RD-27 Note 30
July 1994

A Coincidence Array Demonstrator ASIC for the RD27 Muon Trigger

John Dowdell, Alan Letchford
Rutherford Appleton Laboratory, Chilton, DIDCOT, OXON, OX11 0QX, UK

Ettore Gennari, Emilio Petrolo, Stefano Veneziano
INFN Roma, 00185 Roma – Piazzale Aldo Moro, 2, ITALY

Nick Ellis
CERN, Geneva, Switzerland

1.0 INTRODUCTION

The coincidence array is a general purpose technique for comparing two vectors and has been used for Muon triggering in several experiments [1,2]. However, its use need not be restricted to the applications discussed in this document alone.

Denote the input vectors as $\underline{X}$ and $\underline{Y}$. $\underline{X}$ has n bits, $\underline{Y}$ has m. The output vector is denoted $\underline{Z}$ and has n bits, the same as $\underline{X}$.

The coincidence array uses an array of bits, $\underline{\underline{A}}$, of dimension n by m, to specify which coincidences between $\underline{X}$ and $\underline{Y}$ are 'valid'.

The value of the coefficient $\underline{\underline{A}}_{ij}$ determines the response of the array when inputs $\underline{X}_i$ and $\underline{Y}_j$ are both '1' simultaneously. If the coefficient is '1' then this is a valid combination and the corresponding $\underline{Z}_i$ output will be '1'. If the coefficient is zero then that combination is not valid and the $\underline{Z}_i$ output depends on the other combinations of the same $\underline{X}_i$. If no combination of $\underline{X}_i$ with $\underline{Y}$ is valid then the output $\underline{Z}_i$ is '0'.

Figure 1.1 is a diagram of a generic coincidence array.

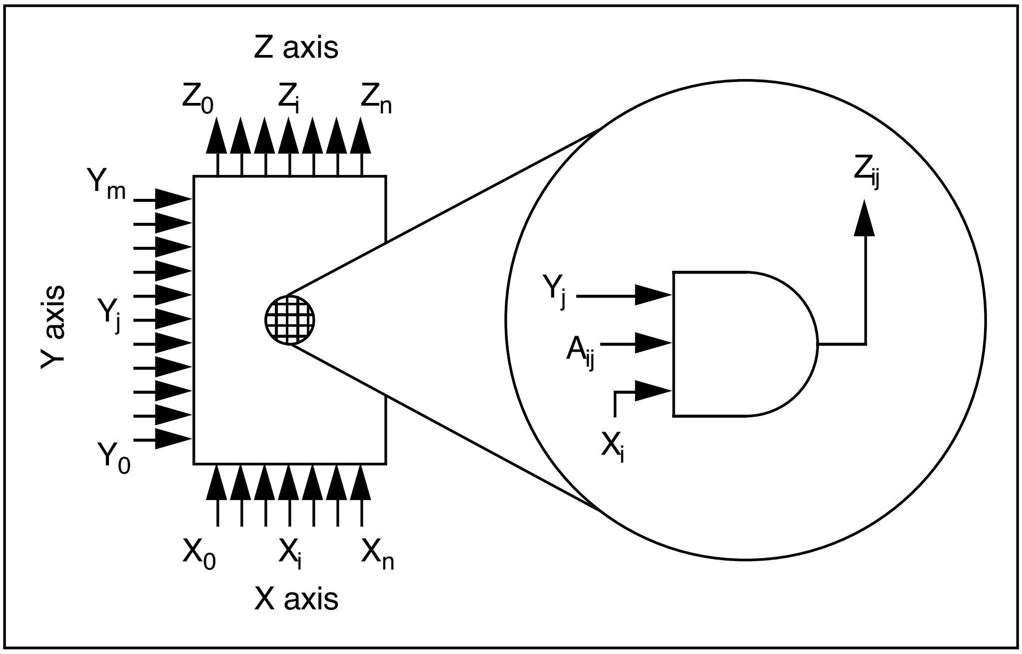

Figure 1.1 - A Generic Coincidence Array

Each intersection can be thought of as a three input AND gate, producing the AND of the corresponding bit of the input vectors and coefficient array. The $\underline{Z}_i$ output is then the OR of the outputs of all ANDs in the column (i.e. for the same $\underline{X}_i$).

2.0 THE RD27 MUON TRIGGER

The RD27 muon trigger is described in more detail elsewhere, this discussion only highlights the more important features in order to appreciate what the ASIC has to demonstrate.

Consider the simplified muon chamber system shown in figure 2.1.

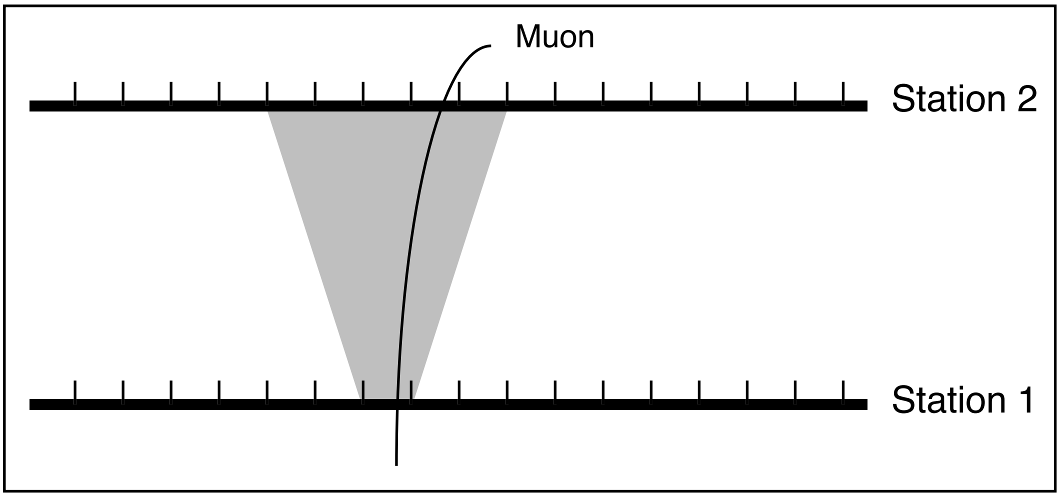

Figure 2.1 – A simplified muon chamber system

The 'stations' or 'planes' are divided into strips oriented into the page, corresponding to the vertical tick marks shown.

