SuperNEMO Collaboration
SuperNEMO Demonstrator Trigger strategy details
Version 0.1 (draft)

Abstract

This document aims to describe the SuperNEMO trigger strategy for the full detector (calorimeter and tracker). It also describes the global process in an electronic way with a practical physical point of view.

Source (LaTeX) :
`https://nemo.lpc-caen.in2p3.fr/svn/??`
Reference : NemoDocDB-doc-4039-v1

Versions :

- NemoDocDB-doc-4039-v1 : 0.1 (draft)
  - Calorimeter trigger strategy
  - Tracker trigger strategy
  - Coincidences trigger strategy

Contents

note: Après réunion au LAL : ajouter choses sur les Trigger ID en calorimètre + tracker Trigger ID équivalent a event ID. 5 bits poids faibles (LSB) sur les 32.
exemple : calo -> ID 3
tracker associé -> ID 3
tracker retardé -> ID 4
calo nouveau -> ID 5

note: A voir: calo decision : mode polling pour jihane ? ou thierry qui envoie un signal

note: L1 calo decision : quelle voies lit Jihane, HT seulement ou LT + HT (au niveau du readout) <– a voir absolument

note: A définir : que sort-on de la trigger board ?

1 Trigger strategy introduction

The goal of the trigger system is to select only events of physical interest (electron, gamma, alpha . . .), reject spurious events (self triggering of the tracker drift cells or PMTs) and reduce the general acquisition rate. At each clock tick (25ns), we collect trigger primitive (TP) signals associated to each tracker or calorimeter channel. The trigger system is designed to merge all these TP signals and make a decision from this information within a central programmable trigger board. Once the decision is made to record an event of interest, dedicated signals are sent back to the CBs and FEBs to initiate the data acquisition sequence for the selected channels. Due to the large number of channels (both for the calorimeter and the tracker) and the geometry of detector, it has been decided, as for NEMO3, to consider only the information from the X-Y view of the detector. That means that no Z information (vertical axis) is used to build the TPs.

1.1 Detector zoning

In order to be able to apply some spatial coincidence criteria between hits from different parts of the detector (PMT, geiger hits) at the central trigger level, 10 tracker triggering zones (TTZ) have been defined on each side of the tracking chamber. Each TTZ is defined in the $x - y$ view only (flatten detector). A TTZ possibly has several neighbors : main calo or X-wall columns, some other TTZs. For more information, see the SNDER [1], page 35.

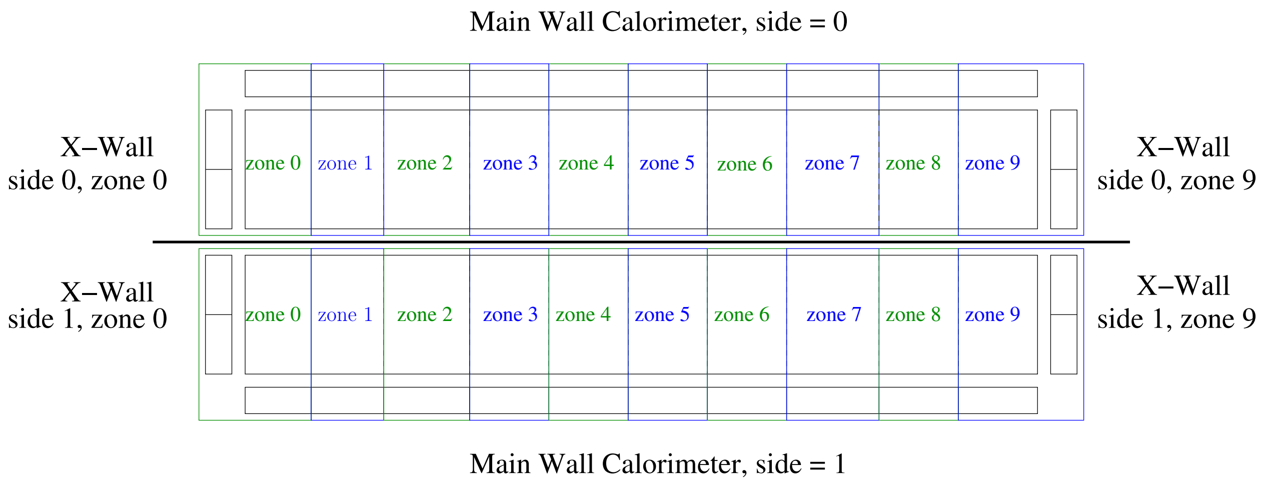


Figure 1: Zoning of the detector with X-Walls.

zone ID	Geiger row IDs	[#]	Main calo. column IDs	X-wall wall ID.column IDs
0	0-8	[9]	0-1	0.0, 0.1
1	9-20	[12]	2-3	–
2	21-32	[12]	4-5	–
3	33-44	[12]	6-7	–
4	45-55	[11]	8-9	–
5	56-67	[12]	10-11	–
6	68-79	[12]	12-13	–
7	80-91	[12]	14-15	–
8	92-103	[12]	16-17	–
9	104-112	[9]	18-19	1.0, 1.1

Table 1: Association of Geiger cell rows, main wall and X-wall blocks and zones for side ID=0 (the same scheme is used for both sides).

1.2 Reminder of information in Front-End Boards (FEB) and Control Boards (CB)

Each 25ns, information from calorimeter Front-End Board (calo FEB) is sent to calorimeter Control Boards (calo CB) and then in the Trigger Board (TB). The goal of the calorimeter trigger strategy is to compute and merge all information coming from the 3 calo CB to construct a Level 1 (L1) calorimeter decision.

Calorimeter Trigger Primitive (TP) @ 25ns

Each column of optical module is connected to a calo FEB. Each calo-FEB build a Trigger Primitive Bitset which will be sent to the dedicated Control Board (CB) each 25ns ($f = 40$ MHz).
A calo TP is composed by 5 bits in each calo FEB (see the SNDER [1] for more details).

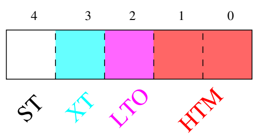


Figure 2: Calo Trigger Primitive Bitset

Calorimeter Crate Trigger Word (C-CTW) @ 25ns

Each 25ns, the three calorimeters CB constructs a word composed by 18 bits called the calorimeter Crate Trigger Word (calo CTW). There is 3 CTWs : 1 for each crate. CTW for crate ID = 0 - 1 are displayed on figures 3 and 4. CTW for crate ID = 2 is displayed on figure 5. The definition for crate = 2 is different because of calo x-walls and gamma veto bits. The location and multiplicity are kept for x-walls but the multiplicity only is kept for gamma veto.

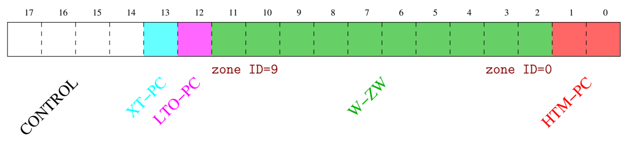


Figure 3: Calorimeter Crate Trigger Word (C-CTW) from Control Board in `crate ID=0`

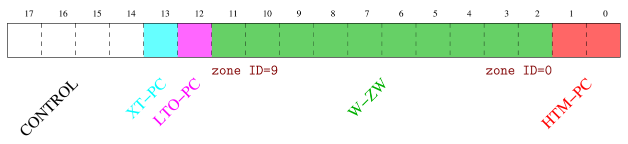


Figure 4: Calorimeter Crate Trigger Word (C-CTW) from Control Board in `crate ID=1`

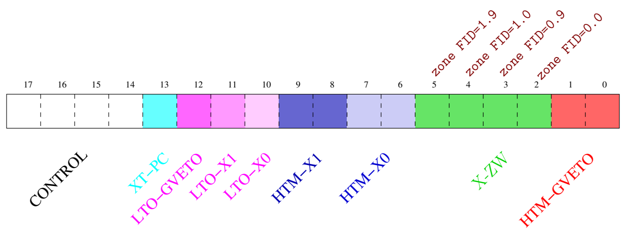


Figure 5: Calorimeter Crate Trigger Word (C-CTW) from Control Board in `crate ID=2`

Tracker Trigger Primitive (TP) @ 800ns

note: information to be check

Each 800ns, the tracker FEBs produce a Tracker Trigger Primitive (tracker TP) composed by 100 bits. In this bitset, the status of 36 Geiger cells is given and others information like the board address composed the TP. Each FEB produces a tracker TP and send it to the dedicated tracker Control Board (CB).

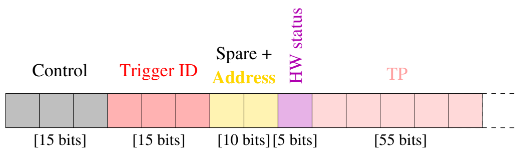


Figure 6: Tracker Trigger Primitive bitset

Tracker Crate Trigger Word (T-CTW) @ 800ns

note: The definition of Tracker Trigger Primitive has been changed. There are some changes in the CTW. To be checked with people at LAL

Reminder : a tracker crate trigger word is sent to the trigger board each 1600 nanoseconds. This limitation is due to the limitation of the data flow between control boards and the trigger board. So, the reference for a clock tick is now 1600ns in the trigger board. One T-CTW on two is not sent to the Trigger Board. The main clock tick in the Trigger Board is now 64 $\times$ 25 nanoseconds.

2 Calorimeter trigger strategy in Trigger Board (TB)

The three calo CTW ([18 bits] $\times$ 3) comes in the TB each 25ns. This information from CTWs has to be exploit to construct a calo L1 decision.

It has been decided to merge information (i.e. zoning and multiplicity) from x-walls into main wall information. To do that, the CTW for crate number 2 has been merged into the others. Multiplicity is sided so multiplicity coming from x-walls can be added to the corresponding side. Then, the zoning word from x-walls is merged with zones 0 or 9 depending of each side.

Each bit of the zoning word is activated when (at least) one calorimeter has passed the high threshold in the corresponding zone. Information from x-walls is included in zone 0 or 9 depending of which calo x-wall is hit.

From 3 $\times$ 18 bits, an intermediate bitset is build and call a **calo record**. This bitset of 32 bits is created **each 25ns** and composed by :

- Side 0 :

  Zoning Word [10 bits]

  HTM-S0 : High Threshold Multiplicity [2 bits]

  LTO-S0 : Low Trigger Only side 0 [1 bit]

- Side 1 :

  Zoning Word [10 bits]

  HTM-S1 : High Threshold Multiplicity [2 bits]

  LTO-S1 : Low Trigger Only side 1 [1 bit]

- Others :

  HTM-GVETO : High Threshold Multiplicity in γ-veto calorimeter [2 bits]

  LTO-GVETO : Low Trigger Only in γ-veto [1 bit]

  XT-info : External information [3 bits]

For a unique nuclear decay, calorimeter datas can be spread on 4 clock ticks @ 25ns due to the particle velocity and detector size. That's why, it has been decided to cumulate 4 calo records, created each 25ns, to build a new bitset called the **calo summary record**. The calo summary record bitset is composed of 35 bits and it is build **each 25ns** based on the actual calo record and the last 3.
To do this computing, a circular buffer with a depth of 4 is used. To build the calo

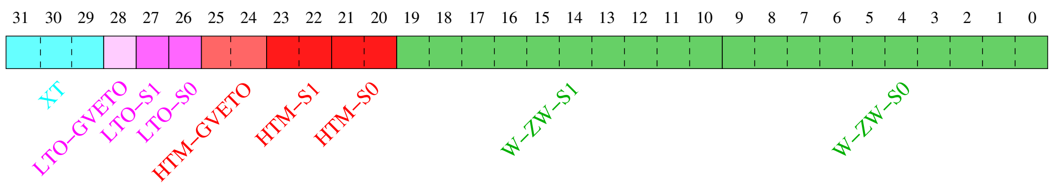


Figure 7: A Calo record bitset.

summary record bitset, the information of the 4 calo records has to be cumulated. For the zoning word, an 'OR' operation is done. For the multiplicities (HTM side 0/1/gveto), an addition is done and for the low trigger bits an 'OR' is done.

