VDK9 R ½

Design document – Version 1.00 – June 12th 2001

Author : M. Zalfany Urfianto (zalfany@opencores.org)

Table Of Contents

Design Specification

1. Code Rate = ½, Constraint Length (K) = 9
   Some consequences of using a constraint length of 9 are:
   - The amount of branch metric is $2^{K} = 512$ branches.
   - The amount of state metric is $2^{K-1} = 256$ states.
   - The depth of trace back process is at least = 5 * (K-1) = 40. The version 1.0 of VDK9R1/2 does the trace back with a depth of 63 stages.
2. Polynomial generator: 753_8 and 561_8.
3. Hard Decision.
   Hard decision distance calculation is used on Version 1.0.
   I don't get sufficient information on how a soft decision Viterbi decoding is performed. Does it have something to do with the modulator? Also with the demodulator? Enlighten me please.
   But as far as I know, to switch to the soft decision from a hard decision Viterbi decoder, the Branch Metric generation process has to be modified slightly. Of course because the output of BMG block is different from the previous one, the ACS block must also be modified to accommodate the changes.

Design Considerations

1. Output Data Rate

Convolutional codes using constraint length = 9 are mainly used on CDMA applications. Therefore, the output data rates should be at least 9.6 kbps.

2. Area efficient VLSI implementation.
   The not-very-high speed requirement would enable us to try to minimize the area used for the design. On Version 1.0, the minimization is achieved by using only 4 ACS processors to do the Add-Compare-Select operations.
3. Memory requirement.
   The largest part of a Viterbi decoder is memory. Let's take an example using constraint length value of 9. If every metric is 8 bits wide, the memory size needed to save all the metrics is: 256 * 8 = 2.048 bits. Double it (because we have to save the current and the next metrics), we need 4.096 bits of memory space. For the survivors, with a trace back depth of 45 (which is the minimum depth we have to use), there survivor memory size is: 256 (states) * 40 (depth) = 10.240 bits.
   Because of its size, it would be better if we put the memory "outside" the design. The Viterbi decoder will provide an interface to the memories, and the physical implementations of the memories are not the main considerations on our design.

VDK9R1/2 Block Diagram

The block diagram of VDK9R1/2 Version 1.0 is shown on figure 1.