A magnetic field is used to exert a bending force on the muons, causing the curved path shown. The amount of bending is inversely proportional to the momentum of the muon; the lower the momentum, the larger the angle through which the muon is bent.

The trigger recognises muons by comparing the output of the two physically separated planes using a coincidence array. Each hit in station one (on the X axis) is compared with any in a 'cone' extended to station two (on the Y axis). The size of the cone is chosen to implement a momentum cut on the muons. It follows that the narrower the cone programmed into the coincidence arrays the higher the momentum of a muon must be for it to pass the trigger. The number of inputs in Y must be greater than that of X because each X input is compared with a number of Y inputs.

Because the size of the cone determines the momentum cut-off applied, we intend to implement, in the one device, the equivalent of two coincidence arrays, processing the same inputs, to provide two decisions that correspond to different momentum thresholds.

The two momentum thresholds are not independent. For example, if the momentum thresholds are at 5GeV and 15GeV then a muon with a momentum of 20GeV will pass both thresholds. Therefore the two thresholds overlap, as shown in figure 2.2.

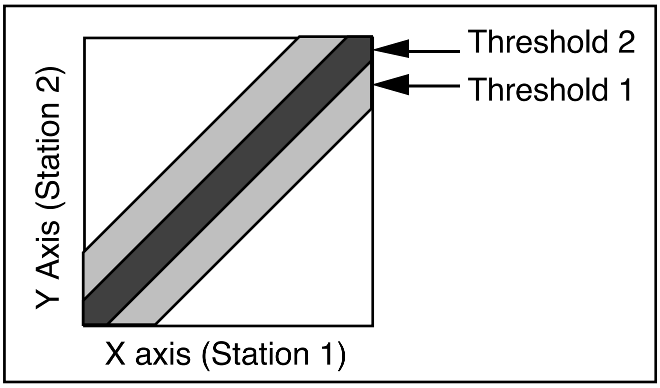

Figure 2.2 – Overlapping momentum thresholds in the coincidence array

Figure 2.3 shows a realistic chamber system, such as that under consideration for the ATLAS detector.

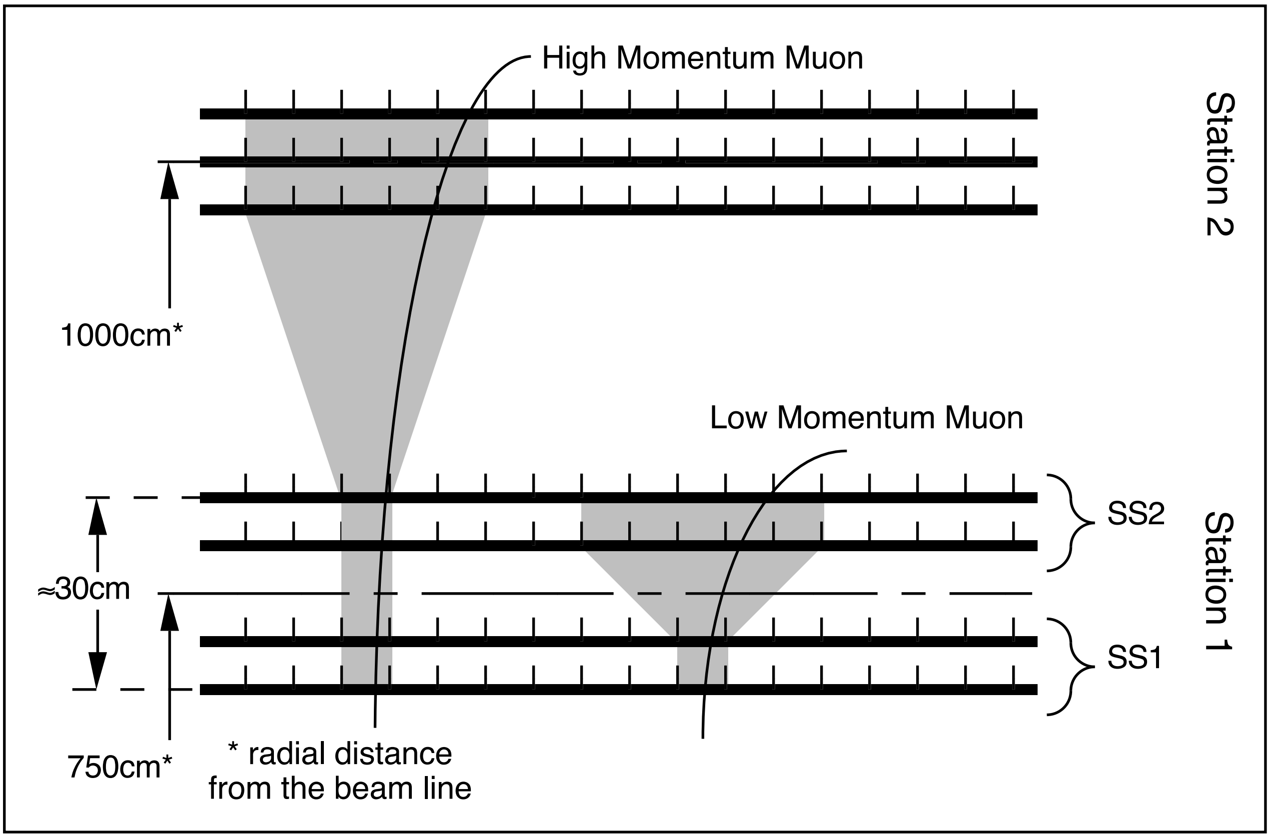

Figure 2.3 – A possible muon chamber layout

The demonstrator is to be designed to prove the viability of implementing both low and high momentum triggers based on this type of layout. Note that this diagram is not drawn to scale and in practice the width of the stations would be small relative to their separation.

2.1 Low Momentum Trigger

Low momentum muons are bent through large angles. In fact the cone that would be required between stations one and two for a 6GeV threshold would be so large as to cause problems for its physical implementation in electronics. This trigger therefore uses the planes of station one only.

A cone is extended from SS1 to SS2 whose width at SS2 depends on the momentum cut-off desired. Because of the smaller separation between the planes the cone will be smaller for the same momentum cut-off than between stations one and two.

The algorithm uses three of four majority logic at each X/Y intersection, as shown in figure 2.4.