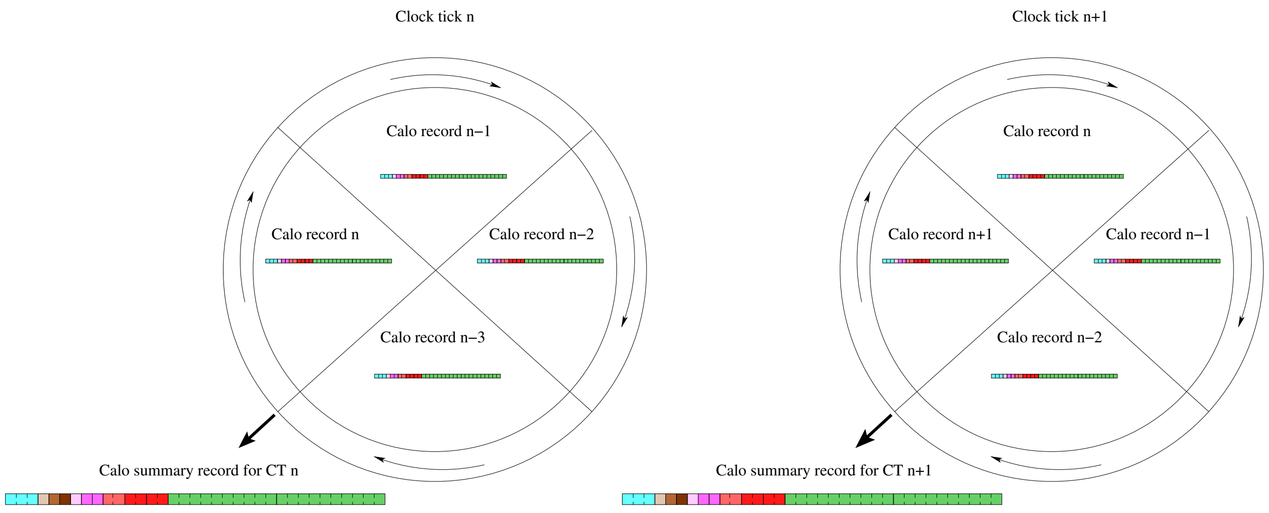


Figure 8: The calorimeter circular buffer to construct the calo summary record. It is based on the actual calo record and the last 3 calo record

So this bitset is quite the same as a calo record bitset but 3 bits are added which are :

- SS coinc : Single Side coincidence bit [1 bit]

  Value `0` Zones of both sides are activated

  Value `1` Only zones of the same side are activated

- TTM : Total Threshold Multiplicity bit [1 bit]

  Value `0` Threshold multiplicity set in the trigger configuration is not exceeded

Value 1 Threshold multiplicity set in the trigger configuration is exceeded

- Calo finale decision [1 bit] : based on configurable criteria

  Value 0 For this clock tick the calo L1 decision is not passed

  Value 1 For this clock tick the calo L1 decision is passed

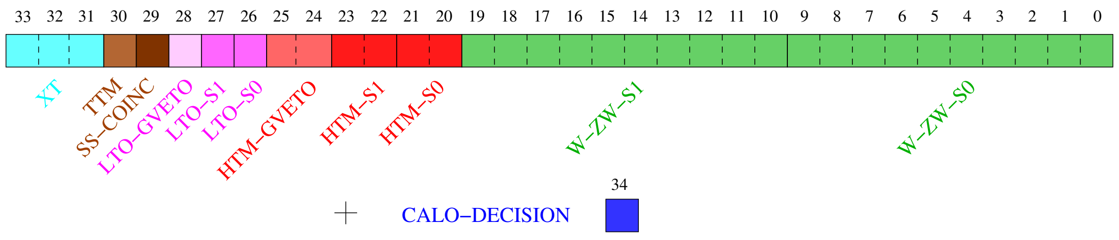


Figure 9: A calo summary record bitset.

Calorimeter status

So, each 25ns the information of the 3 calo CTW (54 bits in total) is reduced into a 32 bits word called a calo record. Then, **for each clock tick of 25ns**, a calo summary record is created based on the actual calo record and the 3 last. It cumulates the information of the last 3 clock ticks and the actual clock tick (quite the same as a gate in 100ns but for each clock tick). One calo summary record is composed by 35 bits and it is produce each 25ns.

The description of the coincidence between calorimeter and tracker will be done in section 4

3 Tracker trigger strategy in Trigger Board (TB)

The main goal of the tracker trigger is to compute if there is a track in the tracker. Each 1600ns, three tracker CTWs are coming from the three tracker CBs. Each CTW is composed by 1900 bits (5700 in total) and 2034 are representing the status of each Geiger cells in the tracker.

Construction of Geiger matrix

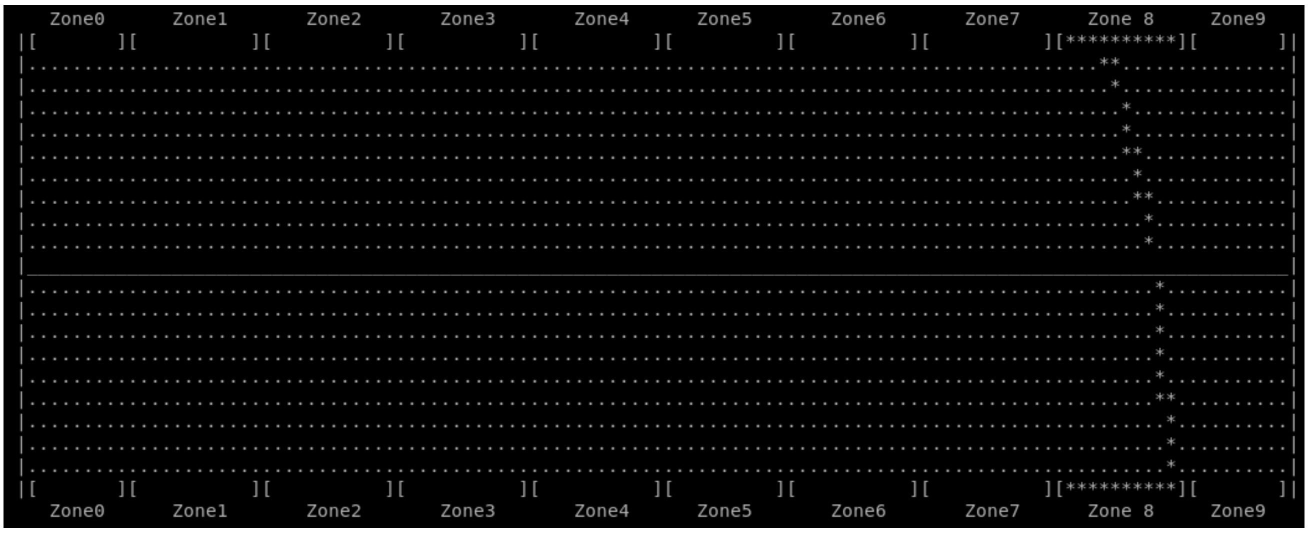
The first thing to do is to build the Geiger status matrix which is a top view of the tracker chamber. This status is based on the anode signal status (Fired/unfired = 1/0).

- One bit can be associated for one Geiger Cell

  Value `0` The Geiger cell is not hit

  Value `1` The Geiger cell is hit

The figure 10 represents the Geiger matrix for a simulated event of double beta decay. Geiger cells fired are representing with stars.



```
   Zone0      Zone1       Zone2       Zone3       Zone4      Zone5       Zone6       Zone7      Zone 8     Zone9
|[       ][          ][          ][          ][          ][         ][          ][          ][**********][       ]|
|................................................................................................**...............|
|.................................................................................................*...............|
|..................................................................................................*..............|
|..................................................................................................*..............|
|..................................................................................................**.............|
|...................................................................................................*.............|
|...................................................................................................**............|
|....................................................................................................*............|
|....................................................................................................*............|
|_________________________________________________________________________________________________________________|
|.....................................................................................................*...........|
|.....................................................................................................*...........|
|.....................................................................................................*...........|
|.....................................................................................................*...........|
|.....................................................................................................*...........|
|.....................................................................................................**..........|
|......................................................................................................*..........|
|......................................................................................................*..........|
|......................................................................................................*..........|
|[       ][          ][          ][          ][          ][         ][          ][          ][**********][       ]|
   Zone0      Zone1       Zone2       Zone3       Zone4      Zone5       Zone6       Zone7      Zone 8     Zone9
```

Figure 10: A typical view of the geiger matrix in the Trigger board after matrix reconstruction. This is an event of neutrinoless double beta decay for ^{82}Se. The source foil is the line in the middle. Hit Geiger cells are represented with stars.

Sliding zone concept

The tracker is divided in 20 different zones. Each side is composed by 10 zones. The goal of the tracker trigger strategy is to identify if there is a track in each tracker zone. The main goal is to identify Geiger cells clusters which sign a charged particle track. An uniform treatment have to be done in the full tracker and edge effect must not affect the efficiency for pattern recognition. To avoid this edge effect, the concept of sliding zones 'floating' over the tracker has been implemented. To construct the response for one zone, 4 sliding zones are used (A,B,C and D) and presented in figure 11. The size of a sliding zone is always 9 layers (vertical) and 8 rows (horizontal).

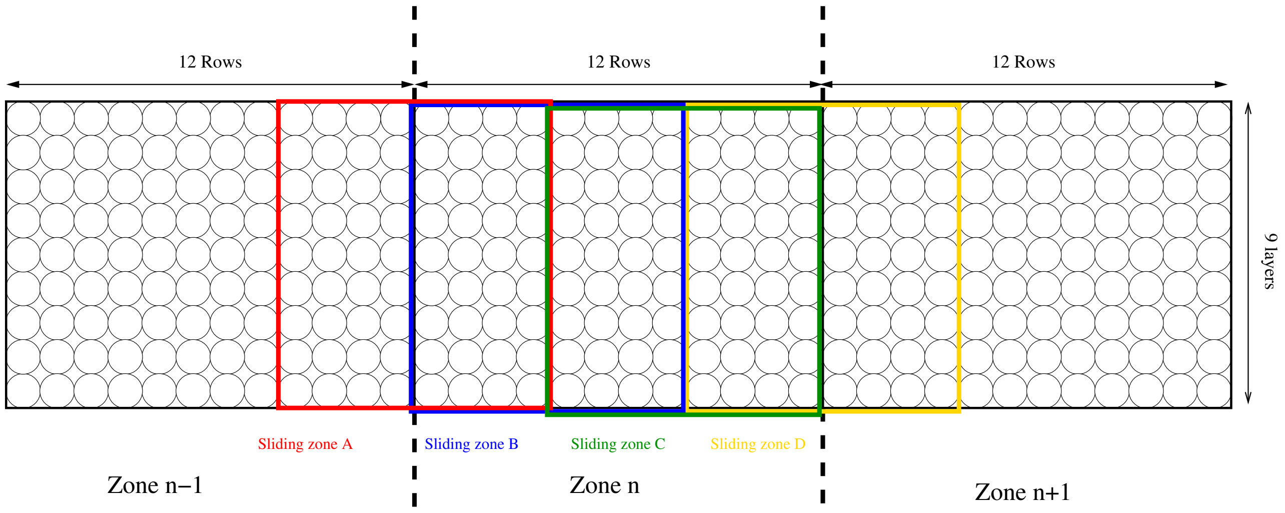


Figure 11: Representation of 3 zones n-1/n/n+1 and the 4 sliding zones corresponding to to the zone n. The size of a sliding zone is 9×8 Geiger cells

Tracker algorithms with example

To introduce this concept, two examples will be taken. One for an electron pattern (blue track), figure 12 and the other for an alpha pattern (purple track), figure 13.

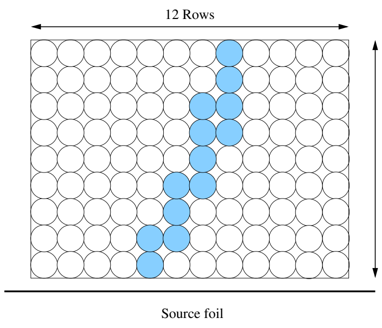


Figure 12: An electron track pattern in the tracker

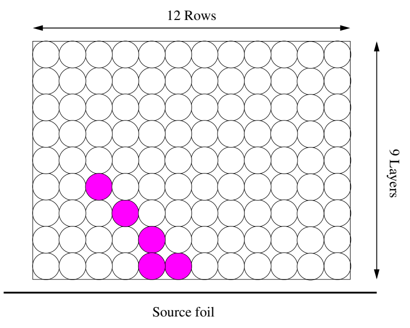


Figure 13: A delayed alpha track pattern in the tracker

warning: Take care of zones at the beginning / center and end of the tracker ! For these special cases see the appendix 24

The size of a 'regular' tracker zone is 9 layers $\times$ 12 rows so 2^{108} combinations are possible (2 states for a Geiger cell). It is impossible to test each combination so the goal of these algorithms is to reduce the combinatorial and search for track patterns thanks to clustering algorithms.

We want to keep the maximum information with a minimum amount of bits that's why projections are used to reduce the combinatorial. The information of a sliding zone is reduced in 2 bitsets which are the **layer projection bitset** (9 bits) and the **row projection bitset** (8 bits). Now, a sliding zone is defined by 17 bits. The next step is to reduce these 2 bitset of 9 and 8 bits and construct a reduced bitset (vertical and horizontal) for each sliding zone.