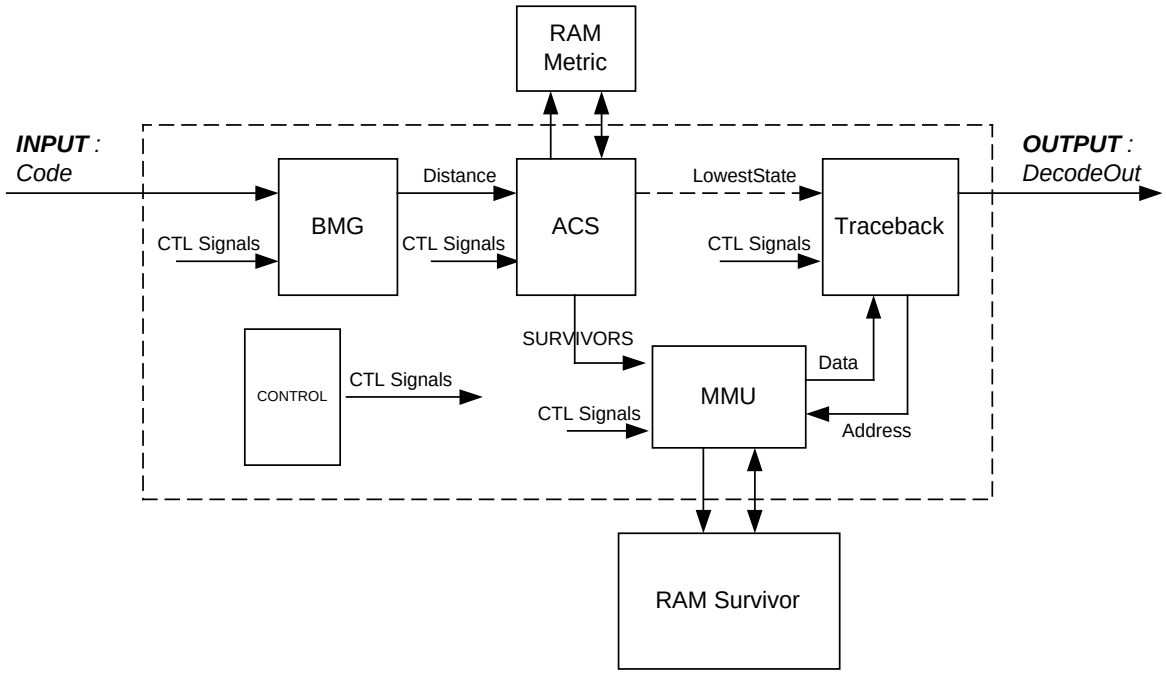

Figure 1. VDK9R1/2 Block Diagram

There are 5 main components of the Viterbi decoder: BMG (Branch Metric Generator) Unit, ACS (Add Compare Select) Unit, Traceback Unit, MMU (Memory Management Unit) and its Survivor Memory, and Control Unit. Each component's function will be briefly described below.

Control Unit

Control Unit is used to provide control signals for the other components. Those signals are listed on Table I.

Table I. Control Unit Signals

Signal Name	Description
`Clock1` and `Clock2`	`Clock1` and Clock2 are signals with frequency equals to ¼ of `Clock` frequency. ▪ The phase of two signals are differ by 90^0. `Clock2` is 90^0 ahead of `Clock1`. ▪ `Clock1` is used by Control Unit to update ACSSegment (which is consequently update the output of BMG Unit and ACS Unit). ▪ `Clock2` is used to indicate when to put address to Address Bus.
`ACSPage [5:0]`	Indicate the current active page memory. Active here means the position of memory page where the survivor data will be written.
`ACSSegment [5:0]`	Indicate the processed state segment. Because of there were only 4 ACS available, and the number of state is 256, we will need 64 iterations.
`Init`	Indicate the beginning of process for 1 `Code` signal (decoder input).
Hold	Indicate the end of process for 1 `Code` signal (decoder input).
`CompareStart`	As shown on the trellis diagram, for first (K – 1) process, on each node, there is only one input coming in. The survivor, then, must be those branch. The `CompareStart` will signal ACS not to do a Compare operation during the first (K-1) process.
`TB_EN`	The trace back process will begin only after there were at least DEPTH sets of survivor in the survivor memory. On Version 1.0, because of the value of DEPTH is 63 then the trace back will begin when `ACSPage` value is 63 (3F hex).

BMG (Branch Metric Generator) Unit

BMG Unit block diagram is shown on Figure 2.

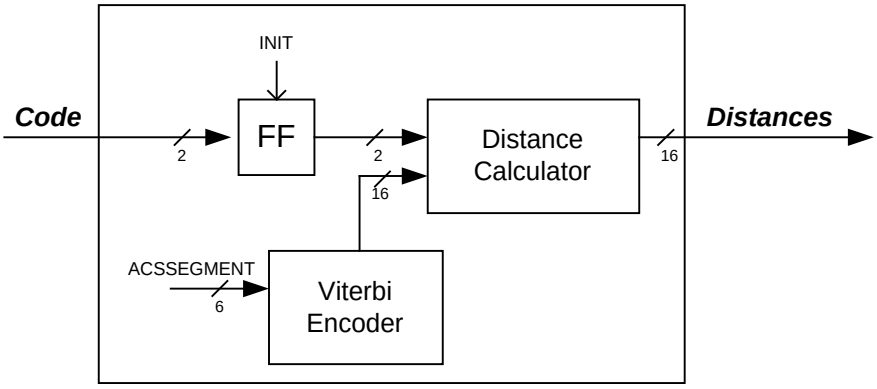

Figure 2. BMG Unit Block Diagram

The BMG receives `Code` signals, calculating its distance with all possibility of branch metric, giving the output of `Distances` signal.

All possible values of branch metric is generated by the Viterbi Encoder block. The values generated is depend on the value of `ACSSegment`.