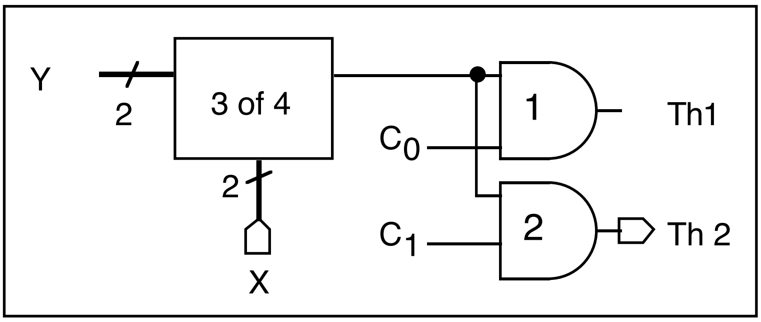

Figure 2.4 – Low momentum threshold logic per intersection

The X input is connected to SS1, the two corresponding strips providing two inputs per column. Similarly the Y input is connected to the two strips in SS2. C_0 and C_1 are co-efficients that specify whether the first or second threshold has been satisfied, respectively.

This algorithm requires that at least three out of the four strips must be hit. This serves two functions:

- It takes into account chamber inefficiency, in order not to suppress muons, and
- It lowers the probability that a combination of random hits, for example induced by low-energy neutrons or photons, will pass the trigger.

2.2 High Momentum Trigger

High momentum muons are bent through smaller angles, therefore it is feasible and indeed necessary to use Station 1 and Station 2 to form the trigger.

In this case the algorithm is similar, except that the X input is the output of the low momentum trigger and the Y input is a two of three majority of the planes in station 2.

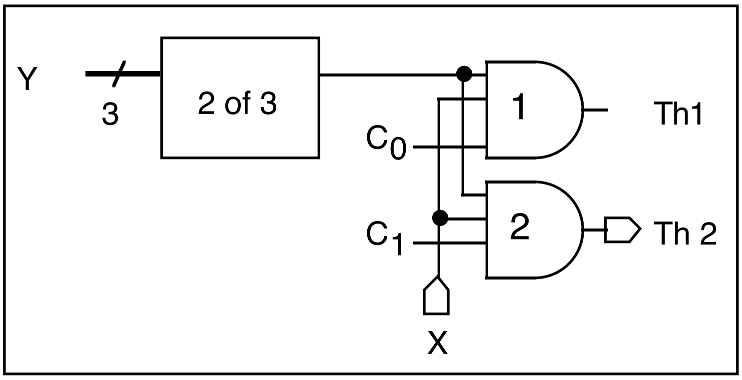

Figure 2.5 – High momentum threshold logic per intersection

2.3 Input Micro-Pipelining

Clearly, it is essential that the inputs to the co-incidence array arrive at or near the same time, despite differing cable lengths and time of flight in the detector.

One possible solution is to work in a pipelined rather than combinatorial mode and have a 'micro-pipeline' on each set of inputs. This is a shift register whose depth can be programmed, such that different inputs can have different depths to align the inputs in time.

To test this method a four stage micro-pipeline is implemented on each input, illustrated in figure 2.6. In this chip the depth can be programmed only for each register (X_A, Y_A etc.). This register can be bypassed to allow strict combinatorial operation.

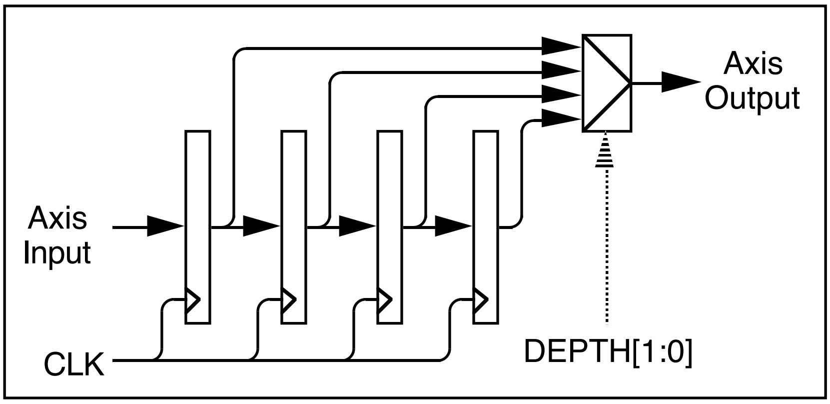

Figure 2.6 – Input Register Micro-Pipeline

3.0 Beam Crossing Identification

Beam Crossing IDentification (BCID) in staggered drift cells is another application where the coincidence array has been used in previous experiments [2].

Consider the drift cells in figure 3.1.

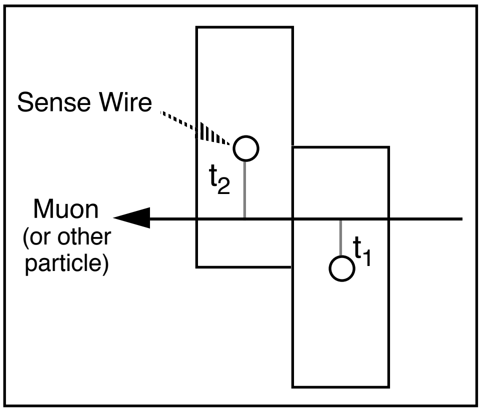

Figure 3.1 - Staggered drift cells for BCID

In the case of detectors like the H1 Forward Muon Chambers the maximum drift time of ionisation in the chamber is greater than the beam crossing time. This means that when a single wire is 'hit', the muon that produced it could have traversed the chamber in any one of a number of previous beam crossings.

It is possible using a coincidence array to process the sense wires of two staggered rectangular drift cells to:

- attribute wire hits to a unique beam crossing, and
- sub-divide the drift cell into smaller drift volumes.

Denote the drift time of ionisation in the gas from the muon track to the two sense wires as t_1 and t_2, respectively. It should be seen that for muons traversing the drift volume at the same angle of incidence the sum of the two drift times is a constant, irrespective of where in the drift volume the particle passed.

If 'valid' muons (from the interaction region, say) always traverse the cells with the same angle of incidence then this technique can be applied.

Consider figure 3.2.