To construct this reduced bitset word, programmable memories are used to compute the projection bitset. The vertical bitset of 9 bits is reduced into 2 bits called the **inner bit** and the **outer bit**. The horizontal bitset is also reduced into 2 bits called the **left bit** and the **right bit**.

The example of the electron track in the tracker is presented in figure14. The response of this track will be construct and each sliding zone can be now described by these 4 bits as it is showed in figure 15. Two memories are used to do this computing. Memory 1 is for the vertical projection and it is an A9D2 memory. The building of memory 1 is described in appendix B. Memory 2 is for the horizontal projections for each sliding zone and it is an A8D2 memory. The building of memory 2 is described in appendix C.

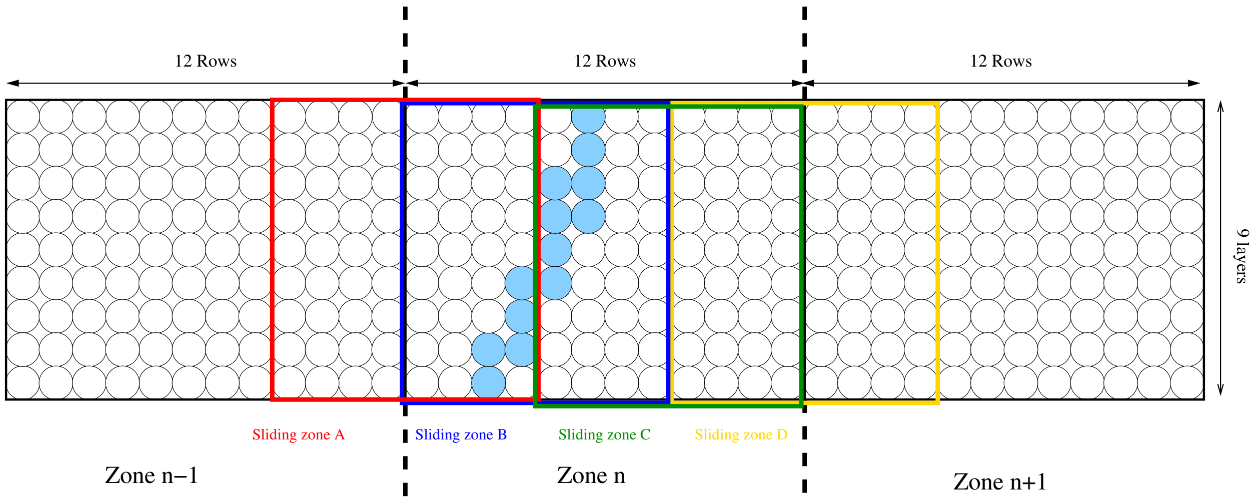


Figure 14: Electron pattern in the tracker. Construction of the response for each sliding zone and then the zone response.

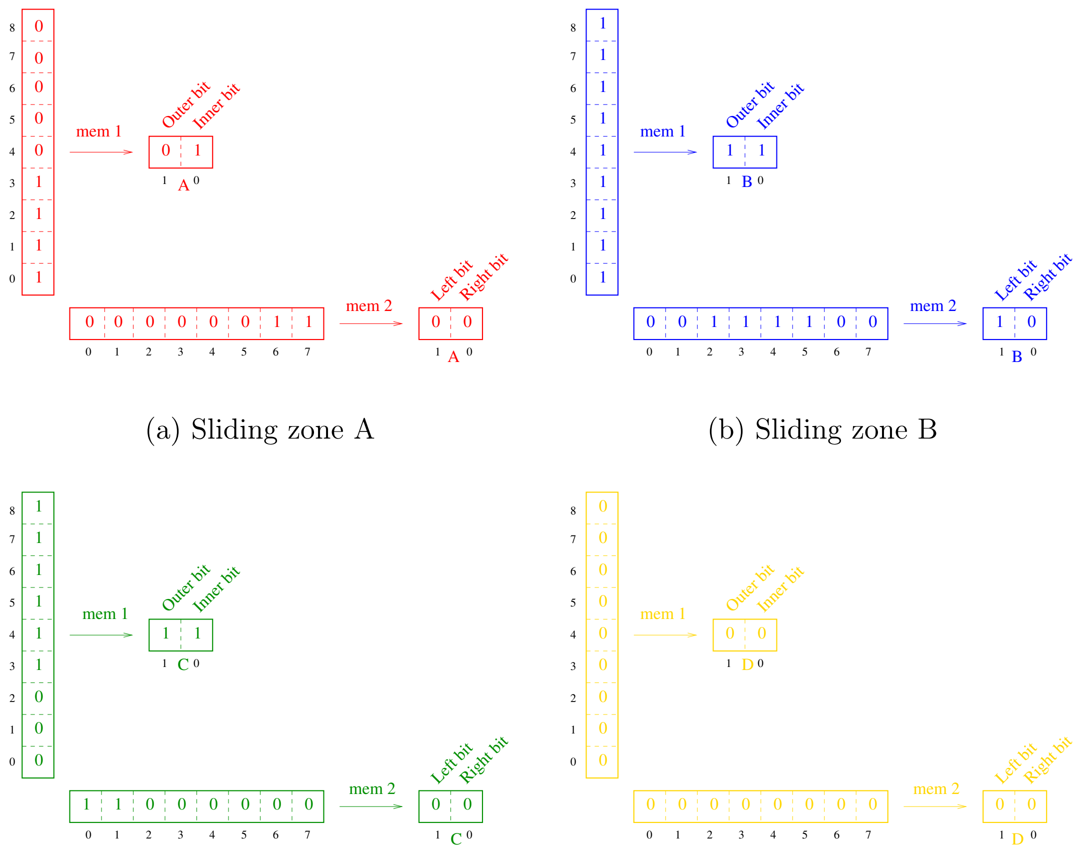
Then, the information of the 4 sliding zones is used to build the finale bitset for **a zone**. The vertical information of each sliding zone (4 SZ $\times$ 2 bits) is concatenate in an 8 bits bitset called the **vertical sliding zones bitset**. The horizontal information of each sliding zone (4 SZ $\times$ 2 bits) is concatenate in an 8 bits bitset called the **horizontal sliding zones bitset**.

The goal is to get 5 bits **per zone** which are 2 bits for **vertical tracker status per zone (VTSZ)** and 3 bits for **horizontal tracker status per zone (HTSZ)**.

The construction of the VTSZ bitset (2 bits) is based on the vertical sliding zones bitset (of the 4 sliding zones A,B,C,D). The construction of the HTSZ bitset (3 bits) is based on the horizontal sliding zones bitset (of the 4 sliding zones A,B,C,D). Three programmable memories are used to do the computing.

Memory 3 is for the VTSZ bitset and it is an A8D2 memory. The building of memory 3 is described in appendix D. Memory 4 is for the HTSZ bitset and it is an A6D2 memory because 2 bits of the horizontal sliding zones bitset are not used. The building of memory 4 is described in appendix E.

Memory 5 is used when it is impossible to use memory 4 because there is not enough horizontal information (for example just 2 rows hits but 8 layers). Each track must have a horizontal position so, vertical information of each sliding zone can be used to determine the horizontal location of the track. The vertical sliding zones bitset of 8 bits is used to construct the HTSZ and memory 5 do the computing. It is an A8D3 memory and the building of this memory is described in appendix F. All of these computing are showed in figure 16.



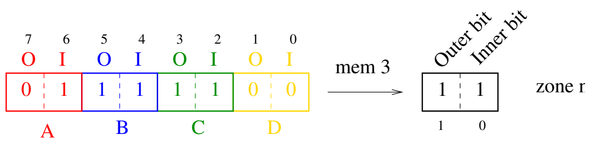
(a) Sliding zone A

(b) Sliding zone B

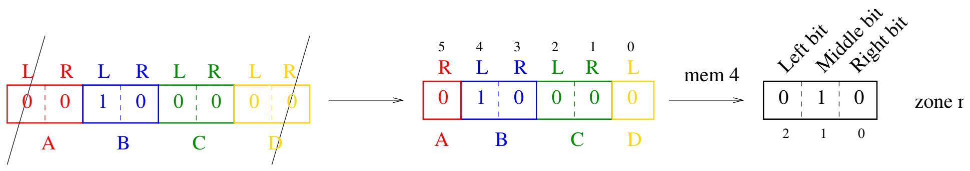
(c) Sliding zone C

(d) Sliding zone D

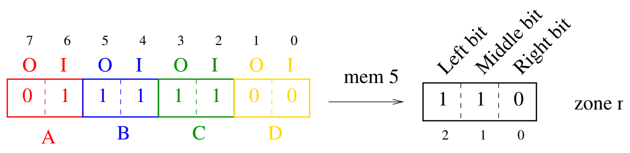
Figure 15: Projection on rows and layers on each sliding zone. Programmable memory 1 and 2 are used to fill the D2 responses.



(a) Vertical sliding zone bitset to vertical tracker status per zone (VTSZ) using memory 3



(b) Horizontal sliding zone bitset to horizontal tracker status per zone (HTSZ) using memory 4



(c) Vertical sliding zone bitset to horizontal tracker status per zone (HTSZ) using memory 5. It is used only if the Horizontal sliding zone bitset is empty.

Figure 16: The strategy to fill the response for a zone based on sliding zone information.

Near source zone information

It is additional information and it is not use for a track recognition but for rejection. The near source zone bits are set when one Geiger cells (or more) is set off in the near source zone. The zone called the near source is defined as the four layers closer to the source. If, a Geiger cell is set off in this region, a corresponding near source bit will be set. For each zone, there are 2 bits of near source zone, one right and one left. They are presented in figure 17.

- Near source zone left bit : [6 × 4 Geiger cells] (dark blue region)
- Near source zone right bit : [6 × 4 Geiger cells] (light blue region)

It will be use to search delayed Geiger cell close to the source foil at the ultimate threshold (1 Geiger cell only).

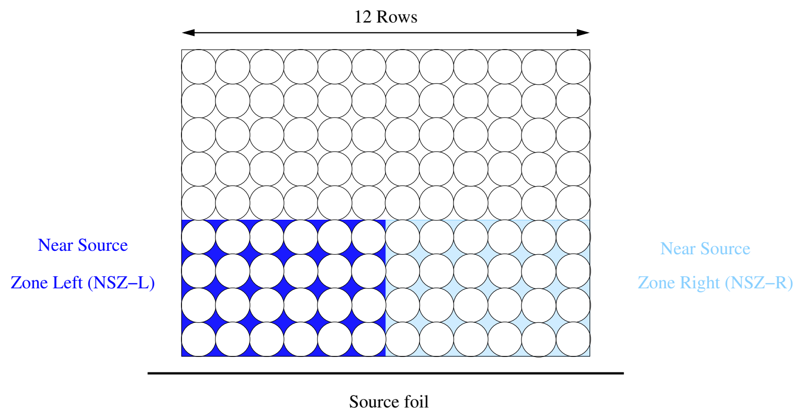


Figure 17: Near source zone information. In light blue, the region of the right bit near source zone and in dark blue, the region of the left bit near source zone.

