital Implementation

2.1 Cluster digital top

The matrix of the pixels clusters is composed of 256 clusters. Each cluster is composed of four pixels digital parts that share a common cluster arbiter that manages the readout of the four pixels figure 2.1. The ready/valid handshake protocol is used as a base for synchronizing transfer from pixels to their cluster arbiter and then to the digital periphery. The vertical arrangement of cluster is shown in figure 2.2.

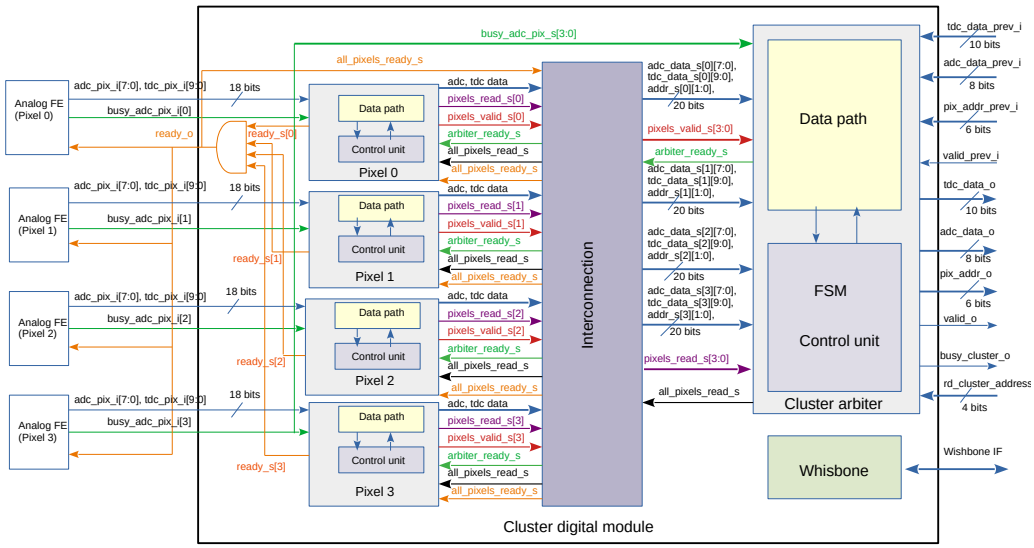


Figure 2.1: Block diagram of the digital cluster hierarchy

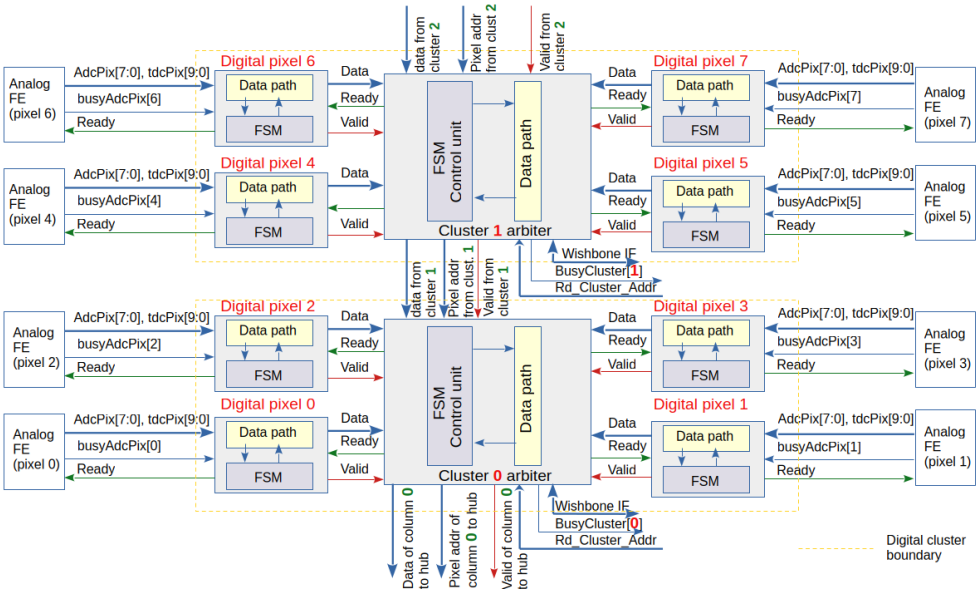


Figure 2.2: Example of clusters vertical arrangement (here clusters 0 and 1) in a chain inside a column (here column 0)

2.1.1 Cluster digital top pinout

Table 2.1: Inputs/Outputs of the digital cluster

Signal	Dir.	Width	Description
clk	IN	1 bits	40 MHz Clock
rst	IN	1 bits	Rest. Active low
cluster_addr_i	IN	4 bits	Cluster internal address. It is hard wired at the top level using TIE_HIGH and TIE_LOW signals to configure the internal address of the cluster.
read_enable_i	IN	1 bits	Read enable from periphery
busy_adc_pix_i	IN	4 x 1 bits	High level on the signal of index i indicates that the ADC with index i is digitizing the data. The rising edge indicates the start of the digitizing, the falling edge indicates the conversion is finished and ADC data is ready to be stored.

busy_tdc_pix_i	IN	4 x 1 bits	High level on the signal of index i indicates that the TDC with index i is digitizing the data. The rising edge indicates the start of the digitizing, the falling edge indicates the conversion is finished and ADC data is ready to be stored.
adc_pix_i	IN	4 x 8 bits	The energy digital data from ADCs
tdc_pix_i	IN	4 x 10 bits	The timing digital data from TDCs
ready_o	OUT	1 bit	Contains the ready signal sent to the ADC/TDC. In order to block the 3 other pixels whenever a pixel is not ready, the ready_o port that is sent to each ADC is the "and" gating of all the other ready_s[i] signals. The signal ready_s[i] of index i contains the ready from pixel with index i to its corresponding ADC/TDC. **ready_s[i] = 1 :** The pixel sub-block with index i is ready to receive new ADC/TDC data from ADC/TDC with index i. **ready_s[i] = 0 :** The pixel sub-block with index i is not yet ready to receive new ADC/TDC data from ADC/TDC with index i.
rd_cluster_address_i	IN	4 bits	The cluster address from the hub sub-module of the digital periphery. If this address matches the cluster's internal address, the current cluster data is sent to the column bus, otherwise the precedent cluster's data (adc_data_prev_i, tdc_data_prev_i, pix_addr_prev_i) is sent

adc_data_prev_i	IN	8 bits	The ADC energy digital data from the preceding cluster in the column. This data is selected by the arbiter output MUX when the cluster address sent by the EOC (Rd_cluster_address) is different from the cluster internal address
tdc_data_prev_i	IN	10 bits	The TDC timing digital data from the preceding cluster in the column. This data is selected by the arbiter output MUX when the cluster address sent by the EOC (Rd_cluster_address) is different from the cluster internal address.
pix_addr_prev_i	IN	10 bits	The cluster address from the preceding cluster in the column. This output address is selected by the arbiter output MUX when the cluster address sent by the EOC (Rd_cluster_address) is different from the cluster internal address.
valid_o	OUT	1 bit	The valid signal for the cluster data.
adc_data_o	OUT	8 bits	ADC Digital data that is sent to the EOC.
tdc_data_o	OUT	10 bits	TDC Digital data that is sent to the EOC.
pix_addr_o	OUT	10 bits	Pixel Address that is sent to the EOC.
wb_stb_i	IN	1 bits	
wb_wen_i	IN	1 bits	
wb_addr_i	IN	9 bits	
wb_data_i	IN	8 bits	
wb_dat_prev_i	IN	8 bits	
wb_ack_prev_i	IN	1 bits	
wb_dat_next_o	OUT	8 bits	
wb_ack_next_o	OUT	1 bits	

2.2 Pixel sub-bloc

Each cluster contains four identical pixel sub-blocks. Each pixel sub-block communicates with its corresponding ADC and TDC, and stores the related timing and energy information, that are TDC and ADC data, respectively. Each pixel uses two simple registers to store this data: one 10-bit register for TDC and another 8-bit register for ADC. The pixel sub-block uses a handshake-based protocol employing ready/valid signals. When it receives a rising edge of the busy signal from the ADC/TDC, the pixel sub-block waits for the ADC to finish in order to store the digitized ADC/TDC words, since the ADC output value is available late after the TDC one. When ADC/TDC word is stored in the pixel registers, the pixel sets its valid signal (pixel_valid_o) and waits for the ready signal (arbiter_ready_o) from the cluster arbiter in order to send its ADC/TDC data to the cluster arbiter's registers. Once the pixel data is read by the cluster arbiter, the pixel can send a ready signal to ADC/TDC (an and gate is performed on all the 4 ready_s signals from the 4 pixels) meaning it is ready to store new data while the previous data is waiting to be sent by the cluster to the digital periphery.
The pixel sub-block is composed of a control unit and a data path.