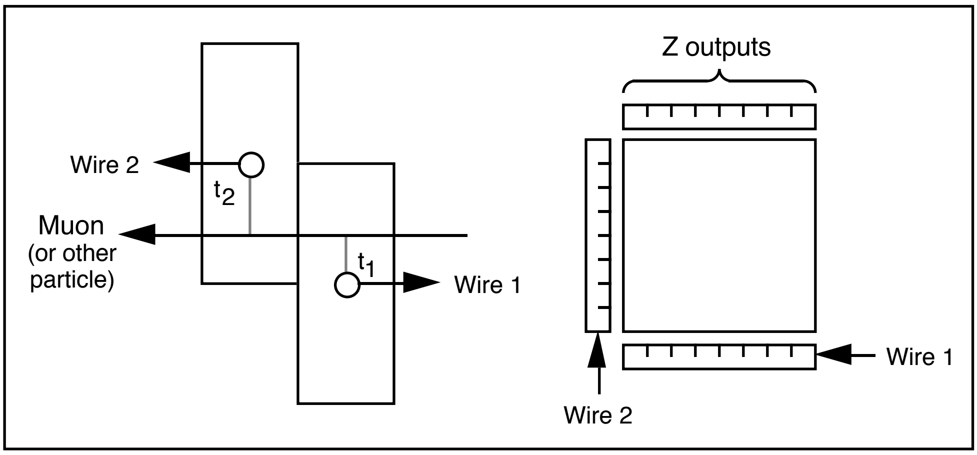

Figure 3.2 - The BCID Algorithm

The coincidence array is driven from shift registers on its X and Y axes, clocked with an ‘axis register clock’ whose frequency is greater than or equal to the beam crossing clock. If, during a clock cycle, a wire is hit by ionisation (i.e. there is a positive edge) a logical ‘1’ is clocked into the shift register, a ‘0’ at all other times (if the signal is longer than one clock cycle it is clipped to one cycle). This has the effect of:

1) Catching signals shorter than the clock period, and
2) Reducing the number of ones in the registers and therefore the number of fake triggers.

The coincidence array is programmed with a pattern (a diagonal as shown) that gives an output for combinations of inputs that correspond to drift times that satisfy:

$$t_1 + t_2 = \mathtt{constant}$$

If the coincidence array is fully programmable then different values of the constant can be programmed by varying the position of this diagonal.

The time at which this lining up occurs is this same constant time after the particle passed through the chamber, digitised in steps of the axis register clock. Because of this digitisation it follows that the faster this clock, the better the time resolution obtained.

It is also true that where in the Z axis the lining up occurs (as given by the position of the output from the coincidence array) is directly related to the position of the track in the chamber. The drift cell can therefore be divided into smaller drift volumes in units of the axis register clock.

4.0 The Coincidence Array Demonstrator ASIC

The ASIC is intended to be flexible enough to demonstrate the high and low momentum muon triggers of RD27, the BCID function and other coincidence array applications, for a reasonable cost.

Therefore this ASIC may not be large enough for the final implementation of a particular system, or may not have certain features that may be desired in a particular implementation. The aim of this ASIC is to demonstrate coincidence array applications, once this technique is proven a new ASIC can be designed that implements the desired functionality, for a given application.

This section is split into two sections. First the ‘run mode’ operation of the device, describing the coincidence array and I/O functions. Secondly the ‘slow control’ functions - loading the array etc.

4.1 Run Mode Operation

The functionality of the device, shown in figure 4.1, has been designed with the above aims in mind.