Tracker finale data bit per zone @ 1600ns

This bitset of 7 bits is represented in figure 18 and composed by :

- horizontal information : [3 bits]
  - Left bit
  - Middle bit
  - Right bit
- vertical information : [2 bits]
  - Outer bit
  - Inner bit
- near source zone (NSZ) information : [2 bits]
  - Left NSZ bit
  - Right NSZ bit

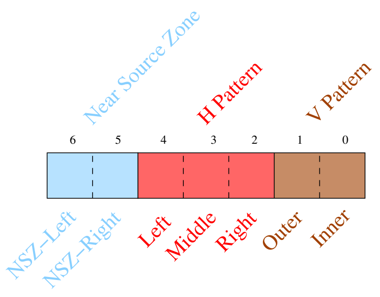


Figure 18: The final tracker bitset for each tracker zone

Tracker status

For the tracker, the information is reduced to 7 bits per zone. There are 20 zones so the full tracker status is represented with 140 bits. A final bitset of 182 bits called a tracker record is construct each 1600ns. This bitset is represented in figure 19 and composed by :

- tracker status per zone : 20 $\times$ 7 bits : [140 bits]

- zoning word pattern : [20 bits]
- zoning word near source : [20 bits]
- single side : [1 bit]

  Value 0 Zones of both sides are activated

  Value 1 Only zones of the same side are activated

- Tracker finale decision [1 bit] : based on configurable criteria

  Value 0 For this clock tick the tracker L1 decision is not passed

  Value 1 For this clock tick the tracker L1 decision is passed

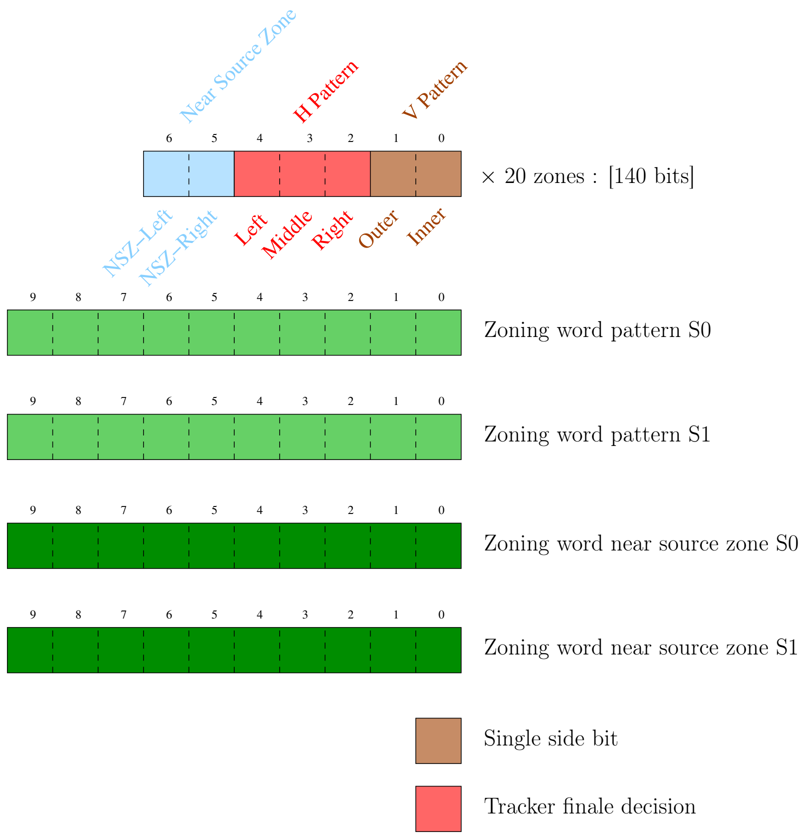


Figure 19: The final tracker record bitset [182 bits] produced each 1600ns

4 Coincidence trigger strategy in Trigger Board (TB)

There are 2 types of trigger coincidences. The first is the coincidence between calorimeter and tracker for a prompt event. The second is between two successive events. This type of coincidence is here to take into account the delayed alpha from the ^{214}Bi ^{214}Po decay.

Rescaling calorimeter at 1600ns

A calorimeter record and a tracker record are build and described in section 2 and section 3. The main issue of these records is the fact that they are not in the same clock. Calorimeter record is construct each 25ns whereas a tracker record is construct each 1600ns.

The objective of this algorithm is to have a trigger decision each 1600ns. The first thing to do is to 'rescale' the calorimeter record at 1600ns.

warning: Particular case : when calo summary record is split on clock tick 63/64/65 between 2 CT 1600

Thanks to the calorimeter algorithm, if one calo record bitset is not empty, four calo summary records are created (clock tick n, n+1, n+2 and n+3) with updates if it necessary. Calo summary records are tagged in time and it is possible to know in which clock tick 1600ns it is. If the calo decision bit is set to true, the first calo summary record (n at 25ns) create a new bitset called a coincidence calo record for a given CT number **@ 1600ns**. It is composed by the same information as the calo summary record (35 bits). The second calo summary record (n+1 at 25ns) will just update this coincidence calo record because it is already created and it is in the same clock tick.

When a coincidence calo record is created, a configurable gate is open (10 $\times$ 1600ns (8 μs)). The first coincidence calo record is duplicated during 10 clock ticks. This calorimeter gate is here to allow Geiger signal deployment (physics of the Geiger signal $\simeq$ 10 μs) due to the drift and take into account the shift in the electronics (maybe until 5 μs).

note: Issue atm : what if a new calorimeter is hit during this 16 μs ? We have to update coincidence calo records to not have a dead time of the calorimeter (not implemented yet)

note: This issue is going to be fixed, if a new calorimeter hit is coming during this 16 μs, it will update the existing calo coincidence record and create new calo coincidence records up to 16 μs

The figure 20 presents the chronogram for the coincidence decision. We have to rescale calorimeter at 1600 ns and then, wait the tracker deployment. The coincidence calorimeter gate is open when the L1 is sent. If a coincidence is find, the coincidence decision L2 is sent to the boards.

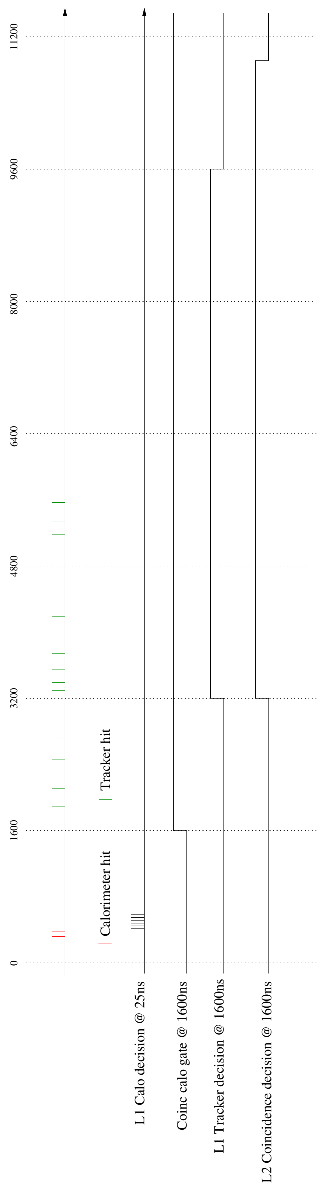


Figure 20: L1 and L2 decision chronogram and rescaling calorimeter at 1600 ns.

4.1 CAlorimeter TRacker COincidence (CARACO)

So now, each 1600ns, a geiger record is synchronized with a coincidence calo record. The spatial coincidence is now possible. The strategy for spatial coincidence is to compare each horizontal pattern zone information of the tracker (3 bits) and try to associate it with a coincidence calo record.

The association of a tracker zone information and a calorimeter zone is presented in table 2. Left pattern for a tracker zone can be associated with the n-1 / n calorimeter zone. Middle pattern with the actual zone and right pattern with the n / n+1 calorimeter zone.

An example is show in figure 21 which present a track with a calorimeter hit. Thanks to the rescaling at 1600ns, it is possible to do the coincidence between tracker and calorimeter.

Tracker zone information	Calorimeter zone
Left	n-1/n
Middle	n
Right	n/n+1

Table 2: Searching for spatial coincidences between tracker and calorimeter depending of tracker zone pattern. Each zone of the tracker have to be compared.

If a coincidence is find between calorimeter and tracker, the trigger board can send the level 1 trigger to begin the acquisition. Furthermore, a new bitset is created and it is called a coincidence event record. It sums up the information of tracker and calorimeter record. It also have an information concerning the trigger mode. For this type of coincidence (tracko-calo association), the trigger mode is 1 (CARACO).

This bitset of coincidence event record have to be **stored during 1 ms** (fixed by the decay of ^{214}Po, $T_{1/2}(^{214}\text{Po}) = 164\ \mu\text{s}$)) in a new bitset call the *previous event record*. The previous event record is stored during 625 clock ticks of 1600ns and have an internal counter. For the moment, it is not possible to trigger a second time when the counter is > 619 because it is mandatory to let the first physical event to finish. This previous event record will be useful for the other type of coincidences and the search for delayed alphas.

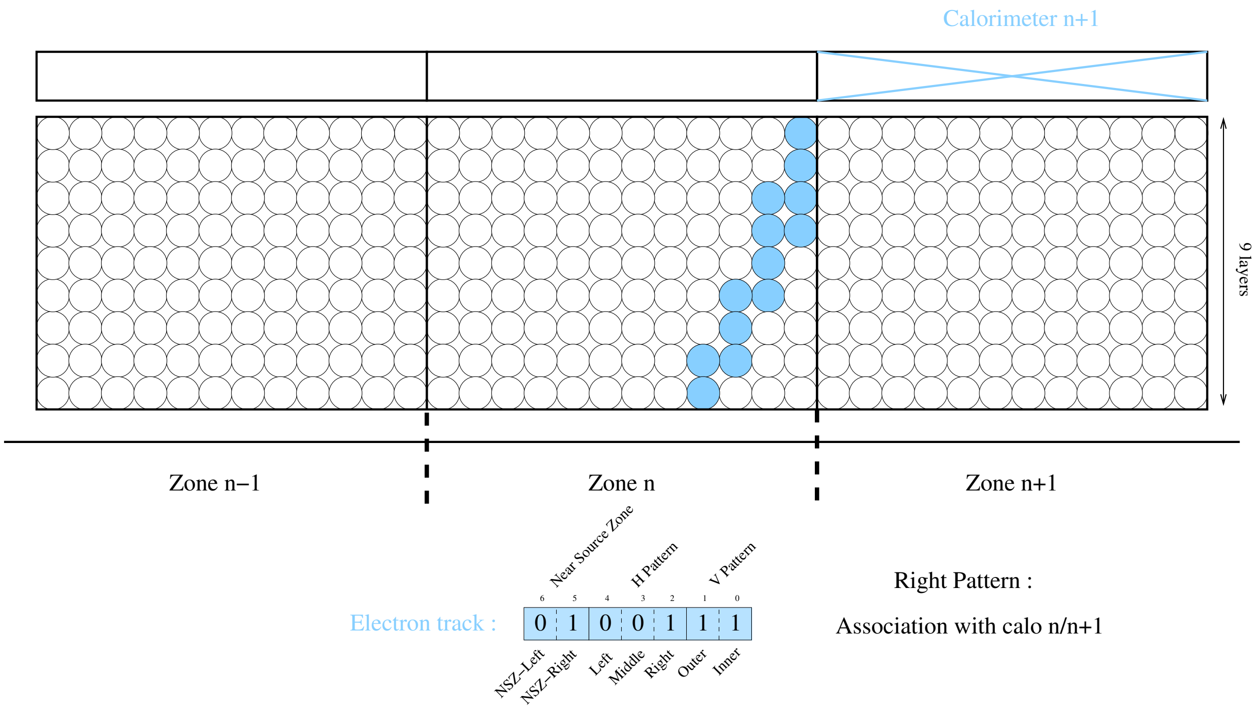


Figure 21: Association between a track and a calorimeter hit at 1600ns.

4.2 Alpha Previous Event coincidence (APE)

It is important to keep the bitset called the previous event record because, if there is a delayed alpha, it will set off some Geiger cells but no calorimeters. The first coincidence (CARACO) is not possible and it is possible to lose this event. The trigger decision is set to 0 just after the last time the CARACO coincidence is ok.

The APE coincidence will try to do an association between the current event with the previous event record bitset stored in the trigger board. The timing of the 'new' event has to be less than 1 ms to do this coincidence so the associated counter of a previous event record have to be different of 0. For this type of coincidence only tracker information (for the moment) will be compared to the previous event record. Same as the CARACO algorithm, each tracker zones of the current event will try to associate with the previous event tracker pattern. As presented in table 3, if the tracker pattern (3 bits) of the current event is tagged as left / middle or right, it will be compared with different positions and zone.