2.2.1 Pixel Sub-Block Pinout

Table 2.2: Inputs/Outputs of the pixel sub-block

Signal	**Direction**	**Width**	**Description**
clk	IN	1 bits	40 MHz Clock
rst	IN	1 bits	Rest. Active low
busy_adc_pix_i	IN	1 bits	High level on this signal indicates that the ADC is digitizing the data. The rising edge indicates the start of the digitizing, the falling edge indicates the conversion is finished and ADC data is ready to be stored.

arbiter_ready_i	IN	bit	Indicates that the cluster arbiter is ready to receive new data. This signal plays the role of the "Ready" function from the cluster arbiter.
all_pixels_read_i	IN	bit	Indicates that the cluster arbiter has received data from the four pixels. The pixel FSM stays in the READ state until all the other pixels are read which is indicated by receiving '1' on the all_pixels_read_i input.
all_pixels_ready_i	IN	bit	Indicates that the all the pixels have finished and are ready to receive new ADC/TDC data. It is an "and" gating of the four ready signals from pixels to ADC/TDCs. It is generated inside the cluster digital top module. On any pixel, a new busy is sampled only if this input is equal to '1'.
adc_pix_i	IN	8 bits	The energy digital data from ADC
tdc_pix_i	IN	10 bits	The timing digital data from TDC
pixel_ready_o	OUT	1 bit	The ready signal from Pixel sub-block to the ADC/TDC. **pixel_ready_o = 1 :** The pixel sub-block is ready to receive new ADC/TDC data from ADC/TDC. **pixel_ready_o = 0 :** The pixel sub-block is not ready to receive new ADC/TDC data from ADC/TDC.
pixel_valid_o	OUT	1 bit	The valid signal from Pixel sub-block to the cluster arbiter. **pixel_valid_o = 1 :** The data at the pixel sub-block output is valid. **pixel_valid_o = 0 :** The data at the pixel sub-block output is not valid.

pixel_read_o	OUT	1 bit	Indicates that the pixel data has been retrieved by the cluster arbiter. **pixel_read_o = 1 :** The pixel data has been read by the cluster arbiter. **pixel_read_o = 0 :** The pixel data is waiting to be read by the cluster arbiter.
adc_data_o	OUT	8 bits	ADC Digital data that is sent to the cluster arbiter.
tdc_data_o	OUT	10 bits	TDC Digital data that is sent to the cluster arbiter.
addr_pix_o	OUT	2 bits	Pixel Address that is sent to the cluster arbiter.

2.2.2 Ready/Valid handshake protocol

In order to assure synchronization between data source and destination from ADC/TDC until transmission to the digital periphery, an AMBA (AMBA AXI and ACE Protocol Specifications, Issue D, ARM October 2011, IHI 0022D) based protocol Ready/Valid is used as in figure 2.3.

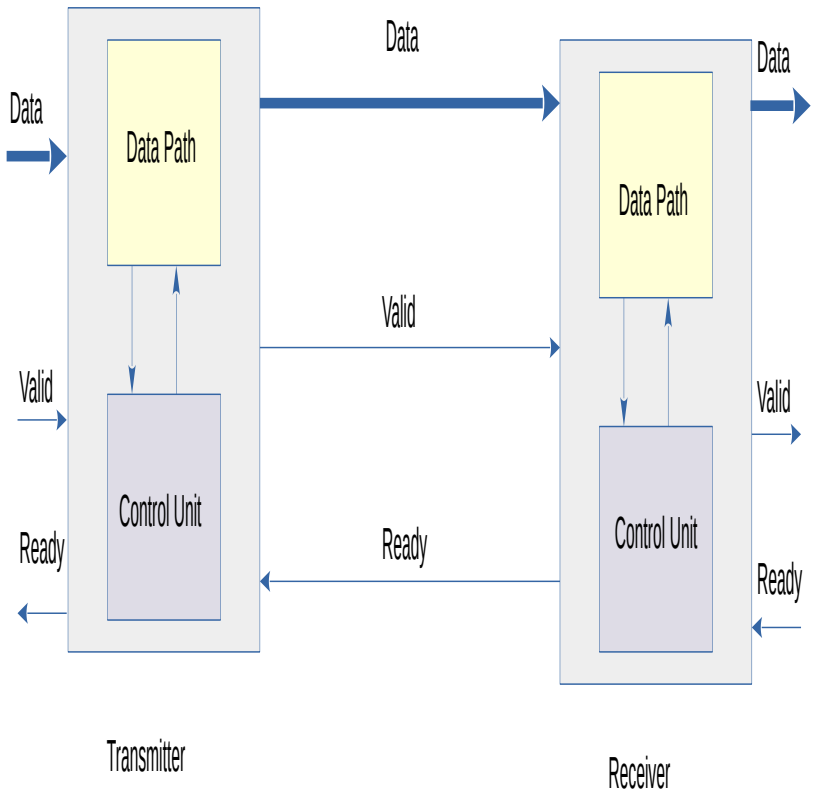


Figure 2.3: Transmitter receiver interface

Table 2.3: Ready/Valid cases

Etat	**Ready**	**Valid**	**Description**
Idle	0	0	Transmitter data is not valid. The receiver is by default ready to receive a data. It set its ready signal to '1'.

Waiting for Ready	0	1	The transmitter holds a valid data but the receiver is not ready to receive it. Data is not transferred. This is the case of data "data1" that must be held another clock cycle
Waiting for Valid	1	0	The receiver is ready to receive a new data but the transmitter doesn't have a data available yet. This is the case of data "data2" in the figure 2.4.
Transfer	1	1	The handshake occurs and the transfer is performed on the rising edge of the clock.

According to AMBA specification two rules must be respected :

- When new data is available, the transmitter must set valid to 1 as soon as possible, without waiting for the receiver to be ready.
- When valid is set to 1, the transmitter is not allowed to set it back to 0 before the handshake is completed. The handshaking occurs at a rising edge where valid and ready are both equal to 1.

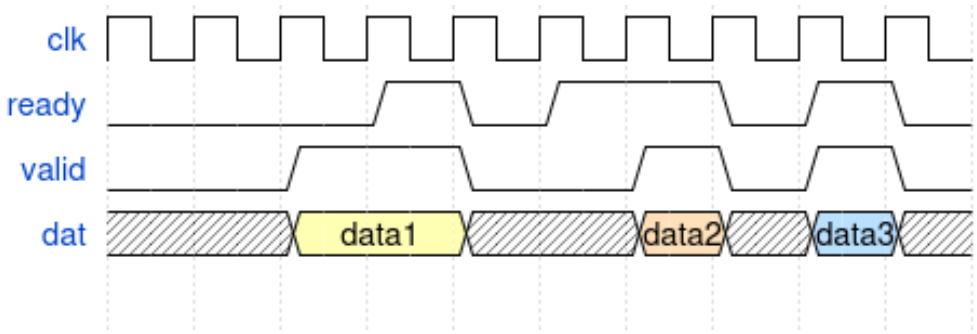


Figure 2.4: Ready/Valid handshake mechanism

2.2.3 Pixel sub-block Control Unit

This part of the pixel sub-block constitutes the control unit, which contains the pixel FSM that generates control signals to manage ADC and

TDC pixel data handling. The generated control signals are typically ready and valid signals for the cluster arbiter and ADC/TDC, respectively, in addition to load and reset flags of the registers and counters of the pixels data paths. The pixel's sub-block control unit implements a Moore Finite State Machine based on the Algorithmic State Machine (ASM) shown in the figure 2.5. The ASM is a chart describing the algorithmic functionality of the control unit. Rectangles represent states. In Mealy types, rectangles with rounded corners represent conditional outputs. The names inside rectangle, represent signals that get the '1' value by the state. Diamonds represent decision boxes. From the figure 2.5, the pixel control unit FSM has four possible states, pixel_ready_st, pixel_busy_st, pixel_valid_st and pixel_read_st. It starts by setting the ready signal to '1'. Once a busy signal (rising edge of the busy_adc_pix_i signal that is busy_adc_pix_re) is received from the ADC/TDC, the pixel enters the pixel_busy_st state and waits for the falling edge of the busy_adc_pixı(busy_adc_pix_fe). Once the busy falling edge is received, the pixel FSM enters the pixel_valid_st state where it sets the valid signal (pixel_valid_o) and latches the ADC and TDC data in the corresponding registers, that are adc_reg_q and tdc_reg_q. The pixel FSM waits in the pixel_valid_st state until it receives the ready input (Arbiter_Ready) from the cluster arbiter. Once in the pixel_read_st state, the FSM sets the flag pixel_read_o and stays in this state until it receives the all_pixels_read_i acknowledge from the cluster arbiter which indicates that the four pixels were read. In this way the four pixels of the cluster stay blocked if any of these four pixels has not been read yet by the cluster arbiter.