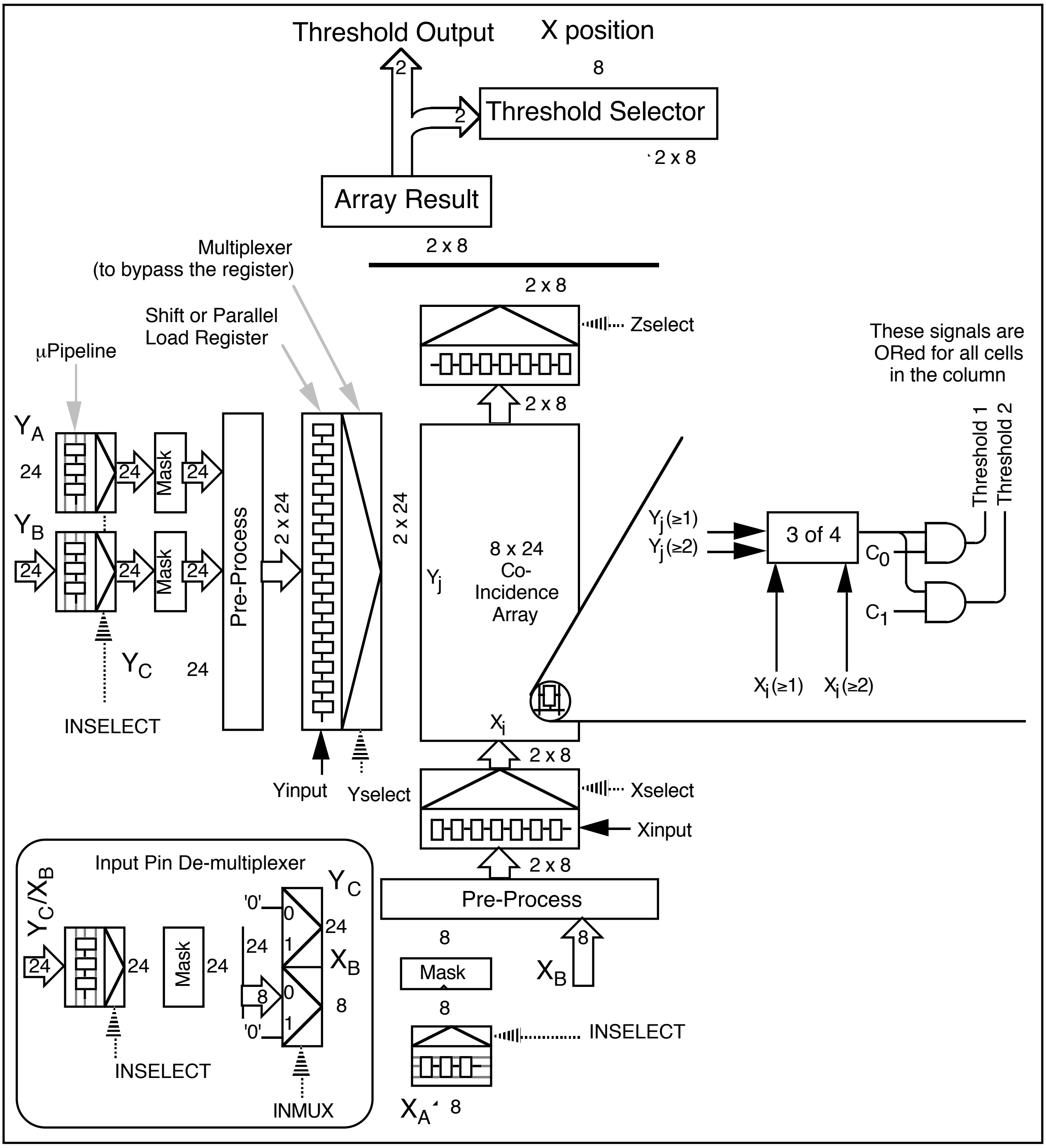

Figure 4.1 - The functionality of the demonstrator ASIC

The Co-incidence Array Logic

As discussed in the section on low and high momentum triggers, the ASIC must make a 3 of 4 majority decision at each intersection of X and Y. In fact it is more efficient to split this function into two steps and perform some processing on the axis inputs (the ‘pre-process’ block) and therefore require less logic at each intersection.

The pre-process step takes the two individual chamber inputs (three in the case of the Y axis) and produces two signals:

≥1 Meaning that more than one high was seen on the inputs, and

≥2 Meaning that more than two highs were seen on the inputs.

The ‘3 of 4’ logic at each location need then only implement the following function:

X_i		Y_j		3 of 4
≥2	≥1	≥2	≥1	decision
0	0	0	0	0
0	1	0	0	0
0	0	0	1	0
1	1	0	0	0
0	0	1	1	0
1	1	0	1	1
0	1	1	1	1
1	1	1	1	1

Note that the combination ≥1 = 0 and ≥2 = 1 is not possible – if there are more than two ones then there must also be more than one one!

This function is simpler than the full ‘3 of 4’ majority function and therefore saves logic overall. This split also has the advantage of allowing the same array logic to implement the low and high momentum trigger schemes. This will be shown later.

The dimensions of the array are eight in X by twenty-four in Y. The two different ‘thresholds’ are implemented using two coefficients per intersection (as shown in the top right of the figure) to encode:

Co-efficients (C_1C_0)	Meaning for the corresponding combination of X_i and Y_j
00	No threshold satisfied
01	Threshold 1 satisfied
10	Threshold 2 satisfied
11	Thresholds 1 & 2 satisfied

Table 4.1 - The array co-efficients

The two AND gates perform the two different threshold functions and then both are individually combined by ORing with all corresponding AND results in the column. The output of the array is therefore two sets of eight column results.

Input Masking

Individual channels can be masked before pre-processing, for example if they are not connected, or if they fail. The logic level that all channels on the chip can be forced to is specified in the `RUNMODE` register.

This function is controlled by four mask registers, one for each of the input groups: X_A, Y_A, Y_B, Y_C/X_B. If bit ‘i’ in the X_A mask register is ‘1’, for example, then that bit is masked before pre-processing, if the bit in the mask register is ‘0’ the bit passes unchanged.

The Array Result

These column results are combined into two single bit outputs, one flags whether threshold 1 is satisfied and the other whether threshold 2 is satisfied. It is possible to have both flags asserted simultaneously.

The X Position

The eight bit column result for the most significant satisfied threshold is then passed out of the chip by the threshold selector. This can be used by external logic to determine the X input(s) that fired the co-incidence array, or used as the input to cascaded co-incidence arrays.

In the case where both thresholds are satisfied simultaneously, threshold 2 is considered more ‘significant’ than threshold 1 and only the position of the threshold 2 co-incidence(s) is produced.

This allows the trigger to be tuned to ‘prefer’ either lower or higher momentum muons, depending on which threshold is programmed into threshold 2.

The Input Registers

The X and Y input registers before the axis pre-processing can be:

- bypassed for combinatorial operation, or
- allow the chip to operate in a programmable depth pipeline mode (μPipelining). The depth of the pipeline is controlled by another internal register, discussed in the slow control section.

In addition, because they can be configured as shift registers and loaded via the slow controls, they allow test vectors to be applied, in-situ, to the pre-processing block and $\mathtt{Y_C/X_B}$ demultiplexer.

The Axis Registers

The X and Y inputs to the array can be driven from the pre-processing either directly or via axis registers, depending on the setting of the respective multiplexers (controlled by the `Xselect` and `Yselect` signals on the diagram).

Similarly the output can be driven from the array or the Z axis register, under the control of the `Zselect` signal.

Note that it is possible to configure the device such that it is completely combinatorial in operation, completely registered, or a mixture of the two.

The output of the pre-processing is two sets of signals, one ≥1 and one ≥2. To accommodate these, and allow the BCID function to be performed, there are in fact two shift registers per axis as shown in figure 4.2 below, for the Y axis. It is the same for the X axis, with the exception that there are only two sets of eight bit inputs to the pre-process stage.