Current event / Previous event	Left	Middle	Right
Side 0	n-1_R/n_L/n_M	n_L/n_M/n_R	n_M/n_R/n+1_L
Side 1	n-1_R/n_L/n_M	n_L/n_M/n_R	n_M/n_R/n+1_L

Table 3: Association tracker / tracker pattern for APE coincidence

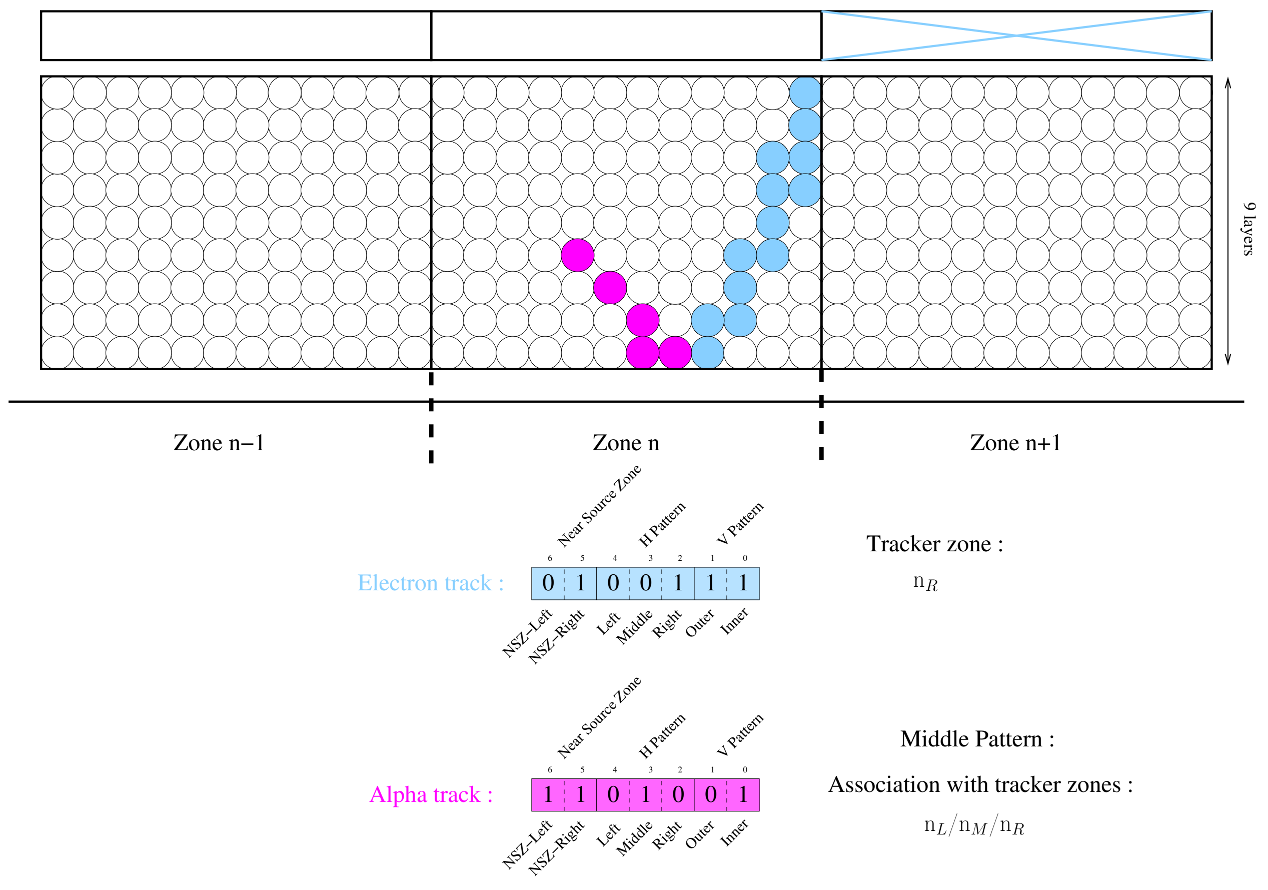


Figure 22: Association between a delayed short track with the previous event, the electron track thanks to the APE trigger.

4.3 Delayed Alpha Veto Event coincidence (DAVE)

The DAVE coincidence is here to accept delayed alpha event close to the source after an other event. It is very critic to have the delayed alpha after the electron and specially if it is close to the source. The problem of delayed alpha is that sometimes just one or two Geiger cells are set off. The threshold to tag a track as a pattern is set to 3 Geiger cells. So we have to keep the information (in the previous event record) near source to accept these type of events during 1 ms.

Each tracker zone have 2 bits called near source zone left and near source zone right. The DAVE coincidence will associate the current near source information with the previous event record information. The association between current event and the previous event are presented in table 4.

Current event / Previous event	Near source left	Near source right
Side 0	n-1_{NSR}/n_{NSL}/n_{NSR}	n_{NSL}/n_{NSR}/n+1_{NSL}
Side 1	n-1_{NSR}/n_{NSL}/n_{NSR}	n_{NSL}/n_{NSR}/n+1_{NSL}

Table 4: Association tracker pattern and near source zone bit for DAVE coincidence

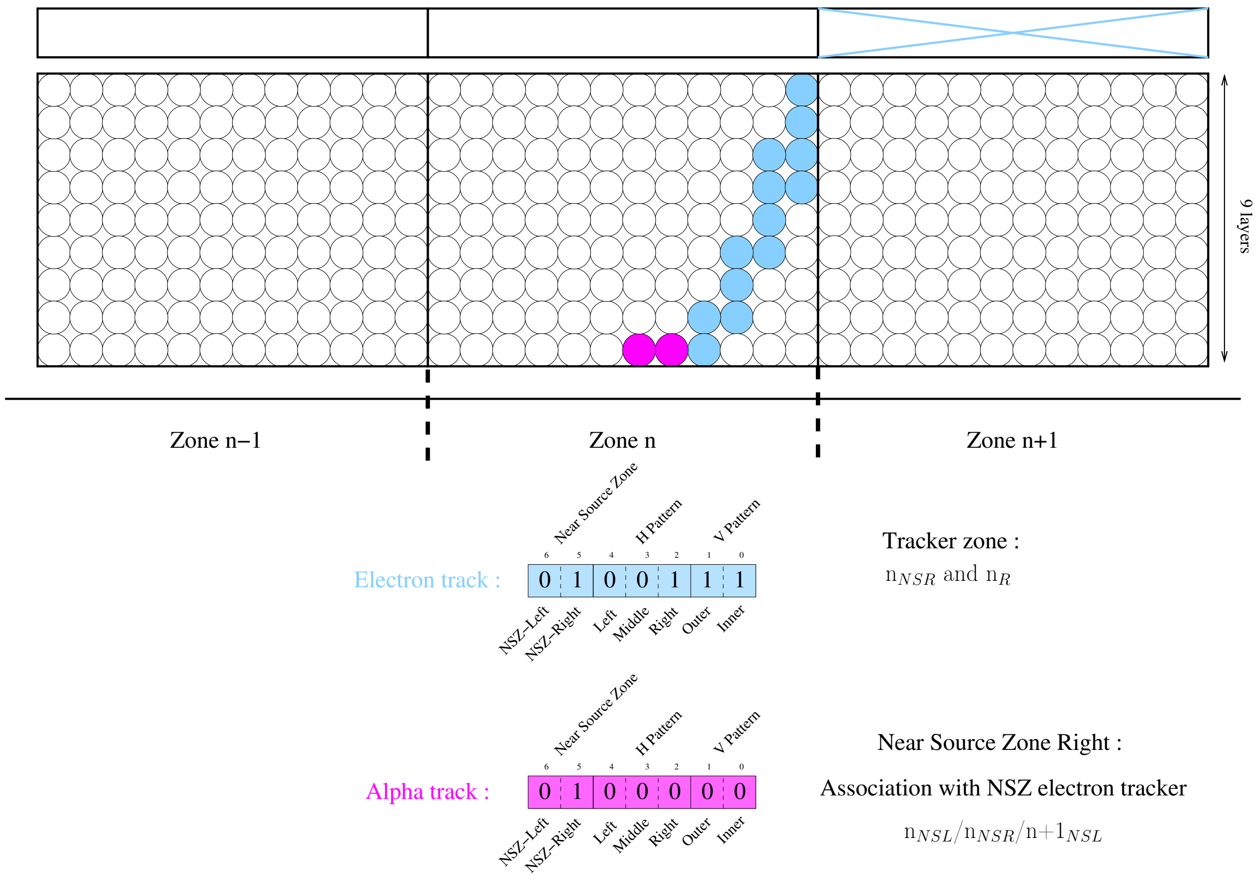


Figure 23: Association between 2 delayed Geiger cells close to the source foil with the previous event, the electron track to do a VETO on this event thanks to the DAVE trigger.

References

[1] Y.Lemière. *SuperNemo Demonstrator Electronics Reference*. In *DocDB 2557*, 2015.

Appendix A Sliding zone concept : special cases zone 0, 5 and 9

There are 3 special cases for the sliding zone concept due to the beginning, the end and the center of the tracker.

Special case 1 : zone 0

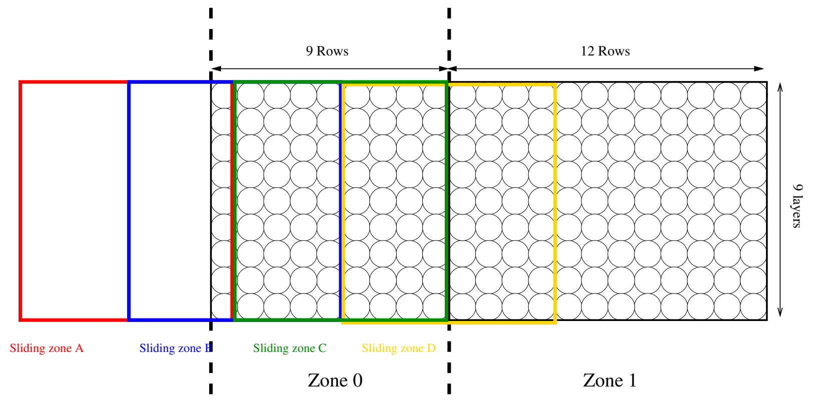


Figure 24: Sliding zones for the zone 0 (begining of the tracker)

Special case 2 : zone 5

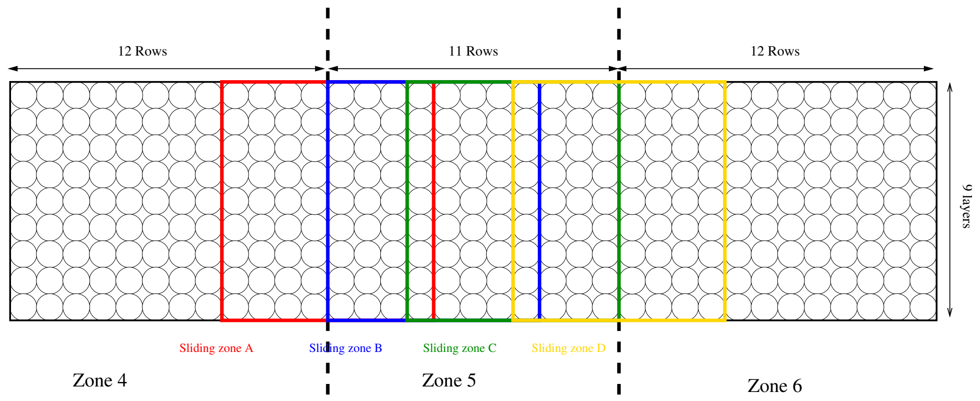


Figure 25: Sliding zones for the zone 5 (middle of the tracker)

Special case 3 : zone 9

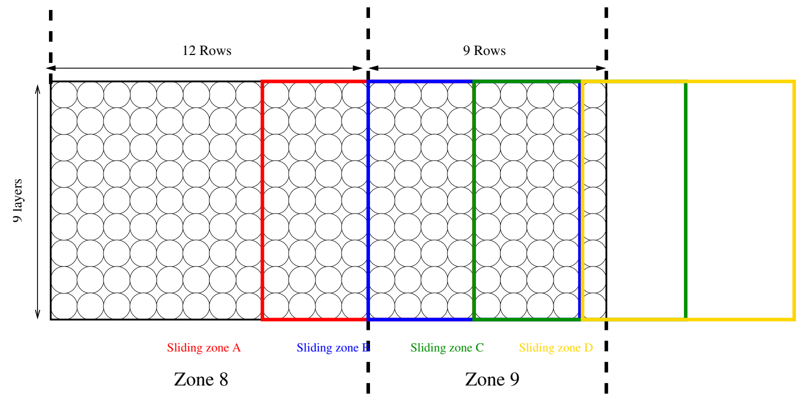


Figure 26: Sliding zones for the zone 9 (end of the tracker)

Appendix B Programmable memory 1 : sliding zone vertical memory A9D2

The goal of the programmable memory 1 is to compute the **projection vertical bitset** of **each sliding zone** into two bits called the inner bit and the outer bit. This memory is an A9D2 memory.

How to build this programmation table ?

There are 4 possibilities for the D2 response **for a sliding zone** :

00	: Vertical Void (VVOID)
01	: Vertical Inner (VINNER)
10	: Vertical Outer (VOUTER)
11	: Vertical Full (VFULL)

To do this computing, criteria to determine what is inner, outer and full are the following :

A full track can be searched between the first layer [0] and the last layer [8]. To tag a track as a full track, the multiplicity of hit layers has to be superior at 6 (on 9 max).

VFULL $\in$ layer position : [0-8] and VFULL multiplicity ≥ 6

An inner track can be searched between the first layer [0] and the layer [4]. To tag a track as an inner track, the multiplicity of hit layers has to be included between 3 and 5.

VINNER $\in$ layer position : [0-4] and VINNER multiplicity ≥ 3 & ≤ 5

An outer track can be searched between the layer [5] and the last layer [8]. To tag a track as an outer track, the multiplicity of hit layers has to be included between 3 and 5.