The DistanceCalculator block computes the hard-distance of `Code` with the branch metric from ViterbiEncoder block.

ACS (Add Compare Select) Unit

The ACS Unit block diagram is shown on Figure 3.

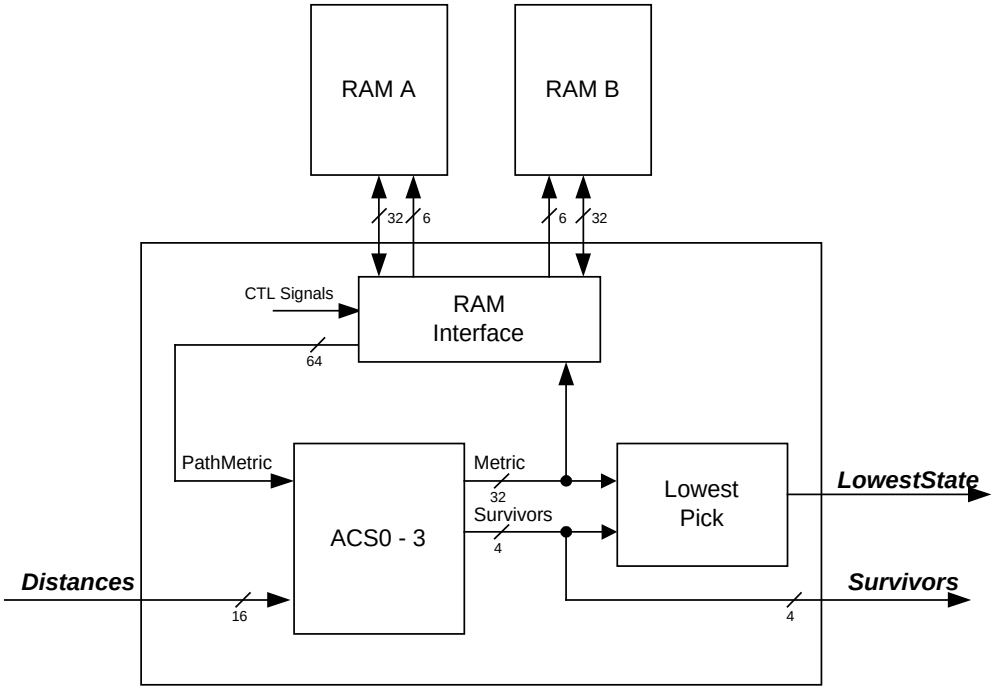

Figure 3. ACS Unit Block Diagram

The ACS Unit consists of ACS0-3 block, Lowest Pick block, and RAM Interface.

ACS0 - ACS3

ACS0 - ACS3 block consists of 4 ACS processors. The ACS processor's architecture is called *Modified Comparison Rule* as described by Shung, C.B. on his paper : "VLSI Architecture for Metric Normalization in the Viterbi Decoders", Proceeding ICC, Vol.4; 1990.

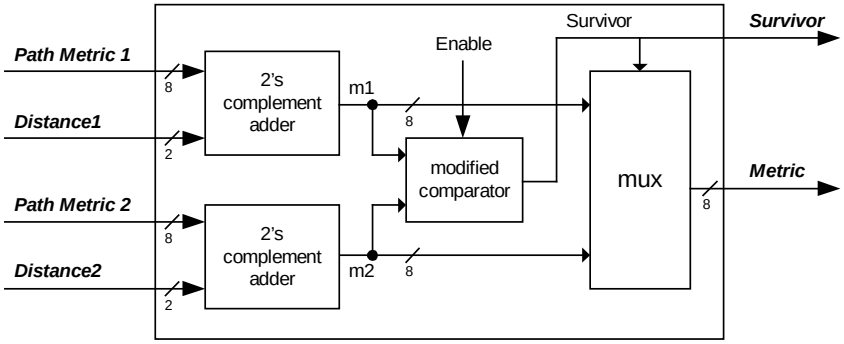

Figure 4. An ACS Processor

The Modified Comparison Rule processor uses two 2's-complement adders and two XOR gate to work.

Note that there is an Enable signal on the comparator. This signal is come from the Control Unit.