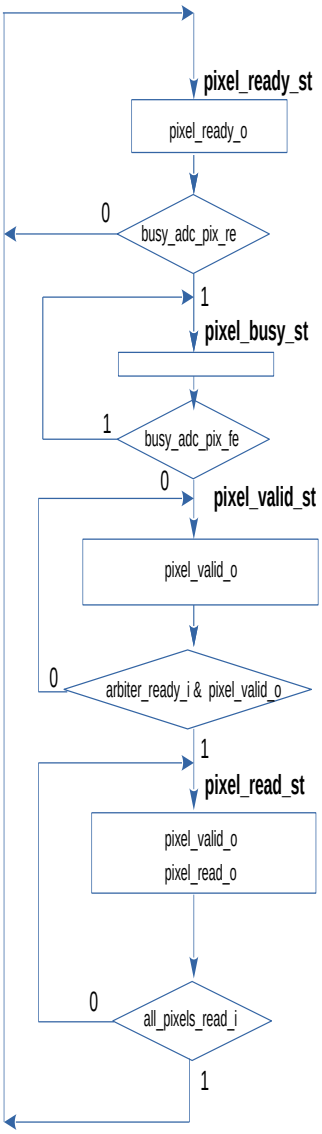
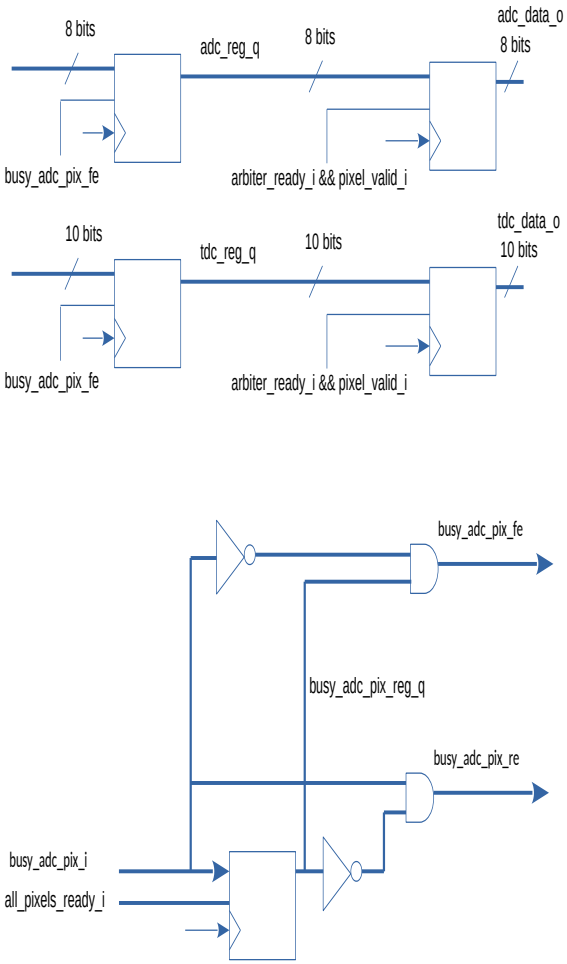


Figure 2.5: Algorithmic State Machine of the pixel sub-bloc

2.2.4 Pixel sub-block Data Path

This part is composed of the dedicated logic to store and dispatch the data (figure 2.6). It contains typically the registers, muxes, counters, etc. In order to ensure good Ready/Valid implementation, and a correct pipelining, a double register buffering is performed inside the pixel data path. The pixel double-registers are (adc_reg_q (8 bits), adc_data_o(8 bits)) and (tdc_reg_q(10 bits), tdc_data_o(10 bits)) that are ADC energy data and TDC timing data respectively. The pixel data path contains also the busy_adc_pix_reg register which is reset by the global rst signal and loaded by the all_pixels_ready_i input of the pixel control unit.



Pixel data path

Register adc_reg_q : ADC (8 bits) : reset : rst, Load : busy_adc_pix_fe

Register tdc_reg_q : TDC (10 bits) reset : rst, Load : busy_adc_pix_fe

Register adc_data_o : ADC (8 bits) : reset : rst, Load : arbiter_ready_i && pixel_valid_i

Register tdc_data_o : TDC (10 bits) reset : rst, Load : arbiter_ready_i && pixel_valid_i

busy_adc_pix_reg_q : 1 bit, reset : rst, Load : all_pixels_ready_i

Figure 2.6: Data path of the pixel sub-bloc

2.3 Cluster arbiter sub-bloc

Same as for the pixel sub-module, the cluster arbiter is composed of a control unit and a data path unit.

2.3.1 Cluster arbiter Pinout

Table 2.4: Inputs/Outputs of the cluster arbiter sub-block

Signal	Dir.	Width	Description
clk	IN	1 bits	40 MHz Clock
rst	IN	1 bits	Rest. Active low
cluster_addr_i	IN	4 bits	Cluster internal address. It is hard wired at the top level using TIE_HIGH and TIE_LOW signals to configure the internal address of the cluster.
read_enable_i	IN	1 bits	Read enable from periphery
rd_cluster_address_i	IN	4 bits	Contains the cluster address sent by the hub sub-module of the digital periphery to select the cluster being read.
pixels_valid_i	IN	4 bits	Contains the Valid signal of the four pixels. **pixels_valid_i[i] = 1 :** The pixel with index i has valid data available for reading by the cluster arbiter. **pixels_valid_i[j] = 0 :** The pixel with index j has not valid data available yet.
pixels_read_i	IN	4 bits	Contains the read status information of the four pixels. **pixels_read_i[j] = 1 :** The pixel with index j has been read by the cluster arbiter. **pixels_read_i[j] = 0 :** The pixel with index j has not been read yet.

busy_adc_pix_{j}_i	IN	1 bit	busy signal from ADC corresponding to pixel with index j. **busy_adc_pix_{j}_i = 1 :** The ADC is digitizing the data. The rising edge indicates the start of the digitizing. **busy_adc_pix_{j}_i = 0 :** The conversion is finished and ADC data is ready to be stored. The falling edge indicates the finish of the digitizing.
adc_pix_{j}_i	IN	8 bits	The energy digital data from ADC of the pixel with index j.
tdc_pix_{j}_i	IN	10 bits	The timing digital data from TDC of the pixel with index j.
addr_pix_{j}_i	IN	2 bits	The address of the pixel with index j.
adc_pix_prev_i	IN	8 bits	The ADC energy digital data from the previous cluster. This data is selected by the arbiter output MUX when the cluster address sent by the EOC (Rd_cluster_address) is different from the cluster internal address
tdc_pix_prev_i	IN	10 bits	The TDC timing digital data from the previous cluster. This data is selected by the arbiter output MUX when the cluster address sent by the periphery (rd_cluster_address_i) is different from the cluster internal address.
addr_prev_i	IN	6 bits	The Cluster address from the previous cluster. This output address is selected by the arbiter output MUX when the cluster address sent by the periphery (rd_cluster_address_i) is different from the cluster internal address(cluster_addr_i).
cluster_valid_prev_i	IN	1 bit	The valid signal from the precedent cluster in the column.
valid_o	OUT	1 bit	The valid signal for the cluster data.
adc_reg_o	OUT	8 bits	ADC Digital data that is sent to the periphery.

tdc_reg_o	OUT	10 bits	TDC Digital data that is sent to the periphery.
addr_pix_o	OUT	6 bits	Pixel address that is sent to the periphery.
arbiter_ready_o	OUT	1 bit	Ready signal from the arbiter cluster sent to each pixel informing that the cluster arbiter is ready to receive new data from the pixel.
all_pixels_read_o	OUT	1 bit	Acknowledge sent to each pixel informing that all the other pixels were read by the cluster arbiter.