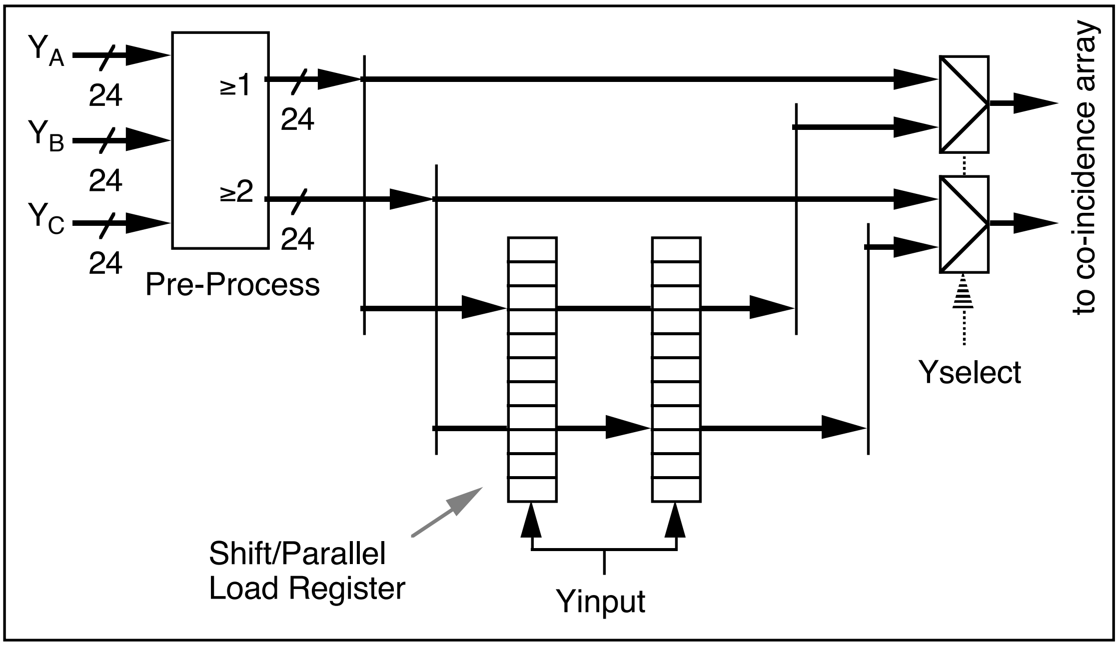

Figure 4.2 – The axis registers

If the axis register is selected to drive the array, then the BCID function can be performed by configuring the axis registers as shift registers and connecting the `Xinput` and `Yinput` signals to the chamber outputs. The `Xinput` and `Yinput` signals are connected to both the ≥1 and ≥2 shift registers so the 3/4 majority logic function of the array need not be changed to accommodate running using the shift register.

These run-time inputs to the shift registers can be configured as either:

- standard synchronous inputs, where the external input applied directly to the shift register, or
- asynchronous, where the input is first passed through a 'one's catching and suppression' circuit, as described in the BCID section.
- synchronous, from the output of the same register. This allows periodic patterns to be applied to the array in order to produce a periodic output that can be used for external debugging.

To implement the RD27 low and high momentum trigger schemes, the axes are configured as combinatorial input, through the parallel inputs. The parallel inputs are then driven as shown below:

Axis Input	Low momentum trigger	High momentum trigger
X_A(7:0)	SS1 plane 1	Low momentum output
X_B(7:0)	SS1 plane 2	'0'
Y_A(23:0)	SS1 plane 3	S2 plane 1
Y_B(23:0)	SS1 plane 4	S2 plane 2
Y_C(23:0)	'0'	S2 plane 3

Therefore, with the functionality shown in figure 4.1, all of the functions described in sections two and three can be performed.

Not all of these parallel inputs can be brought out individually to pins on the package because of a limit on the number of I/O pins. However all five sets of inputs are not used simultaneously therefore X_B can be multiplexed with Y_C, saving eight pins. This is controlled by a bit in an internal register, as described in the next section. The unused inputs internally are driven with a logical '0', in order not to interfere with the majority logic.

4.2 Slow Control

The slow control interface is used primarily to set and check the coincidence co-efficients of the array. Several other useful functions can also be performed, at little extra cost, if the X, Y and Z axis registers can be written to and read from. For example:

- the pre-processors and output logic can be tested directly for correct functionality (both in the test lab and in-system),
- the input to and output from the coincidence array while running in parallel input mode (i.e. through the pre-processors) can be captured in the axis registers and later read-out to monitor the operation of the system, and
- the axis registers can provide a source of test vectors to the coincidence array in the test lab and in-system.

The device has two modes of operation: I/O mode and RUN mode. I/O mode is used to access the axis registers and co-efficients; RUN mode is the normal mode of operation to perform the coincidence function.

The device is switched between I/O and RUN modes using an external `RUN*` signal. When `RUN*` is asserted (low) the device is in RUN mode and the axis registers are controlled by the '`RUNMODE`' register. When `RUN*` is deasserted the axis registers are idle and the device can be accessed via the slow control interface.

The RUNMODE Register

In run mode the axis and input registers can be configured as either parallel load or shift registers. Operations on the registers are synchronous with a global clock, under the control of the following signals:

- a parallel load enable, and
- a shift enable that controls the shifting operation

There are four sets of such signals, one per axis and one for the input registers, allowing each type of register to be configured independently.

Figure 4.3 shows the bit assignment of the RUNMODE register.

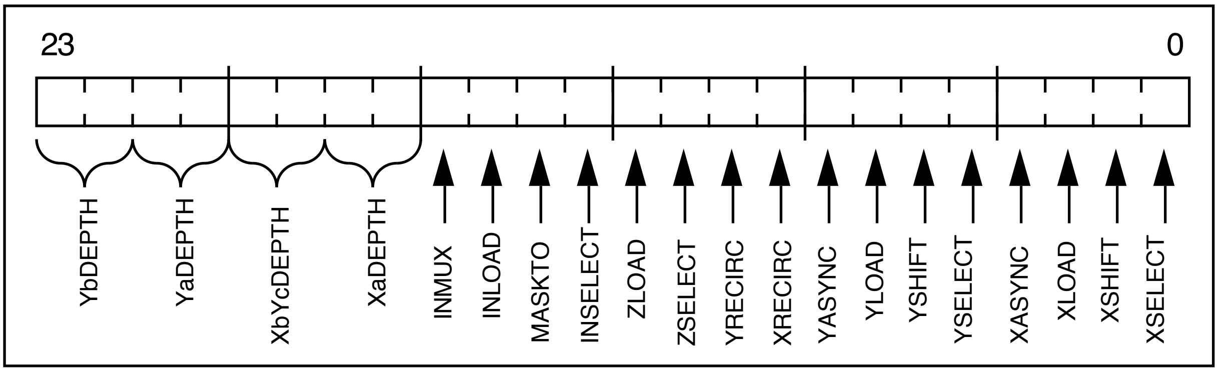

Figure 4.3 - The RUNMODE register

`INSELECT` Controls the input multiplexer:

- 0 selects the micro-pipeline
- 1 bypasses the pipeline for combinatorial operation

`DEPTH` Specifies the depth of the micro-pipeline.