Because of there were only 4 ACS processors available, the input to the ACS has to be arranged so that all state could be processed efficiently. On VDK9R1/2 Version 1.0, the states to be processed are arranged sequentially. Meaning the first cycle will update state 0000 0000, 0000 0001, 0000 0010, and 0000 0011. The next cycle will update state 0000 0100, 0000 0101, 0000 0110, and 0000 0111, and so on, until the 64^{th} cycle will update state 1111 1100, 1111 1101, 1111 1110, and 1111 1111.

Lowest Pick

The Lowest Pick block is used to determine which of all 256 metrics is the smallest one and what state has those smallest metric. The value of state acquired will be used by the trace back to indicate in which state should the trace back process begin.

The Lowest Pick block consists of two parts. The first part is used to find the smallest metric among every 4 outputs of ACS processor. The second part will compare the output of the first part to the previous smallest metric.

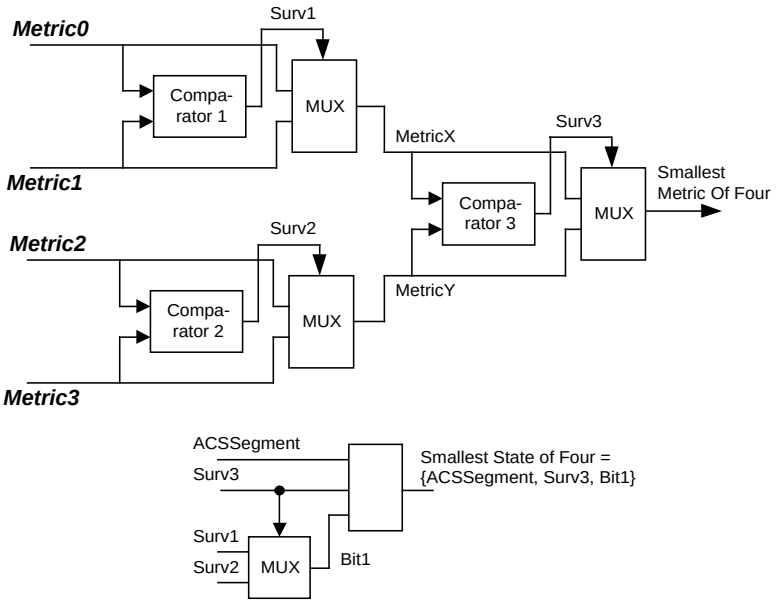

Figure 5. The first part of Lowest Pick Block

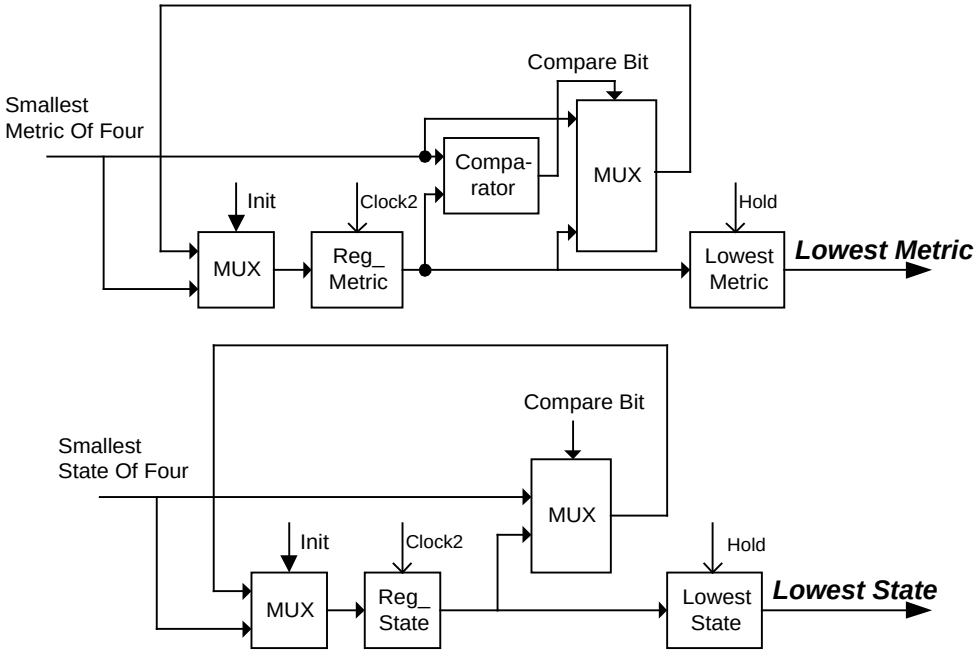

Figure 6. The second part of Lowest Pick Block

Metric Memory

The metric values are saved on Metric Memory. Two blocks of RAM needed as we have to know the current metric values and save the next metric values we've just calculated.

The RAM interface for ACS Block currently have not been coded tidily. There were some considerations such as the need of large data bus (32 bits), the relatively small size (2 Kbytes). Hmm.. there were lot of things to be fixed here.

On the current source code (Version 1.0), I use a block of Registers for the MetricMemory. Each register has its own index (just like index on a matrix), therefore the addressing scheme using signal `MMReadAddress` and `MMWriteAddress` at module RAMINTERFACE could be taken directly from the value of `ACSSegment.` And what makes it better, the metric values inside the memory could be updated directly as if we update a common register.