2.3.2 Cluster arbiter control unit

This part of the cluster arbiter sub-block constitutes the control unit. It contains the cluster arbiter FSM that generates control signals to manage pixel arbitration and data sending to the digital periphery. The FSM generates the control signals that are "Ready" and "Valid" signals for the pixels and the digital periphery, respectively, in addition to load and reset flags of the registers and counters of the cluster data path. It implements a Mealy Finite State Machine based on the Algorithmic State Machine (ASM) as shown in the figure 2.7. From the figure 2.7, the cluster arbiter control unit passes by four possible states, idle_st, wait_all_pixels_st, load_pixels_st, bus_busy_st and send_pixel_st.
The cluster arbiter sub-block FSM starts by setting the Arbiter_Ready signal. Once a valid data is present on a pixel (Pixels_Valid[3:0] bus is non-null), the cluster arbiter enters the wait_all_pixels_st. In this state, the arbiter waits for all the active pixels in the cluster to finish receiving data from the ADC/TDC. In this state, the arbiter reads the Pixels_Valid bus value and compares it to the busy word (busy_word_i). Once the pixels_valid_i is equal to the vector busy_word_i, which means that all the hit pixels have finished, the FSM goes to the state load_pixels_st. When in load_pixels_st state, and depending on the inputs values (pixels_read_i and read_request_i), the FSM goes to the send_pixel_st state, if a read request is received from the periphery, or to the bus_busy_st state, if all the hit pixels data are available. In bus_busy_st state the cluster data is available and the cluster arbiter waits for "read enable" from he digital periphery which occurs when rd_cluster_address_i matches the cluster internal address clus-

ter_addr_i. When this condition is met, the FSM enters the send_pixel_st state state, where the cluster arbiter sets the cluster_valid_o signal and starts to send out the ADC/TDC/ADDR 28-bits data of the four pixels sequentially, one pixel after the other and increments the cnt_i counter. When the number of output pixels equals 4 (cnt_i = 3), the FSM returns back to to the wait_all_pixels_st state if it receives a new data from a pixel (pixels_valid_i is not null) or to the idle_st if no new ADC/TDC data is available from pixels (pixels_valid_i is null).

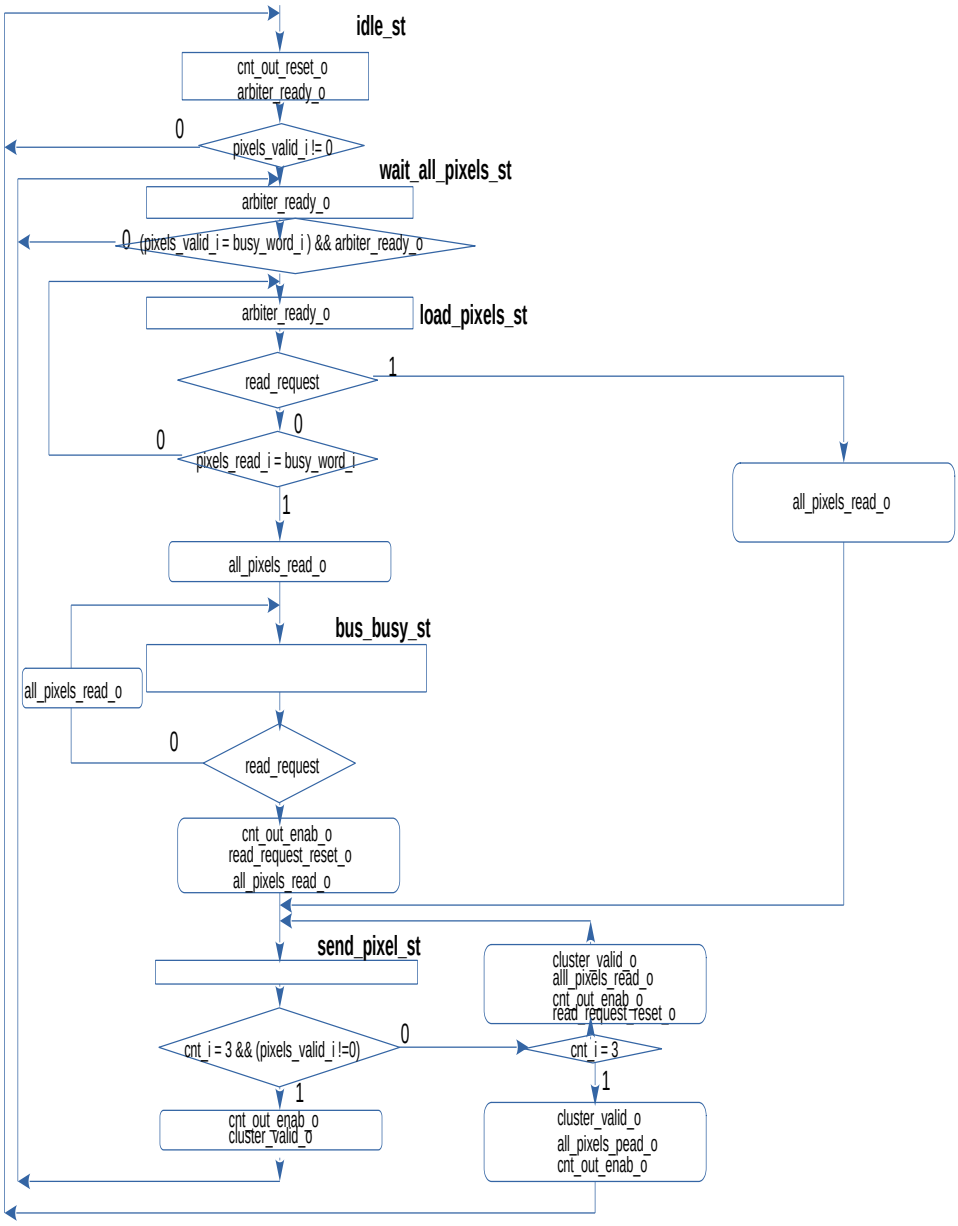


Figure 2.7: Algorithmic State Machine of the cluster arbiter sub-bloc

2.3.3 Cluster arbiter data path

This part is composed of the dedicated logic to store and dispatch the data as depicted in figure 2.8. The data path content is described in table 2.5.