- 0 one pipeline stage
- 1 two pipeline stages
- 2 three pipeline stages
- 3 four pipeline stages

`INMUX` Controls the de-multiplexing of the X_B and Y_C registers:

- 0 the lower eight bits of X_B/Y_C are used as the X_B input
- 1 X_B/Y_C is used as the Y_C input

`ASYNC` Controls the serial input:

- 0 selects straight synchronous input
- 1 selects asynchronous processed input (one's catching and suppression)

The following bits have the following functions for each axis:

`SELECT` Controls the axis multiplexer:

- 0 selects axis register
- 1 bypasses the register for combinatorial operation

`SHIFT` Controls shifting:

- 0 axis register static
- 1 axis register shifts during running

`LOAD` Controls parallel loading:

0 axis register static

1 axis register is parallel loaded with the combinatorial input every clock cycle

NOTE: the `LOAD` signal overrides the `SHIFT` signal if both are asserted simultaneously

`RECIRC` Controls recirculating the register output:

0 selects input pin as input to register (synchronous or asynchronous)

1 selects register output as input to register

`MASKTO` Specifies which state to mask the specified channels to:

0 mask to ‘0’

1 mask to ‘1’

Accessing the RUNMODE and Axis Registers

The following ‘physical registers’ are present in the coincidence array demonstrator:

- the four input μPipelines (X_A, Y_A, Y_B, Y_C/X_B),
- the four input mask registers (X_A, Y_A, Y_B, Y_C/X_B),
- the `Xgt1/Xgt2` axis register,
- the `Ygt1` and `Ygt2` axis registers,
- the `Zth1/Zth2` axis registers, and
- the `RUNMODE` register.

Only one of the four registers in each input micro-pipelines (X_A, Y_A, Y_B, Y_C/X_B) can be accessed at once because the number of slow control registers is restricted to sixteen. However, for testability, they must all be accessible. This is resolved by selecting one of the four pipeline registers to be accessed using the `DEPTH` bits in the `RUNMODE` register:

DEPTH	Pipeline stage selected for slow control access
0	first
1	second
2	third
3	fourth

All are accessed as shift registers connected between an input pin and an output pin, as illustrated in figure 4.4.

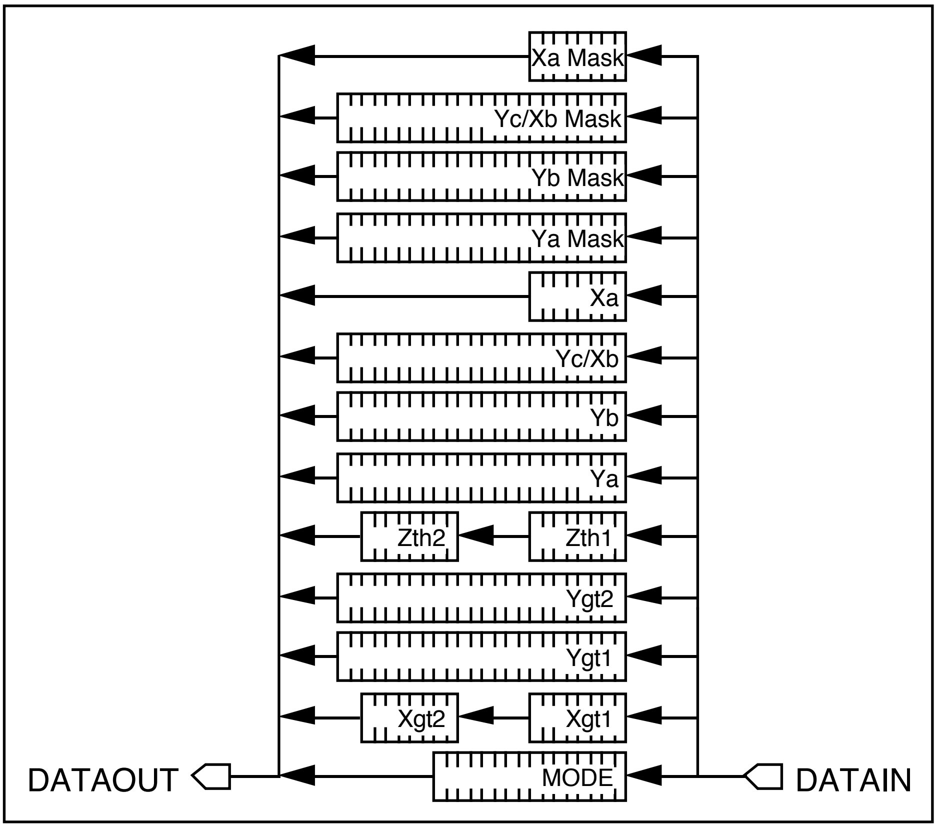

Figure 4.4 - The registers

These physical registers and those used to manipulate the co-efficients of the array are mapped into a set of sixteen registers accessible from the slow control interface. The interface to these registers consists of:

- An active low chip select `CS*`.
- Four address bits to select which slow control register is to be accessed; `A2`, `A1` and `A0`, and
- An active low strobe that controls access to the registers; `STROBE*`.

The following timing specifications must be met:

- The `CS*` and `STROBE*` signals must be synchronous with the falling edge of the clock,
- The address and `CS*` signals must be asserted at least one clock cycle before `STROBE*`,
- The address and `CS*` signals must be de-asserted at least one clock cycle after `STROBE*`.

The slow control register assignment is shown in table 4.3:

Address (hex)	Register	Description	No. of bits
0	`RUNMODE`	the run mode register	16
1	`Zth1/Zth2`	the Z axis registers	16
2	`Xgt1/Xgt2`	the `Xgt1` and `Xgt2` axis registers	8 + 8
3	`Ygt1`	the `Ygt1` axis register	24
4	`Ygt2`	the `Ygt2` axis register	24
5	`YA`	the Y_A input register	24
6	`YB`	the Y_B input register	24
7	`YC/XB`	the Y_C/X_B input register	24
8	`XA`	the X_A input register	8
9	`YA`	the Y_A mask register	24
A	`YB`	the Y_B mask register	24
B	`YC/XB`	the Y_C/X_B mask register	24
C	`XA`	the X_A mask register	8
D	`RESET`	reset all registers except `RUNMODE`	-
E	`WR`	Write co-efficients	–
F	`RD`	Read co-efficients	–

Table 4.3 - Slow control register assignment

The steps in a slow control operation are as follows:

(1) Assert the chip select of the chip to be accessed.