VOUTER $\in$ layer position : [5-8] and VOUTER multiplicity ≥ 3 & ≤ 5

```
000000000 => 00 000000001 => 00 000000010 => 00 000000011 => 00 000000100 => 00 000000101 => 00 000000110 => 00
000000111 => 01 000001000 => 00 000001001 => 00 000001010 => 00 000001011 => 01 000001100 => 00 000001101 => 01
000001110 => 01 000001111 => 01 000010000 => 00 000010001 => 00 000010010 => 00 000010011 => 01 000010100 => 00
000010101 => 01 000010110 => 01 000010111 => 01 000011000 => 00 000011001 => 01 000011010 => 01 000011011 => 01
000011100 => 01 000011101 => 01 000011110 => 01 000011111 => 01 000100000 => 00 000100001 => 00 000100010 => 00
000100011 => 01 000100100 => 00 000100101 => 01 000100110 => 01 000100111 => 01 000101000 => 00 000101001 => 01
000101010 => 01 000101011 => 01 000101100 => 01 000101101 => 01 000101110 => 01 000101111 => 01 000110000 => 00
000110001 => 01 000110010 => 01 000110011 => 01 000110100 => 01 000110101 => 01 000110110 => 01 000110111 => 01
000111000 => 01 000111001 => 01 000111010 => 01 000111011 => 01 000111100 => 01 000111101 => 01 000111110 => 01
000111111 => 11 001000000 => 00 001000001 => 00 001000010 => 00 001000011 => 00 001000100 => 00 001000101 => 00
001000110 => 00 001000111 => 01 001001000 => 00 001001001 => 00 001001010 => 00 001001011 => 01 001001100 => 00
001001101 => 01 001001110 => 01 001001111 => 01 001010000 => 00 001010001 => 00 001010010 => 00 001010011 => 01
001010100 => 00 001010101 => 01 001010110 => 01 001010111 => 01 001011000 => 00 001011001 => 01 001011010 => 01
001011011 => 01 001011100 => 01 001011101 => 01 001011110 => 01 001011111 => 11 001100000 => 00 001100001 => 00
001100010 => 00 001100011 => 01 001100100 => 00 001100101 => 01 001100110 => 01 001100111 => 01 001101000 => 00
001101001 => 01 001101010 => 01 001101011 => 01 001101100 => 01 001101101 => 01 001101110 => 01 001101111 => 11
001110000 => 10 001110001 => 01 001110010 => 01 001110011 => 01 001110100 => 01 001110101 => 01 001110110 => 01
001110111 => 11 001111000 => 01 001111001 => 01 001111010 => 01 001111011 => 11 001111100 => 01 001111101 => 11
001111110 => 11 001111111 => 11 010000000 => 00 010000001 => 00 010000010 => 00 010000011 => 00 010000100 => 00
010000101 => 00 010000110 => 00 010000111 => 01 010001000 => 00 010001001 => 00 010001010 => 00 010001011 => 01
010001100 => 00 010001101 => 01 010001110 => 01 010001111 => 01 010010000 => 00 010010001 => 00 010010010 => 00
010010011 => 01 010010100 => 00 010010101 => 01 010010110 => 01 010010111 => 01 010011000 => 00 010011001 => 01
010011010 => 01 010011011 => 01 010011100 => 01 010011101 => 01 010011110 => 01 010011111 => 11 010100000 => 00
010100001 => 00 010100010 => 00 010100011 => 01 010100100 => 00 010100101 => 01 010100110 => 01 010100111 => 01
010101000 => 00 010101001 => 01 010101010 => 01 010101011 => 01 010101100 => 01 010101101 => 01 010101110 => 01
010101111 => 11 010110000 => 10 010110001 => 01 010110010 => 01 010110011 => 01 010110100 => 01 010110101 => 01
010110110 => 01 010110111 => 11 010111000 => 01 010111001 => 01 010111010 => 01 010111011 => 11 010111100 => 01
010111101 => 11 010111110 => 11 010111111 => 11 011000000 => 00 011000001 => 00 011000010 => 00 011000011 => 00
011000100 => 00 011000101 => 00 011000110 => 00 011000111 => 01 011001000 => 00 011001001 => 00 011001010 => 00
011001011 => 01 011001100 => 00 011001101 => 01 011001110 => 01 011001111 => 11 011010000 => 10 011010001 => 10
011010010 => 10 011010011 => 01 011010100 => 10 011010101 => 01 011010110 => 01 011010111 => 11 011011000 => 10
011011001 => 01 011011010 => 01 011011011 => 11 011011100 => 01 011011101 => 11 011011110 => 11 011011111 => 11
011100000 => 10 011100001 => 10 011100010 => 10 011100011 => 01 011100100 => 10 011100101 => 01 011100110 => 01
011100111 => 11 011101000 => 10 011101001 => 01 011101010 => 01 011101011 => 11 011101100 => 01 011101101 => 11
011101110 => 11 011101111 => 11 011110000 => 10 011110001 => 01 011110010 => 01 011110011 => 11 011110100 => 01
011110101 => 11 011110110 => 11 011110111 => 11 011111000 => 01 011111001 => 11 011111010 => 11 011111011 => 11
011111100 => 11 011111101 => 11 011111110 => 11 011111111 => 11 100000000 => 00 100000001 => 00 100000010 => 00
100000011 => 00 100000100 => 00 100000101 => 00 100000110 => 00 100000111 => 01 100001000 => 00 100001001 => 00
100001010 => 00 100001011 => 01 100001100 => 00 100001101 => 01 100001110 => 01 100001111 => 01 100010000 => 00
100010001 => 00 100010010 => 00 100010011 => 01 100010100 => 00 100010101 => 01 100010110 => 01 100010111 => 01
100011000 => 00 100011001 => 01 100011010 => 01 100011011 => 01 100011100 => 01 100011101 => 01 100011110 => 01
100011111 => 11 100100000 => 00 100100001 => 00 100100010 => 00 100100011 => 01 100100100 => 00 100100101 => 01
100100110 => 01 100100111 => 01 100101000 => 00 100101001 => 01 100101010 => 01 100101011 => 01 100101100 => 01
100101101 => 01 100101110 => 01 100101111 => 11 100110000 => 10 100110001 => 01 100110010 => 01 100110011 => 01
100110100 => 01 100110101 => 01 100110110 => 01 100110111 => 11 100111000 => 01 100111001 => 01 100111010 => 01
100111011 => 11 100111100 => 01 100111101 => 11 100111110 => 11 100111111 => 11 101000000 => 00 101000001 => 00
101000010 => 00 101000011 => 00 101000100 => 00 101000101 => 00 101000110 => 00 101000111 => 01 101001000 => 00
101001001 => 00 101001010 => 00 101001011 => 01 101001100 => 00 101001101 => 01 101001110 => 01 101001111 => 11
101010000 => 10 101010001 => 10 101010010 => 10 101010011 => 01 101010100 => 10 101010101 => 01 101010110 => 01
101010111 => 11 101011000 => 10 101011001 => 01 101011010 => 01 101011011 => 11 101011100 => 01 101011101 => 11
101011110 => 11 101011111 => 11 101100000 => 10 101100001 => 10 101100010 => 10 101100011 => 01 101100100 => 10
101100101 => 01 101100110 => 01 101100111 => 11 101101000 => 10 101101001 => 01 101101010 => 01 101101011 => 11
101101100 => 01 101101101 => 11 101101110 => 11 101101111 => 11 101110000 => 10 101110001 => 01 101110010 => 01
101110011 => 11 101110100 => 01 101110101 => 11 101110110 => 11 101110111 => 11 101111000 => 01 101111001 => 11
101111010 => 11 101111011 => 11 101111100 => 11 101111101 => 11 101111110 => 11 101111111 => 11 110000000 => 00
110000001 => 00 110000010 => 00 110000011 => 00 110000100 => 00 110000101 => 00 110000110 => 00 110000111 => 01
110001000 => 00 110001001 => 00 110001010 => 00 110001011 => 01 110001100 => 00 110001101 => 01 110001110 => 01
110001111 => 11 110010000 => 10 110010001 => 10 110010010 => 10 110010011 => 01 110010100 => 10 110010101 => 01
110010110 => 01 110010111 => 11 110011000 => 10 110011001 => 01 110011010 => 01 110011011 => 11 110011100 => 01
110011101 => 11 110011110 => 11 110011111 => 11 110100000 => 10 110100001 => 10 110100010 => 10 110100011 => 01
110100100 => 10 110100101 => 01 110100110 => 01 110100111 => 11 110101000 => 10 110101001 => 01 110101010 => 01
110101011 => 11 110101100 => 01 110101101 => 11 110101110 => 11 110101111 => 11 110110000 => 10 110110001 => 01
110110010 => 01 110110011 => 11 110110100 => 01 110110101 => 11 110110110 => 11 110110111 => 11 110111000 => 01
110111001 => 11 110111010 => 11 110111011 => 11 110111100 => 11 110111101 => 11 110111110 => 11 110111111 => 11
111000000 => 10 111000001 => 10 111000010 => 10 111000011 => 10 111000100 => 10 111000101 => 10 111000110 => 10
111000111 => 11 111001000 => 10 111001001 => 10 111001010 => 10 111001011 => 11 111001100 => 10 111001101 => 11
111001110 => 11 111001111 => 11 111010000 => 10 111010001 => 10 111010010 => 10 111010011 => 11 111010100 => 10
111010101 => 11 111010110 => 11 111010111 => 11 111011000 => 10 111011001 => 11 111011010 => 11 111011011 => 11
111011100 => 11 111011101 => 11 111011110 => 11 111011111 => 11 111100000 => 10 111100001 => 10 111100010 => 10
111100011 => 11 111100100 => 10 111100101 => 11 111100110 => 11 111100111 => 11 111101000 => 10 111101001 => 11
111101010 => 11 111101011 => 11 111101100 => 11 111101101 => 11 111101110 => 11 111101111 => 11 111110000 => 10
111110001 => 11 111110010 => 11 111110011 => 11 111110100 => 11 111110101 => 11 111110110 => 11 111110111 => 11
111111000 => 11 111111001 => 11 111111010 => 11 111111011 => 11 111111100 => 11 111111101 => 11 111111110 => 11
111111111 => 11
```

Figure 27: Sliding zone vertical memory A9D2 : programmable memory 1

Appendix C Programmable memory 2 : sliding zone horizontal memory A8D2

The goal of the programmable memory 2 is to compute the **projection horizontal bitset** of **each sliding zone** into two bits called the left bit and the right bit. This memory is an A8D2 memory.

How to build this programmation table ?

There are 4 possibilities for the D2 response **for a sliding zone** :

00 : Horizontal Void (HVOID)

01 : Horizontal Narrow Right (HNRIGHT)

10 : Horizontal Narrow Left (HNLEFT)

11 : Horizontal Wide (HWIDE)

To do this computing, patterns are searched in the 8 bits bitset. 4 Patterns can be searched in the 8 bits to tag a the sliding zone as void, left, right or wide. The patterns are :

- "1111"
- "111"
- "1101"
- "1011"