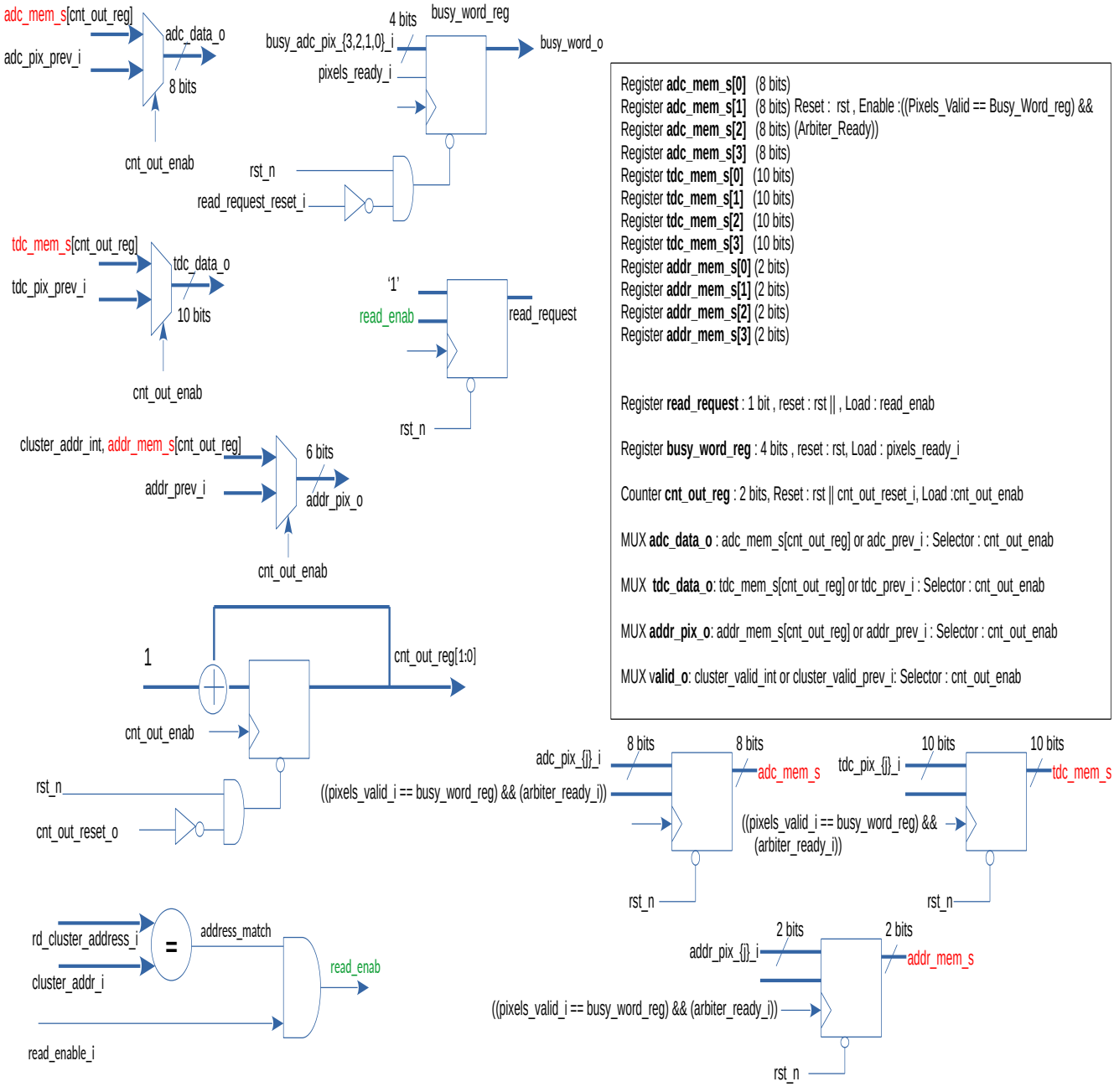


Figure 2.8: Cluster arbiter data path

Table 2.5: Data path of the cluster arbiter

Register name	Width	Rest signal	Load signal	Function	Description
adc_mem_s[0]	8 bits	rst_n		Register	ADC data from Pixel 0
adc_mem_s[1]	8 bits	rst_n		Register	ADC data from Pixel 1
adc_mem_s[2]	8 bits	rst_n		Register	ADC data from Pixel 2
adc_mem_s[3]	8 bits	rst_n		Register	ADC data from Pixel 3
tdc_mem_s[0]	10 bits	rst_n		Register	TDC data from Pixel 0
tdc_mem_s[1]	10 bits	rst_n		Register	TDC data from Pixel 1
tdc_mem_s[2]	10 bits	rst_n		Register	TDC data from Pixel 2
tdc_mem_s[3]	10 bits	rst_n		Register	TDC data from Pixel 3
addr_mem_s[0]	2 bits	rst_n		Register	ADDR of Pixel 0
addr_mem_s[1]	2 bits	rst_n		Register	ADDR of Pixel 1
addr_mem_s[2]	2 bits	rst_n		Register	ADDR of Pixel 2
addr_mem_s[3]	2 bits	rst_n		Register	ADDR of Pixel 3
read_request_o	1 bits	rst_n	'1'	Register	This flag gets the '1' value when there is a high value on read_enable_i input and equality between the cluster internal address (cluster_address_i) and the received read address from periphery (rd_cluster_address_i).

busy_word_reg	4 bits	rst_n	pixels_ready_i	Register	The value of the busy_word_o output formed by the busy_adc_pix_{j}_i inputs of the 4 pixels (j=0,1,2,3). The busy_word_o output is sent to the cluster arbiter control unit (FSM).
cnt_out_reg	2 bits	rst_n or cnt_out_reset_i	cnt_out_enab_i	Counter	Number of the pixel being sent by the cluster arbiter to the digital periphery.
adc_data_o	8 bits			MUX	Selects the output of the cluster depending on the cnt_out_enab_i value. **cnt_out_enab_i = 1 :** adc_data_o contains the ADC data of the current cluster. **cnt_out_enab_i = 0 :** adc_data_o contains the ADC data of the previous cluster.

tdc_data_o	10 bits			MUX	Selects the output of the cluster depending on the cnt_out_enab_i value. **cnt_out_enab_i = 1 :** tdc_data_o contains the TDC data of the current cluster. **cnt_out_enab_i = 0 :** tdc_data_o contains the TDC data of the previous cluster.
addr_pix_o	10 bits			MUX	Selects the output of the cluster depending on the cnt_out_enab_i value. **cnt_out_enab_i = 1 :** addr_pix_o contains the address of the current pixel. **cnt_out_enab_i = 0 :** addr_pix_o contains the address from the previous cluster.

cluster_valid_o	10 bits			MUX	Selects the output of the cluster depending on the cnt_out_enab_i value. **cnt_out_enab_i = 1 :** cluster_valid_o contains the "Valid" flag of the current addressed cluster. **cnt_out_enab_i = 0 :** cluster_valid_o contains the "Valid" signal from the precedent cluster (cluster_valid_prev_i).

2.4 Pipelining

The usage of the ready/valid handshake mechanism for data transfer between the pixels and their clusters arbiter enables throughput optimization and dead time reduction. Indeed, the pixels can accept new data while the cluster arbiter is still sending the previous pixels data. This feature is useful in case of double hit, where two close consecutive hits strike a cluster. That is the four pixels of a cluster are blocked only when any of the pixels is still waiting for the source to transmit data. This happens when the late pixel FSM is still in the pixel_busy_st state. The other waiting pixels FSM are waiting in the read_pixel_st state. Once all the hit pixels finished receiving the data (when they all receive the all_pixels_read_i) flag, they can immediately enter the pixel_ready_st state meaning they are ready to receive a new hit by rising back the output ready_o. The cluster arbiter may be still waiting for a read request from the digital periphery in order to send the data of the previous hit. The digital dead-time is then equal to the time where the ready_o signal from the cluster to the ADC/TDC is

low. As show in figure 2.9, the pixel enters the pixel_ready_st state and the ready signal of the pixels (pixel_ready_o) goes high while the arbiter is still in the bus_busy_st state and its ready signal (arbiter_ready_o) is low. In this way even if the arbiter is still busy, the pixels can get a new hit which is illustrated by the busy_adc_pix_i rising edge when the pixel_ready_o is high. The digital dead time , which is the delay between the falling edge of busy_adc_pix_i and the rising edge of pixel_ready_o is hence reduced to less than 4 cycles.

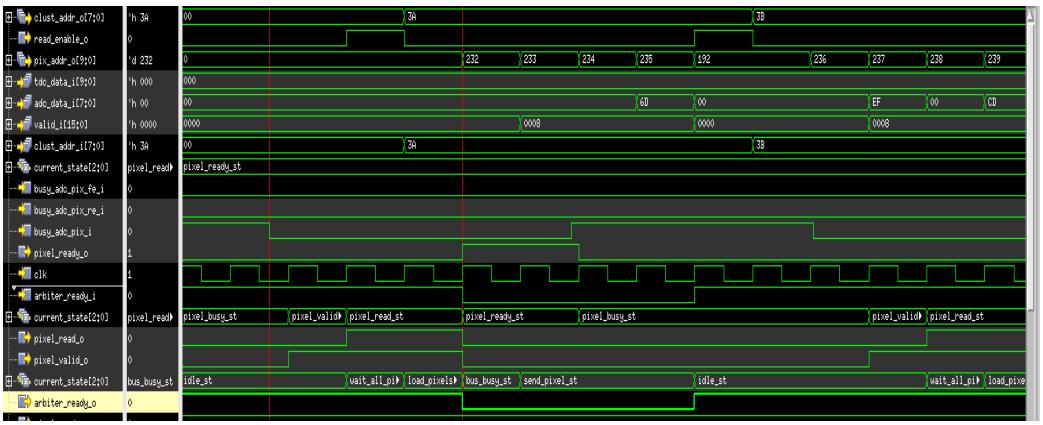


Figure 2.9: Double hit

Chapter 3

Digital Periphery

3.1 Digital Periphery

The digital periphery (figure 3.1) operates at 40 MHz. It generates read requests to the clusters through clusters address (8 bits) generation and accepts valid data from clusters through a synchronous handshake. Each pixel buffers its own hits and waits for the corresponding cluster arbiter to be ready to accept the pixels data. If the cluster readiness is delayed, it blocks further pixels data acquisition. This strategy effectively prevents overflows without losing synchronization. The digital periphery is responsible for clusters' address decoding based upon the busy signal received from the clusters matrix. It is composed of a hub, a fifo periphery and a read matrix handler.