I did it this way because of that the data bus is large (32 bit for read process, and 16 bit for write process), I think it's better to implement the metric memory inside the decoder (using a register), rather than using an external RAM.

Anyway, should we use standard RAM operations the RAM Interface job is to do the following process :

- Select which block of RAM used to read the metric and which one is used to write the metrics we've just calculated. (On the source code, `MMBlockSelect` signal do this).
- Upon `RESET`, the values of metrics on both block of RAM have to be set to 0.
- The update process on the ACS processor happened on every rising edge of `Clock1`. Therefore, the read process from the RAM (to get metric values) has to be performed on the previous rising edge of `Clock2`. The write process to RAM (to save the calculated perform) could be performed on falling edge of `Clock2`.

The timing diagram could be shown as of Figure 7 below.

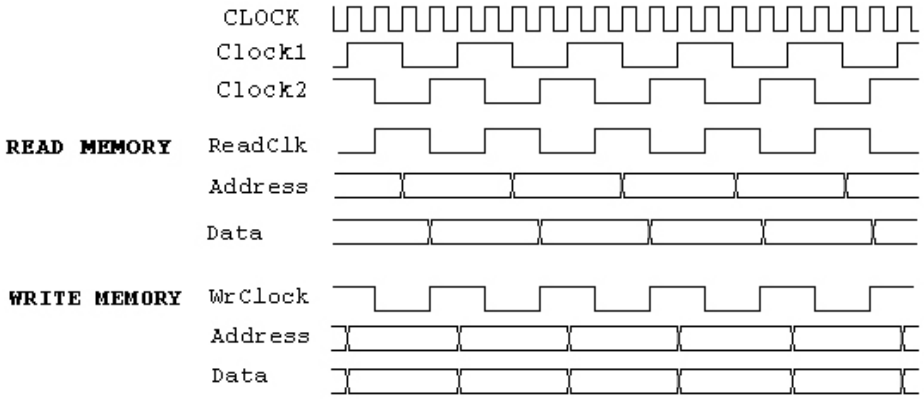

Figure 7. Metric Memory Timing Diagram

Note that the Read and Write operation of metric memory happened upon the falling edge of ReadClk and WrClock signal respectively.

Traceback Unit

The trace back algorithm implemented on VDK9R1/2 Version 1.0 could be described as follows :

1. On time *t*, select a node where the trace back process will start.
   In VDK9R1/2 the starting node is the state which has smallest metric value. This value is coming from the ACS Unit.
2. From the survivor memory, get survivor value of those node.
3. The previous state (on level *t –1*) is calculated by shifting left the node value by 1 bit, the LSB part then will be filled with the survivor value gathered from the memory.
4. Back to step 2 for the state on level *t-1*. Continue until node on level *t – DEPTH*.
5. The decoded values is the MSB of node on level *t – DEPTH*.