These patterns can be searched by the left or by the right on the bitset. The left research is privileged compared to the right research. This research of patterns works only if a multiplicity of 3 is needed to construct the horizontal response.

```
00000000 => 00 00000001 => 00 00000010 => 00 00000011 => 00 00000100 => 00 00000101 => 00
00000110 => 00 00000111 => 01 00001000 => 00 00001001 => 00 00001010 => 00 00001011 => 01
00001100 => 00 00001101 => 01 00001110 => 01 00001111 => 01 00010000 => 00 00010001 => 00
00010010 => 00 00010011 => 00 00010100 => 00 00010101 => 00 00010110 => 01 00010111 => 01
00011000 => 00 00011001 => 00 00011010 => 01 00011011 => 01 00011100 => 01 00011101 => 01
00011110 => 01 00011111 => 01 00100000 => 00 00100001 => 00 00100010 => 00 00100011 => 00
00100100 => 00 00100101 => 00 00100110 => 00 00100111 => 01 00101000 => 00 00101001 => 00
00101010 => 00 00101011 => 01 00101100 => 10 00101101 => 01 00101110 => 01 00101111 => 01
00110000 => 00 00110001 => 00 00110010 => 00 00110011 => 00 00110100 => 10 00110101 => 10
00110110 => 01 00110111 => 01 00111000 => 10 00111001 => 10 00111010 => 01 00111011 => 01
00111100 => 10 00111101 => 01 00111110 => 01 00111111 => 11 01000000 => 00 01000001 => 00
01000010 => 00 01000011 => 00 01000100 => 00 01000101 => 00 01000110 => 00 01000111 => 01
01001000 => 00 01001001 => 00 01001010 => 00 01001011 => 01 01001100 => 00 01001101 => 01
01001110 => 01 01001111 => 01 01010000 => 00 01010001 => 00 01010010 => 00 01010011 => 00
01010100 => 00 01010101 => 00 01010110 => 01 01010111 => 01 01011000 => 10 01011001 => 10
01011010 => 11 01011011 => 11 01011100 => 10 01011101 => 11 01011110 => 11 01011111 => 11
01100000 => 00 01100001 => 00 01100010 => 00 01100011 => 00 01100100 => 00 01100101 => 00
01100110 => 00 01100111 => 01 01101000 => 10 01101001 => 10 01101010 => 10 01101011 => 11
01101100 => 10 01101101 => 11 01101110 => 11 01101111 => 11 01110000 => 10 01110001 => 10
01110010 => 10 01110011 => 10 01110100 => 10 01110101 => 10 01110110 => 11 01110111 => 11
01111000 => 10 01111001 => 10 01111010 => 11 01111011 => 11 01111100 => 10 01111101 => 11
01111110 => 11 01111111 => 11 10000000 => 00 10000001 => 00 10000010 => 00 10000011 => 00
10000100 => 00 10000101 => 00 10000110 => 00 10000111 => 01 10001000 => 00 10001001 => 00
10001010 => 00 10001011 => 01 10001100 => 00 10001101 => 01 10001110 => 01 10001111 => 01
10010000 => 00 10010001 => 00 10010010 => 00 10010011 => 00 10010100 => 00 10010101 => 00
10010110 => 01 10010111 => 01 10011000 => 00 10011001 => 00 10011010 => 01 10011011 => 01
10011100 => 01 10011101 => 01 10011110 => 01 10011111 => 11 10100000 => 00 10100001 => 00
10100010 => 00 10100011 => 00 10100100 => 00 10100101 => 00 10100110 => 00 10100111 => 01
10101000 => 00 10101001 => 00 10101010 => 00 10101011 => 01 10101100 => 10 10101101 => 11
10101110 => 11 10101111 => 11 10110000 => 10 10110001 => 10 10110010 => 10 10110011 => 10
10110100 => 10 10110101 => 10 10110110 => 11 10110111 => 11 10111000 => 10 10111001 => 10
10111010 => 11 10111011 => 11 10111100 => 10 10111101 => 11 10111110 => 11 10111111 => 11
11000000 => 00 11000001 => 00 11000010 => 00 11000011 => 00 11000100 => 00 11000101 => 00
11000110 => 00 11000111 => 01 11001000 => 00 11001001 => 00 11001010 => 00 11001011 => 01
11001100 => 00 11001101 => 01 11001110 => 01 11001111 => 11 11010000 => 10 11010001 => 10
11010010 => 10 11010011 => 10 11010100 => 10 11010101 => 10 11010110 => 11 11010111 => 11
11011000 => 10 11011001 => 10 11011010 => 11 11011011 => 11 11011100 => 10 11011101 => 11
11011110 => 11 11011111 => 11 11100000 => 10 11100001 => 10 11100010 => 10 11100011 => 10
11100100 => 10 11100101 => 10 11100110 => 10 11100111 => 11 11101000 => 10 11101001 => 10
11101010 => 10 11101011 => 11 11101100 => 10 11101101 => 11 11101110 => 11 11101111 => 11
11110000 => 10 11110001 => 10 11110010 => 10 11110011 => 11 11110100 => 10 11110101 => 11
11110110 => 11 11110111 => 11 11111000 => 10 11111001 => 11 11111010 => 11 11111011 => 11
11111100 => 11 11111101 => 11 11111110 => 11 11111111 => 11
```

Figure 28: Sliding zone horizontal memory A8D2 : programmable memory 2

Appendix D Programmable memory 3 : zone vertical memory A8D2

The goal of the programmable memory 3 is to compute the **vertical sliding zones bitset** which is the concatenation of the 2 bits of each sliding zone into a two bits bitset called the **vertical tracker status per zone (VTSZ)**. This memory is an A8D2 memory.

How to build this programmation table ?

There are 4 possibilities for the D2 response **for a zone** :

00 : Vertical Void (VVOID)

01 : Vertical Inner (VINNER)

10 : Vertical Outer (VOUTER)

11 : Vertical Full (VFULL)

This response is based on the vertical information of each sliding zone A,B,C,D. To do this computing, an 'OR' operation is done on each inner bit of each sliding zone to activate or not the inner bit for the zone. An 'OR' operation is done on each outer bit of each sliding zone to activate or not the outer bit for the zone.

A_{inner} OR B_{inner} OR C_{inner} OR D_{inner} $\rightarrow$ Zone inner bit : 1
A_{outer} OR B_{outer} OR C_{outer} OR D_{outer} $\rightarrow$ Zone outer bit : 1

warning: special case : 10000001 and 01000010, can't be a VFULL track because sliding zone A and D don't touch. The priority is given to VINNER so these special cases are not VFULL but VINNER.

```
00000000 => 00 00000001 => 01 00000010 => 10 00000011 => 11 00000100 => 01 00000101 => 01
00000110 => 11 00000111 => 11 00001000 => 10 00001001 => 11 00001010 => 10 00001011 => 11
00001100 => 11 00001101 => 11 00001110 => 11 00001111 => 11 00010000 => 01 00010001 => 01
00010010 => 11 00010011 => 11 00010100 => 01 00010101 => 01 00010110 => 11 00010111 => 11
00011000 => 11 00011001 => 11 00011010 => 11 00011011 => 11 00011100 => 11 00011101 => 11
00011110 => 11 00011111 => 11 00100000 => 10 00100001 => 11 00100010 => 10 00100011 => 11
00100100 => 11 00100101 => 11 00100110 => 11 00100111 => 11 00101000 => 10 00101001 => 11
00101010 => 10 00101011 => 11 00101100 => 11 00101101 => 11 00101110 => 11 00101111 => 11
00110000 => 11 00110001 => 11 00110010 => 11 00110011 => 11 00110100 => 11 00110101 => 11
00110110 => 11 00110111 => 11 00111000 => 11 00111001 => 11 00111010 => 11 00111011 => 11
00111100 => 11 00111101 => 11 00111110 => 11 00111111 => 11 01000000 => 01 01000001 => 01
01000010 => 01 01000011 => 11 01000100 => 01 01000101 => 01 01000110 => 11 01000111 => 11
01001000 => 11 01001001 => 11 01001010 => 11 01001011 => 11 01001100 => 11 01001101 => 11
01001110 => 11 01001111 => 11 01010000 => 01 01010001 => 01 01010010 => 11 01010011 => 11
01010100 => 01 01010101 => 01 01010110 => 11 01010111 => 11 01011000 => 11 01011001 => 11
01011010 => 11 01011011 => 11 01011100 => 11 01011101 => 11 01011110 => 11 01011111 => 11
01100000 => 11 01100001 => 11 01100010 => 11 01100011 => 11 01100100 => 11 01100101 => 11
01100110 => 11 01100111 => 11 01101000 => 11 01101001 => 11 01101010 => 11 01101011 => 11
01101100 => 11 01101101 => 11 01101110 => 11 01101111 => 11 01110000 => 11 01110001 => 11
01110010 => 11 01110011 => 11 01110100 => 11 01110101 => 11 01110110 => 11 01110111 => 11
01111000 => 11 01111001 => 11 01111010 => 11 01111011 => 11 01111100 => 11 01111101 => 11
01111110 => 11 01111111 => 11 10000000 => 10 10000001 => 01 10000010 => 10 10000011 => 11
10000100 => 11 10000101 => 11 10000110 => 11 10000111 => 11 10001000 => 10 10001001 => 11
10001010 => 10 10001011 => 11 10001100 => 11 10001101 => 11 10001110 => 11 10001111 => 11
10010000 => 11 10010001 => 11 10010010 => 11 10010011 => 11 10010100 => 11 10010101 => 11
10010110 => 11 10010111 => 11 10011000 => 11 10011001 => 11 10011010 => 11 10011011 => 11
10011100 => 11 10011101 => 11 10011110 => 11 10011111 => 11 10100000 => 10 10100001 => 11
10100010 => 10 10100011 => 11 10100100 => 11 10100101 => 11 10100110 => 11 10100111 => 11
10101000 => 10 10101001 => 11 10101010 => 10 10101011 => 11 10101100 => 11 10101101 => 11
10101110 => 11 10101111 => 11 10110000 => 11 10110001 => 11 10110010 => 11 10110011 => 11
10110100 => 11 10110101 => 11 10110110 => 11 10110111 => 11 10111000 => 11 10111001 => 11
10111010 => 11 10111011 => 11 10111100 => 11 10111101 => 11 10111110 => 11 10111111 => 11
11000000 => 11 11000001 => 11 11000010 => 11 11000011 => 11 11000100 => 11 11000101 => 11
11000110 => 11 11000111 => 11 11001000 => 11 11001001 => 11 11001010 => 11 11001011 => 11
11001100 => 11 11001101 => 11 11001110 => 11 11001111 => 11 11010000 => 11 11010001 => 11
11010010 => 11 11010011 => 11 11010100 => 11 11010101 => 11 11010110 => 11 11010111 => 11
11011000 => 11 11011001 => 11 11011010 => 11 11011011 => 11 11011100 => 11 11011101 => 11
11011110 => 11 11011111 => 11 11100000 => 11 11100001 => 11 11100010 => 11 11100011 => 11
11100100 => 11 11100101 => 11 11100110 => 11 11100111 => 11 11101000 => 11 11101001 => 11
11101010 => 11 11101011 => 11 11101100 => 11 11101101 => 11 11101110 => 11 11101111 => 11
11110000 => 11 11110001 => 11 11110010 => 11 11110011 => 11 11110100 => 11 11110101 => 11
11110110 => 11 11110111 => 11 11111000 => 11 11111001 => 11 11111010 => 11 11111011 => 11
11111100 => 11 11111101 => 11 11111110 => 11 11111111 => 11
```

Figure 29: Zone vertical memory A8D2 : programmable memory 3

Appendix E Programmable memory 4 : zone horizontal memory A6D3

The goal of the programmable memory 4 is to compute the **horizontal sliding zones bitset** which is the concatenation of the 2 bits of sliding zone B and C and the right bit of the zone A and the left bit of the zone B into a three bits bitset called the **horizontal tracker status per zone (HTSZ)**. This memory is an A6D3 memory.

How to build this programmation table ?

There are 8 possibilities for the D3 response **for a zone** :

000 : Horizontal Void (HVOID)

001 : Horizontal Right (HRIGHT)

010 : Horizontal Middle (HMIDDLE)

011 : Horizontal Right-Middle (HRIGHT-MIDDLE)

100 : Horizontal Left (HLEFT)

101 : Horizontal Left-Right (HLEFT-RIGHT)

110 : Horizontal Left-Middle (HLEFT-MIDDLE)

111 : Horizontal Wide (HWIDE)

This response is based on the horizontal information of each sliding zone A,B,C,D. To do this computing, the left bit of the sliding zone A and the right bit of the sliding zone D are not take into account because they are on the neighboring zone. The horizontal bitset is composed by 6 bits instead of 8. The left / middle or right bit is activated depending of which sliding zone is touch. There are the rules to activate each bit :

A_{right} OR $B_{left} \rightarrow$ LEFT
B_{right} OR $C_{left} \rightarrow$ MIDDLE
C_{right} OR $D_{left} \rightarrow$ RIGHT

```
000000 => 000 000001 => 001 000010 => 001 000011 => 001
000100 => 010 000101 => 011 000110 => 011 000111 => 011
001000 => 010 001001 => 011 001010 => 011 001011 => 011
001100 => 010 001101 => 011 001110 => 011 001111 => 011
010000 => 100 010001 => 101 010010 => 101 010011 => 101
010100 => 110 010101 => 111 010110 => 111 010111 => 111
011000 => 110 011001 => 111 011010 => 111 011011 => 111
011100 => 110 011101 => 111 011110 => 111 011111 => 111
100000 => 100 100001 => 101 100010 => 101 100011 => 101
100100 => 110 100101 => 111 100110 => 111 100111 => 111
101000 => 110 101001 => 111 101010 => 111 101011 => 111
101100 => 110 101101 => 111 101110 => 111 101111 => 111
110000 => 100 110001 => 101 110010 => 101 110011 => 101
110100 => 110 110101 => 111 110110 => 111 110111 => 111
111000 => 110 111001 => 111 111010 => 111 111011 => 111
111100 => 110 111101 => 111 111110 => 111 111111 => 111
```

Figure 30: Zone horizontal memory A6D3 : programmable memory 4

Appendix F Programmable memory 5 : zone horizontal with vertical information memory A8D3

The goal of the programmable memory 5 is to compute the **vertical sliding zones bitset** into a three bits bitset called the **horizontal tracker status per zone (HTSZ)**. This memory is used only if memory 4 cannot be used. This memory is an A8D3 memory.

How to build this programmation table ?