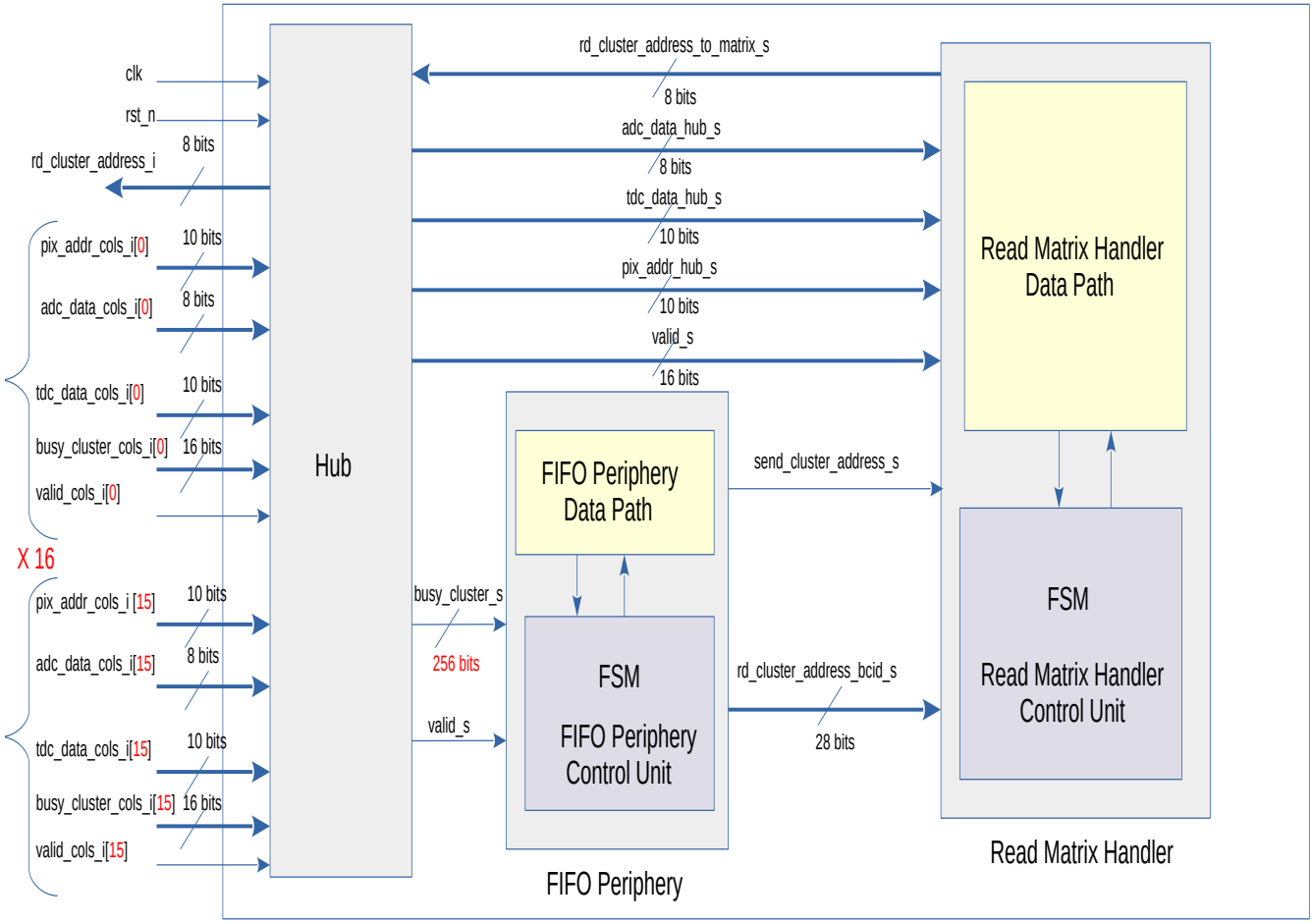


Figure 3.1: Block diagram of the digital periphery

3.1.1 Digital periphery pinout

Table 3.1: Inputs/Outputs of the digital periphery

Signal	Direction	Width	Description
clk	IN	1 bits	40 MHz Clock
rst_n	IN	1 bits	Rest. Active low
read_enable_o	OUT	1 bits	Read enable signal to the clusters. It is generated by the FSM of Read_Matrix_Handler module. Its value is equal to signal latch_periphary_fifo_clust_addr_s
rd_cluster_address_o	OUT	8 bits	The address of the cluster to read. If this address matches the cluster internal address, the current cluster data is sent to the column bus, otherwise the precedent cluster's data is sent
pix_addr_cols_i	IN	Array of 10-bits x 16 columns	Address bus
tdc_data_cols_i	IN	Array of 10-bits x 16 columns	TDC data
adc_data_cols_i	IN	Array of 8-bits x 16 columns	ADC data
busy_cluster_cols_i	IN	Array of 16-bits x 16 columns	Busy signal array from the 256 clusters.
valid_cols_i	IN	Array of 1-bits x 16 columns	Valid signals from columns.
wb_stb_i	IN	1 bits	
wb_wen_i	IN	1 bits	
wb_addr_i	IN	8 bits	
wb_data_i	IN	8 bits	
wb_dat_from_columns	IN	8 bits	
wb_ack_from_columns	IN	1 bits	

3.2 Hub sub-block

The hub sub-block (figure 3.1) plays a multiplexer role that generates from the 16 column inputs arrays (adc_data_cols_i[15:0][7:0], tdc_data_cols_i[15:0][9:0] and pix_addr_cols_i[15:0][9:0]) a multiplexed output set (adc_data_i[7:0],tdc_data_i[9:0] and pix_addr_i[9:0]). The multiplexing selection is based on the column address, that is the 4 MSB bits of the rd_cluster_address_to_matrix_s[7:0] bus. It generates also the busy_cluster_s[255:0] and valid_s[15:0] buses which contain the busy signals of all the clusters and the valid signal from each column respectively.

3.3 fifos periphery sub-block

The fifos periphery sub-block performs the decoding of the busy word received from the clusters matrix. Based on the value of this busy word, the address of the active clusters is generated. This sub-block contains two fifos, busy_fifo used to store the rising edges of the clusters' bits of the busy word (busy_cluster_i[255:0] vector) and the address_array_fifo that stores the BCID of the event, the number of hit clusters and the addresses of the active clusters. At each rising edge in one of the bits of the busy_cluster_i, the corresponding rising edges vector of the busy_cluster_i is stored in the busy_fifo fifo. The busy_fifo fifo is then read to generate the busy_word that is used to decode the addresses of the active clusters. The address of a cluster is generated if there was a detection of a rising edge on the corresponding busy_cluster_i bit. In this way, the clusters' addresses are generated and stored sequentially in the 8 bit LSB of the address_array fifo with the corresponding BCID and the number of hit clusters. The clusters' addresses are then sent sequentially to the read matrix handler.
The fifos periphery sub-block is composed of a control unit and data path unit.

3.3.1 Fifos periphery Control Unit

This part of the fifos periphery sub-block constitutes the control unit, which contains the fifos periphery FSM that generates control signals to manage busy_cluster_i vector decoding and fifos (busy_fifo and address_array_fifo) write and read.

It implements a Moore Finite State Machine based on the Algorithmic State Machine (ASM) shown in the figure 3.2. The ASM is a chart describing the algorithmic functionality of the control unit. Rectangles represent states. From the figure 3.2, the FIFOs periphery control unit passes by four possible states, idle_st, read_busy_st, address_array_calc_st and address_decode_st.