Let's take a simple example using a K=2 convolutional code case. Suppose that on level *t*, we take node 00 as the starting point (*step 1*). The survivor data for state 00 is 1 (*step 2*). Then we know that the surviving branch must come from state : 01 (*step 3*). Back to step 2 for state 01 on level t-1 (step 4). The survivor value of state 01 level t-1 is 1 (*step 2*), then we know that the surviving branch must come from state : 11 (*step 3*). And so on until level *t-DEPTH*.

The Traceback Unit block diagram is shown on figure 8.

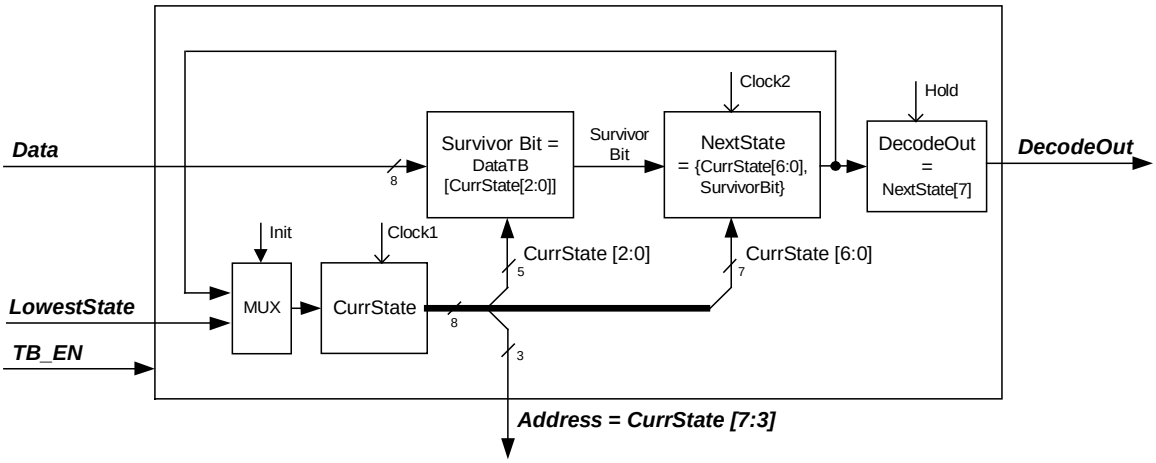

Figure 8. Traceback Unit block diagram

When there was an `Init` signal, the LowestState value will be used as CurrState value. Traceback Unit will then send Address to the MMU, requesting survivor data from the survivor memory. Because the width of survivor memory data bus is 8, and there were 256 states, there will 32 blocks of survivor data. The `Address` signal from the Traceback Unit, which is 5 bit width, will be used to select one block out of 32 possible blocks.

The `CurrState[2:0]` then will be used to select 1 `Survivor Bit` from the 8 bit `Data` taken from memory.

The `NextState` value is then determined as the concatenation of `CurrState[5:0]` and `Survivor Bit`.

When there was a Hold signal, the Traceback Unit will output a `DecodeOut` signal.

Memory Management Unit (MMU) and Survivor Memory

The survivor memory is used to save the survivors data calculated by the ACS Unit. The survivor data will be later used by the Traceback unit to find decoded data. The MMU block will control the operation of survivor memory.

RAM

Assume that we use a typical RAM which has input ports : `Address, Enable, Write Clock, Read Clock, and RWSelect`, and an inout port `Data`.

The RAM timing diagram could be simplified as of shown at Figure 9.

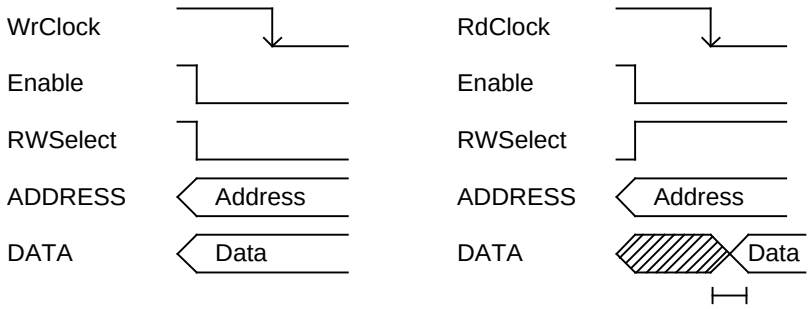

Figure 9. Survivor RAM Simplified Timing Diagram

To perform a write operation, we have to put the data and address at Data Bus and Address Bus, and then give the WrClock signal. To ensure the signal validity, there have to be enough time after we put data and address on the line before the WrClock is given.