(2) The address of the register to be accessed is presented on the address lines.

(3) The strobe is used to pace the operation with the global clock.

Figure 4.5 shows a timing diagram for accessing the `RUNMODE` register. Accesses to the other, physical, registers are performed in the same way with the exception that the number of bits transferred is different. Note that when accessing the registers the content of the `RUNMODE` register is overridden. Therefore the axis registers do not have to be configured as shift registers before a slow control operation can be performed. It must be stressed that the contents of the `RUNMODE` register are not **overwritten**, just **overridden**.

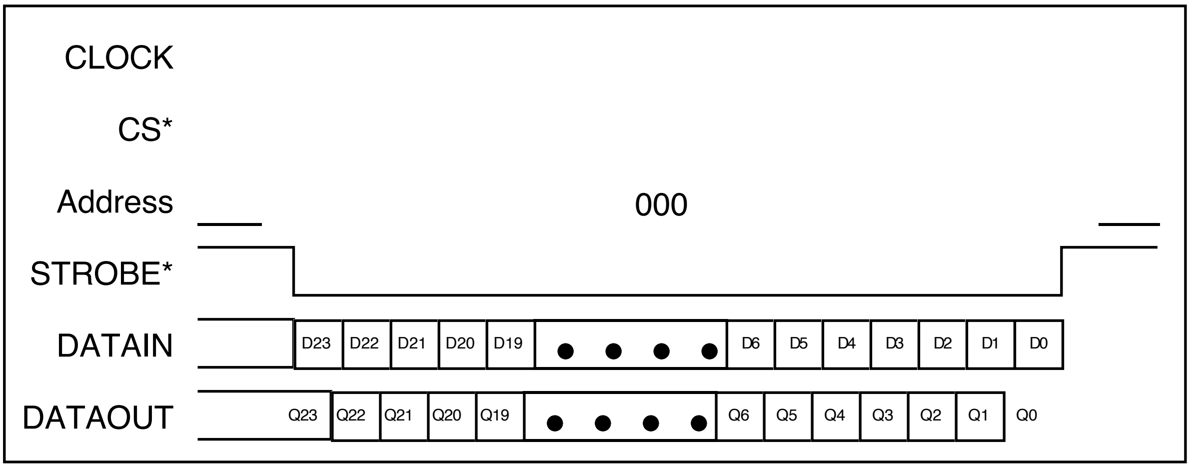

Figure 4.4 - Accessing the RUNMODE register

Before this operation the `RUNMODE` register contained [`Q23`,`Q22`,…,`Q0`]. After operation the `RUNMODE` register contains [`D23`,`D22`,…,`D0`]. As shown in the figure, data is enters `DATAIN` most significant bit (MSB) first, and leaves `DATAOUT` MSB first.

During reading, the output data can be recirculated externally back into the input to leave the register with the same contents as before the read operation.

The three non-physical registers (`RESET`, `WR` and `RD`) are special cases of the above. The operation is performed in the same way but the strobe need only be active for one cycle. The operation of these registers is discussed below.

The Reset Register

Accessing this register resets the input and axis registers to ‘0’. This could be useful in the case of BCID, to ensure that old data does not get compared with new, which could happen if new data were simply clocked in ‘over the top’ of old data.

Writing co-efficients into the array

Co-efficients C_0 and C_1 are accessed, on a row-by-row basis, via the `WR` register. The sequence of events for changing the coefficients of a row in the array is:

(1) Place the pattern to write into the array into the `X` axis registers. The C_0 co-efficient of column `i` will be changed to the value of `Xgt1[i]` and C_1 of column `i` to the value of `Xgt2[i]`.

(2) Specify which row(s) of the array to write the pattern into using the `Ygt2` axis register. The `Ygt2` register is used as a ‘write mask’; i.e. the co-efficients of row `j` are changed if `Ygt2[j]` is ‘1’, otherwise the co-efficients in row `j` are untouched.

(3) Access the `WR` register.

Reading co-efficients from the array

When the `RD` register is accessed, the output of the co-incidence array is written into the `Z` register. The C_0 co-efficient of column `i` will be written into `Zth1[i]` and C_1 of to `Zth2[i]`.

This can be used to test the array and to read the value of the co-efficients. For example, if bit `j` of the `Ygt1` and `Ygt2` registers is set to ‘1’ and all `Xgt1` and `Xgt2` bits are set to ‘1’, the bit pattern loaded into the Z register will correspond to the value of the co-efficients in row `j`.

4.3 The Device Pinout

The I/O pins of the coincidence array demonstrator are detailed below:

Pin name	No. of pins	Description
X_A[7:0]	8	The 'A' group of X inputs.
Y_A[23:0]	24	The 'A' group of Y inputs.
Y_B[23:0]	24	The 'B' group of Y inputs.
Y_C/X_B[23:0]	24	Either the 'C' group of Y inputs or the 'B' group of X inputs.
THR1	1	Threshold 1 satisfied.
THR2	1	Threshold 2 satisfied.
XPOS[7:0]	8	The X axis position of the most significant coincidence.
Xinput	1	The serial input to the X axis.
Yinput	1	The serial input to the Y axis.
CLK	1	The global register clock.
RUN*	1	The run mode strobe.
CS*	1	Chip select.
DATAIN	1	The data input to the slow control registers.
DATAOUT	1	The data output from the slow control registers.
A[3:0]	4	The slow control register address.
STROBE*	1	The slow control strobe.
TOTAL pins:	102	

Signal I/O:

All signals are single ended TTL.

Power:

The device runs from a 3.3V power supply. The I/O can be powered by 5V to give standard 5V I/O.

5.0 REFERENCES

[1] "A Fast Processor for Muon Triggering Using Resistive Plate Chambers"; E. Gennari, E. Petrolo and S. Veneziano; Conference Record of the 1991 IEEE Nuclear Science Symposium, Santa Fe, New Mexico, November 2-9, 1991; pp 579-583

[2] "Fast Pipelined Trigger Processors for the Forward Muon Chambers of the H1 Experiment on the Hadron Electron Ring Accelerator (HERA) at DESY, Hamburg"; J. Dowdell, M. French, S. Jaroslawski and M. Prydderch; Conference Record of the 1991 IEEE Nuclear Science Symposium, Santa Fe, New Mexico, November 2-9, 1991; pp 898-902