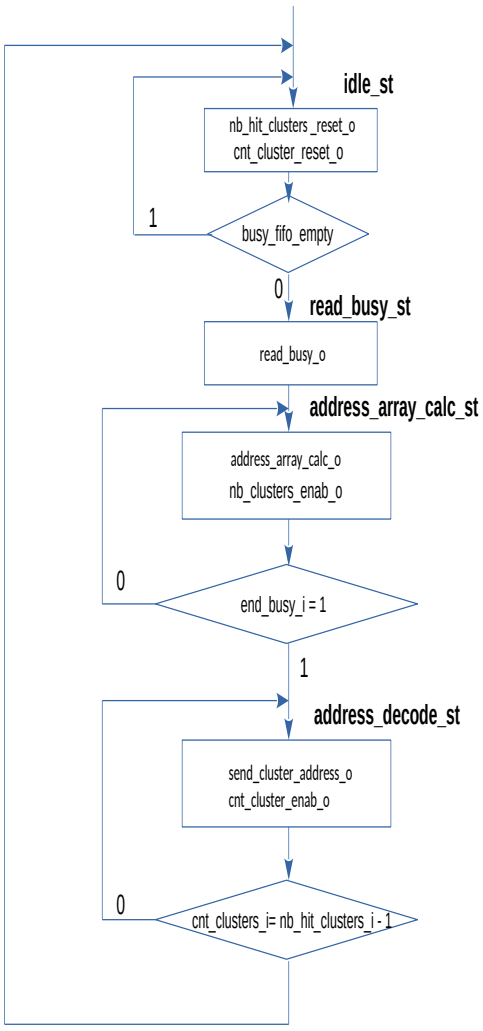


Figure 3.2: ASM chart of the FIFOs periphery sub-bloc

3.3.2 FIFOS periphery Data Path

As shown in figure 3.3, the data path module contains the two fifos (busy_fifo and address_array_s) in addition to a 12-bits register bcid_latched_s and 4 counters that are bcid_s, cnt_clusters_s, cnt_clusters_s, nb_hit_clusters_s_1 and nb_hit_cluster_s. The address is decoded from the busy word using a system verilog task that performs priority encoder and generates at each 40 MHz an address word equal to the position or the index of the cluster that is hit. If a bit of the busy word (busy_word_s[255:0]) , that has index or position i, is equal to '1' than the address of the corresponding cluster is i converted on a 8 bits vector. The busy bit is then reset to interpret the next bit of the busy_word_s vector that is equal to '1'. The empty and full flags of the busy fifo, busy_fifo_empty_s and busy_fifo_full_s, are generated by comparing read and the write pointers. These pointers are both one bit wider than needed to address the fifo. For example, the depth of the busy_fifo is 8 and a pointer of 3 bits is hence sufficient to address all the positions in the fifo. But the read and write pointers , rd_ptr_s and wr_ptr_s, have 4 bits each. The empty flag, busy_fifo_empty_s, goes high when the rd_ptr_s is equal to wr_ptr_s. The full flag goes high when the 3 lsbs of the pointers are equal but the msb bit is different. This same scheme si used also for the read matrix handler block fifos ash shown later in the figure 3.5.

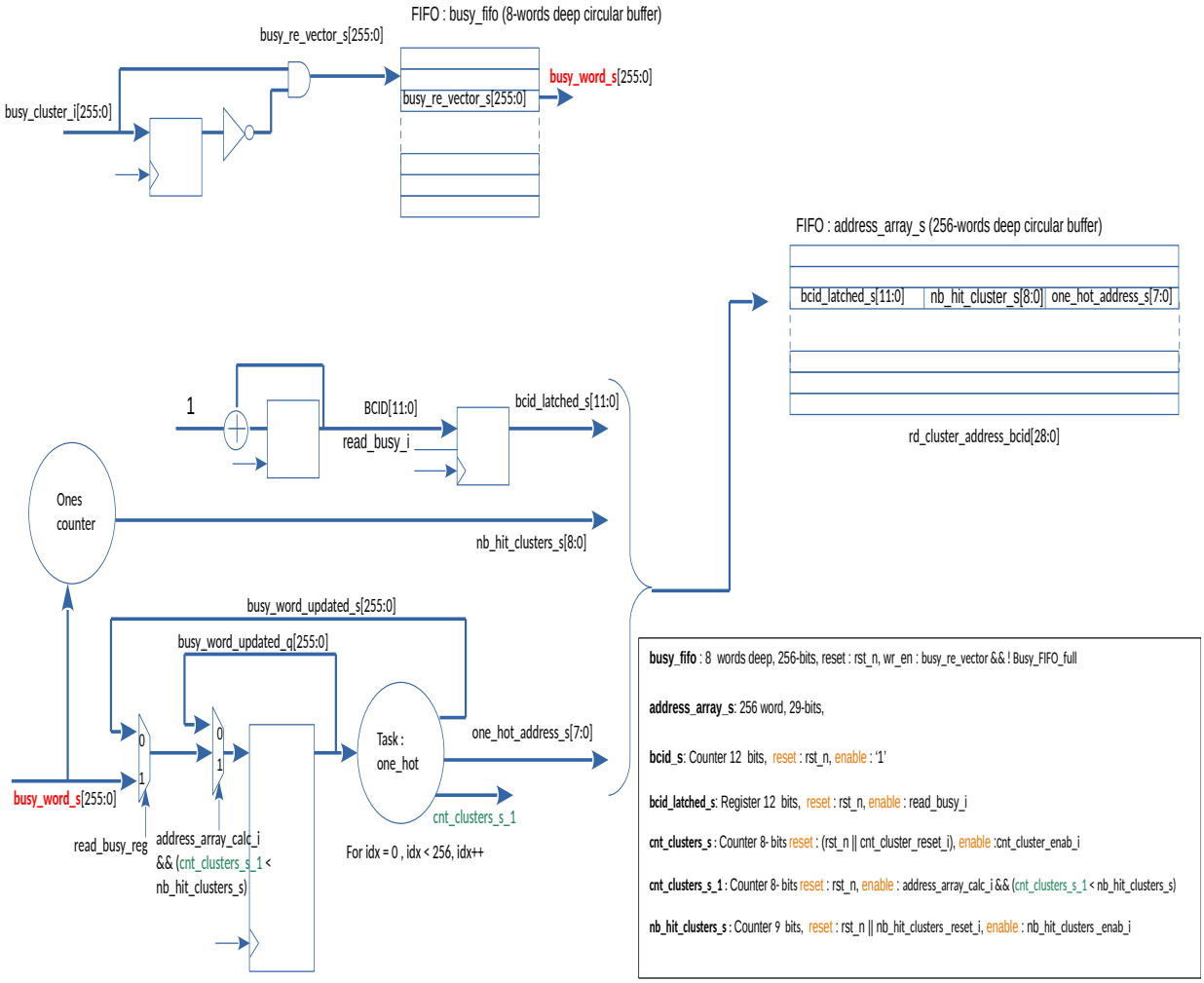


Figure 3.3: Data path of the fifos periphery sub-bloc

3.4 Read Matrix Handler sub-block

The read matrix handler sub-block performs the sending of the addresses of the active clusters that have to be read and fetching and storing the corresponding clusters' data that is adcData, tdcData and pixAddr with the

appropriate BCID counter value. Following the same architecture logic as as the previous modules, this contains a control unit and a data path unit.