There are 8 possibilities for the D3 response **for a zone** :

000 : Horizontal Void (HVOID)

001 : Horizontal Right (HRIGHT)

010 : Horizontal Middle (HMIDDLE)

011 : Horizontal Right-Middle (HRIGHT-MIDDLE)

100 : Horizontal Left (HLEFT)

101 : Horizontal Left-Right (HLEFT-RIGHT)

110 : Horizontal Left-Middle (HLEFT-MIDDLE)

111 : Horizontal Wide (HWIDE)

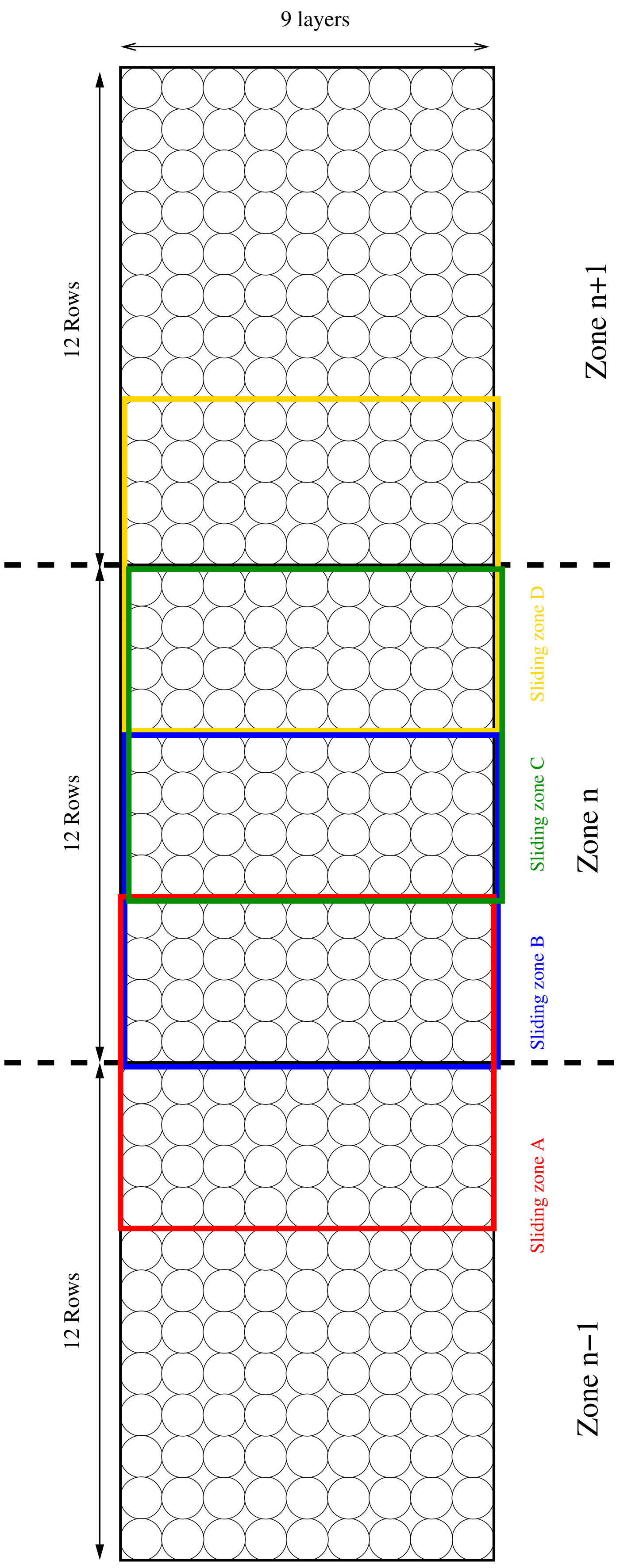
This response is based on the vertical information of each sliding zone A,B,C,D. This memory is coded in reversal logic because it was easier to interpret. The D3 response is 111 and the goal is to inhibit the left / middle or right bit. The rules for the computing are the following :

$\overline{A} \rightarrow \overline{LEFT}$: left bit is set to 0
$\overline{D} \rightarrow \overline{RIGHT}$: right bit is set to 0
$\overline{B\&C} \rightarrow \overline{MIDDLE}$: middle bit is set to 0

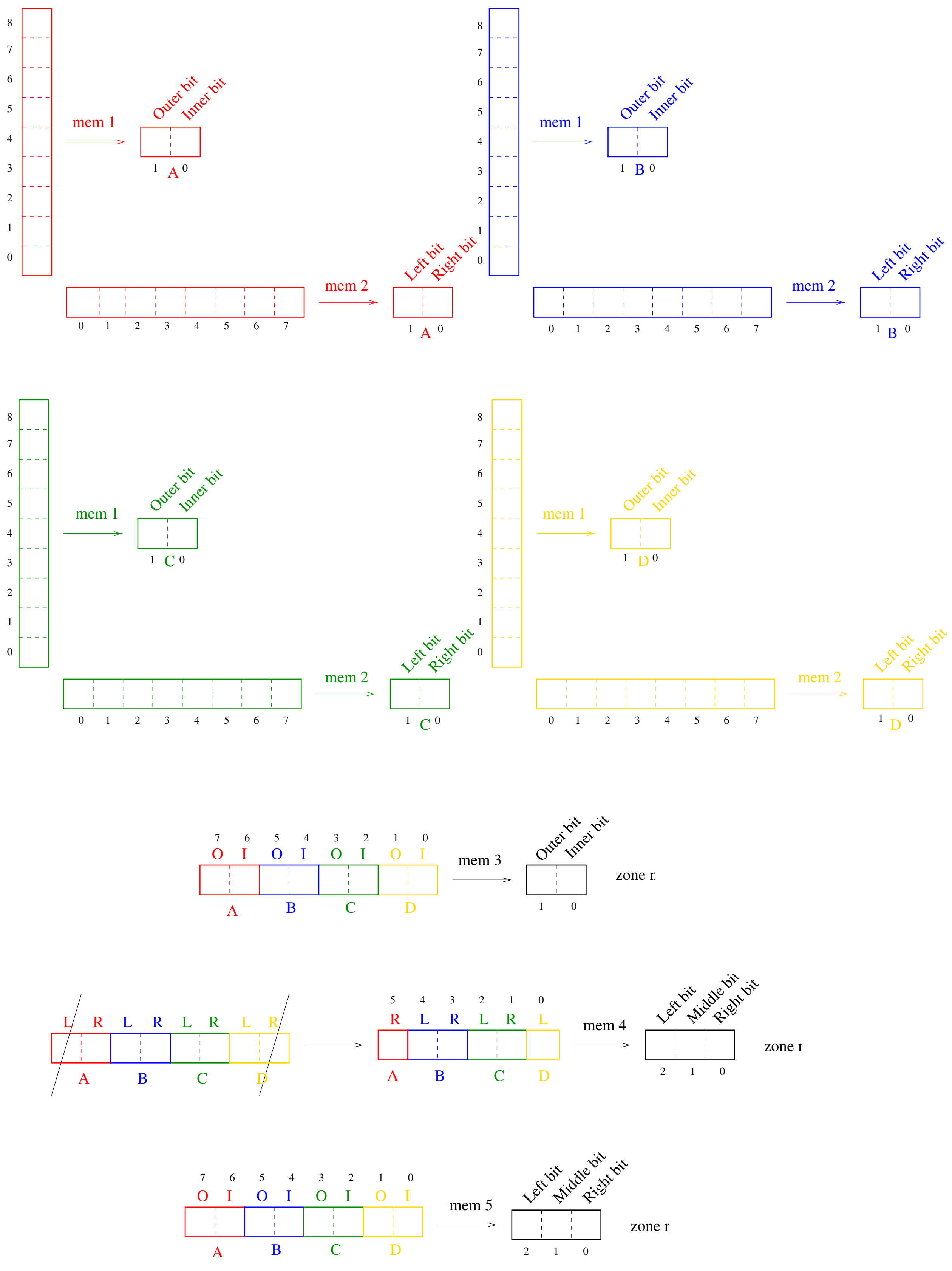
```
00000000 => 000 00000001 => 001 00000010 => 001 00000011 => 001 00000100 => 010 00000101 => 011
00000110 => 011 00000111 => 011 00001000 => 010 00001001 => 011 00001010 => 011 00001011 => 011
00001100 => 010 00001101 => 011 00001110 => 011 00001111 => 011 00010000 => 010 00010001 => 011
00010010 => 011 00010011 => 011 00010100 => 010 00010101 => 011 00010110 => 011 00010111 => 011
00011000 => 010 00011001 => 011 00011010 => 011 00011011 => 011 00011100 => 010 00011101 => 011
00011110 => 011 00011111 => 011 00100000 => 010 00100001 => 011 00100010 => 011 00100011 => 011
00100100 => 010 00100101 => 011 00100110 => 011 00100111 => 011 00101000 => 010 00101001 => 011
00101010 => 011 00101011 => 011 00101100 => 010 00101101 => 011 00101110 => 011 00101111 => 011
00110000 => 010 00110001 => 011 00110010 => 011 00110011 => 011 00110100 => 010 00110101 => 011
00110110 => 011 00110111 => 011 00111000 => 010 00111001 => 011 00111010 => 011 00111011 => 011
00111100 => 010 00111101 => 011 00111110 => 011 00111111 => 011 01000000 => 100 01000001 => 101
01000010 => 101 01000011 => 101 01000100 => 110 01000101 => 111 01000110 => 111 01000111 => 111
01001000 => 110 01001001 => 111 01001010 => 111 01001011 => 111 01001100 => 110 01001101 => 111
01001110 => 111 01001111 => 111 01010000 => 110 01010001 => 111 01010010 => 111 01010011 => 111
01010100 => 110 01010101 => 111 01010110 => 111 01010111 => 111 01011000 => 110 01011001 => 111
01011010 => 111 01011011 => 111 01011100 => 110 01011101 => 111 01011110 => 111 01011111 => 111
01100000 => 110 01100001 => 111 01100010 => 111 01100011 => 111 01100100 => 110 01100101 => 111
01100110 => 111 01100111 => 111 01101000 => 110 01101001 => 111 01101010 => 111 01101011 => 111
01101100 => 110 01101101 => 111 01101110 => 111 01101111 => 111 01110000 => 110 01110001 => 111
01110010 => 111 01110011 => 111 01110100 => 110 01110101 => 111 01110110 => 111 01110111 => 111
01111000 => 110 01111001 => 111 01111010 => 111 01111011 => 111 01111100 => 110 01111101 => 111
01111110 => 111 01111111 => 111 10000000 => 100 10000001 => 101 10000010 => 101 10000011 => 101
10000100 => 110 10000101 => 111 10000110 => 111 10000111 => 111 10001000 => 110 10001001 => 111
10001010 => 111 10001011 => 111 10001100 => 110 10001101 => 111 10001110 => 111 10001111 => 111
10010000 => 110 10010001 => 111 10010010 => 111 10010011 => 111 10010100 => 110 10010101 => 111
10010110 => 111 10010111 => 111 10011000 => 110 10011001 => 111 10011010 => 111 10011011 => 111
10011100 => 110 10011101 => 111 10011110 => 111 10011111 => 111 10100000 => 110 10100001 => 111
10100010 => 111 10100011 => 111 10100100 => 110 10100101 => 111 10100110 => 111 10100111 => 111
10101000 => 110 10101001 => 111 10101010 => 111 10101011 => 111 10101100 => 110 10101101 => 111
10101110 => 111 10101111 => 111 10110000 => 110 10110001 => 111 10110010 => 111 10110011 => 111
10110100 => 110 10110101 => 111 10110110 => 111 10110111 => 111 10111000 => 110 10111001 => 111
10111010 => 111 10111011 => 111 10111100 => 110 10111101 => 111 10111110 => 111 10111111 => 111
11000000 => 100 11000001 => 101 11000010 => 101 11000011 => 101 11000100 => 110 11000101 => 111
11000110 => 111 11000111 => 111 11001000 => 110 11001001 => 111 11001010 => 111 11001011 => 111
11001100 => 110 11001101 => 111 11001110 => 111 11001111 => 111 11010000 => 110 11010001 => 111
11010010 => 111 11010011 => 111 11010100 => 110 11010101 => 111 11010110 => 111 11010111 => 111
11011000 => 110 11011001 => 111 11011010 => 111 11011011 => 111 11011100 => 110 11011101 => 111
11011110 => 111 11011111 => 111 11100000 => 110 11100001 => 111 11100010 => 111 11100011 => 111
11100100 => 110 11100101 => 111 11100110 => 111 11100111 => 111 11101000 => 110 11101001 => 111
11101010 => 111 11101011 => 111 11101100 => 110 11101101 => 111 11101110 => 111 11101111 => 111
11110000 => 110 11110001 => 111 11110010 => 111 11110011 => 111 11110100 => 110 11110101 => 111
11110110 => 111 11110111 => 111 11111000 => 110 11111001 => 111 11111010 => 111 11111011 => 111
11111100 => 110 11111101 => 111 11111110 => 111 11111111 => 111
```

Figure 31: Zone horizontal memory A8D3 with sliding zone vertical information : programmable memory 5

Appendix G Abacus : Empty tracker matrix



Appendix H Abacus : Empty tracker bitsets


mem 1
Outer bit
Inner bit
A
B
C
D
mem 2
Left bit
Right bit
mem 3
zone r
O I O I O I O I
L R L R L R L R
R L R L R L
mem 4
Left bit
Middle bit
Right bit
mem 5