3.4.1 Read matrix handler control unit

This part contains the control unit that is responsible for controlling the data path component of the read matrix handler. The functioning algorithm is depicted in the ASM chart shown in figure 3.4

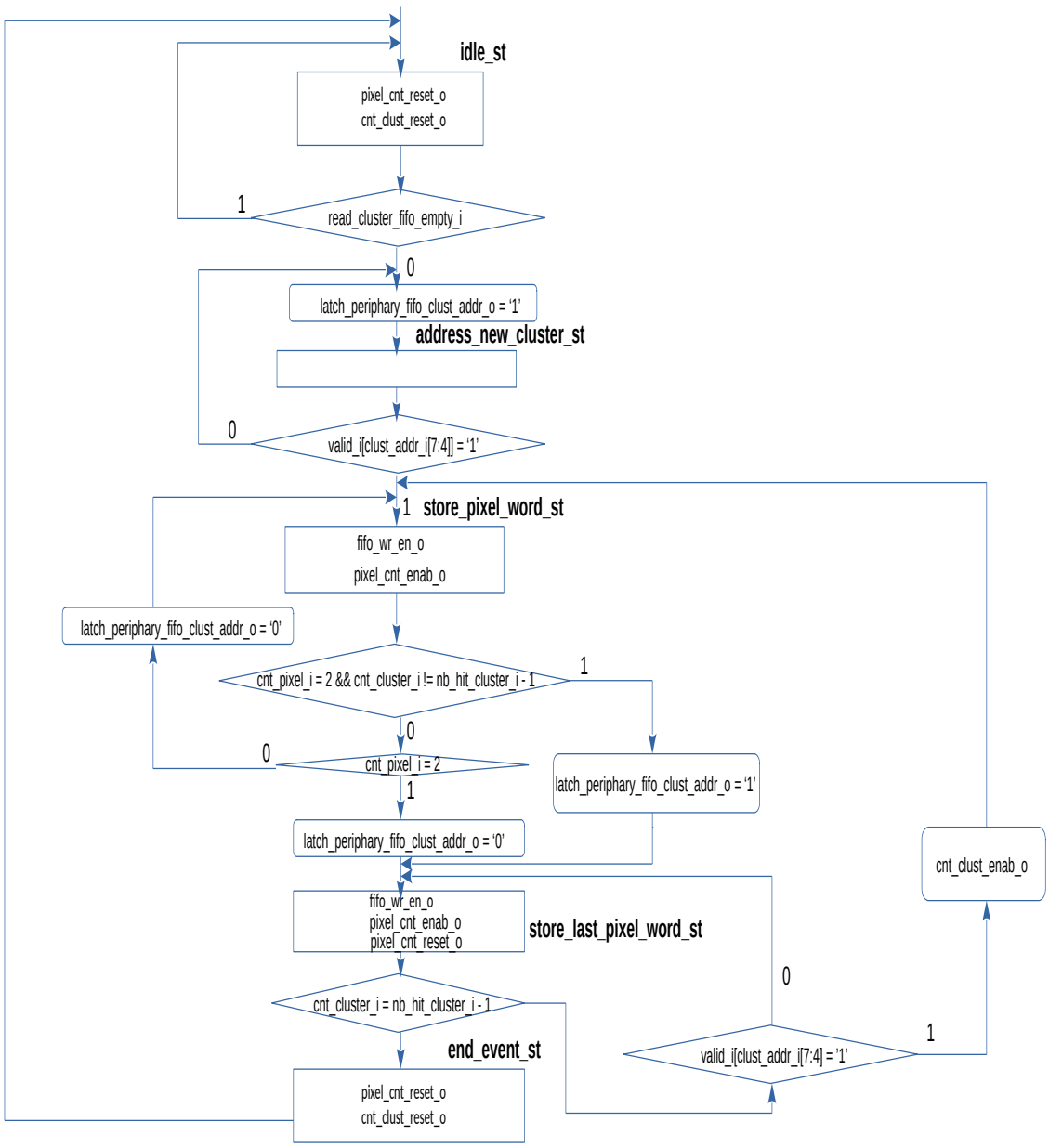


Figure 3.4: ASM chart of the Read_Matrix_Handler Sub-Block

The FSM implemented for this sub-bloc is of a Mealy type in order to optimize the delay between clusters address generation that is useful for the digital dead time minimizing. It is composed of 5 states: idle_st, ad-

dress_new_cluster_st, store_pixel_word_st, store_last_pixel_word_st and end_event_st.

3.4.2 Read matrix handler data path

As shown in figure figure 3.5, the data path of the read matrix handler is composed of FIFOs, registers and counters. The two FIFOs are read_cluster_fifo and pixel_word_fifo. The read_cluster_fifo is used to store the addresses of the clusters to read. The read_cluster_fifo is a 256 depth and 17 bits width FIFO. The FIFO word contains the number of hit clusters and the cluster address. A second fifo, pixel_word_fifo, which is 41-bits wide and 32 word deep, stores the read data (the ADC data adc_data_i[7:0], the TDC data tdc_data_i[9:0] and the pixel address pix_addr_i[9:0]) from the addressed clusters in addition to the BCID of the event which is stored in the 12 MSB bits. The pixel_word_fifo is just a temporary fifo. It will be replaced by outputs to the formatting and serializer modules.

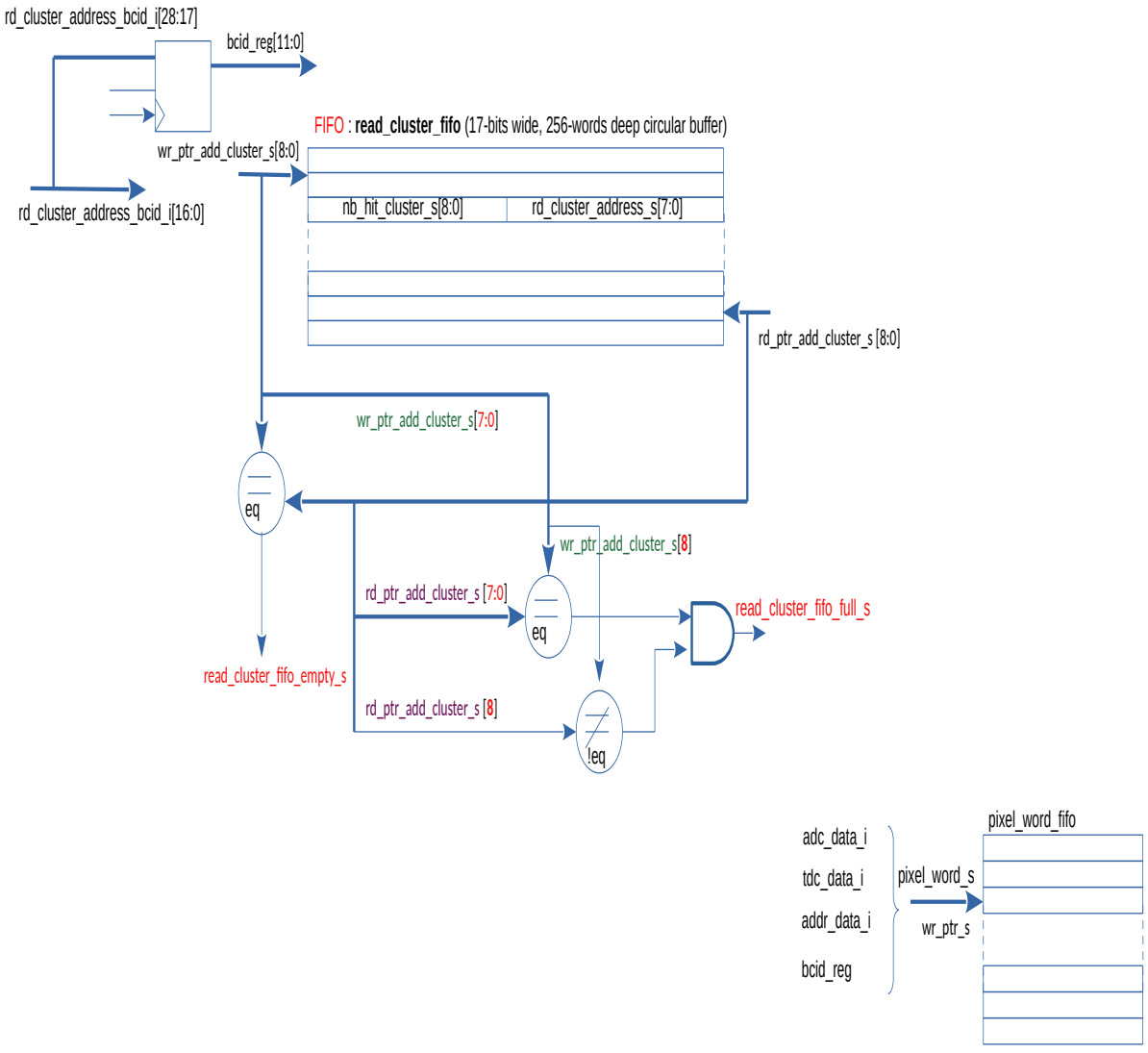


Figure 3.5: Data path of the read matrix handler sub-block