To perform a read operation, put an address, give a RdClock signal, and after a slight delay, the data will be available on Data Bus.

The size of the RAM needed to perform a trace back operation with a DEPTH of 63 is (63+1) * 256 bit = 16.384 bit. One extra page is used to save the survivor of current `Code`. The rest are filled with complete sets of survivor from the previous operations. Using a Data Bus width of 8, there will be ($^{2}\log 16384 - {}^{2}\log 8$) = (14 – 3) = 11 bit wide Address Bus.

MMU

The block diagram of MMU is shown on Figure 10.

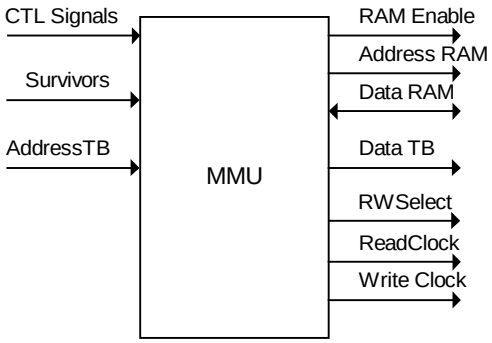

Figure 10. MMU Block Diagram

The write operation on the survivor memory will occur every 2 falling edge of `Clock1`. Remember that the number of ACS is only 4, while the Data Bus is 8 bit wide.

The read operation will occur on every rising edge of Clock1.

The survivor memory timing diagram is shown on Figure 11.

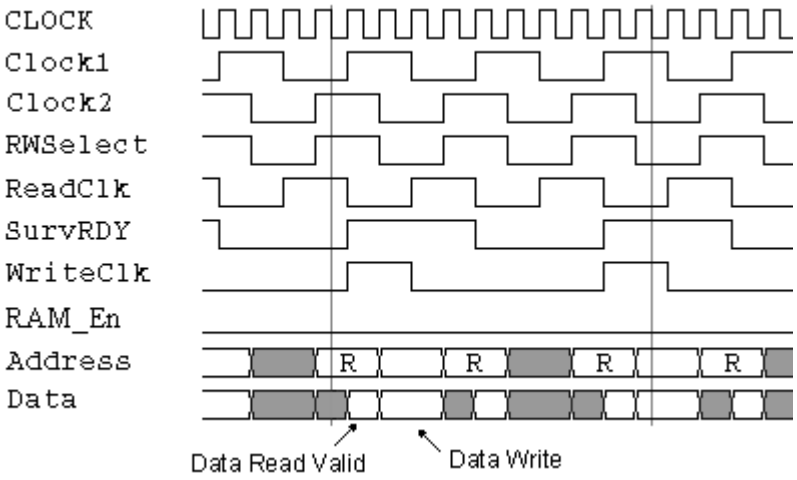

Figure 11. Survivor Memory Timing Diagram

The Survivor Memory Addressing Scheme

Write Operation

The address for write operation is assembled from the combination of the value of ACSPage [5:0] and the value of ACSSegment [5:1].

ACSSegment [5:1] \ ACSPage	0	1	2	3	.	.	.	15	.	.	.	.	.	.	.	59	60	61	62	63
1																				
2																				
3																		B		
.																				
.																				
1D																				
1E								A												
1F																				
Hex	0	1	2	.	.	.		0F	.	.	.	.	.	.	.	3B	3C	3D	3E	3F

Let's take an example, for position A, the address would be : $\{0F_{16}, 1E_{16}\} = \{00\_1111_2,1\_1110_2\} = 1FE_{16}$. For position B, the address is $\{3D_{16},3_{16}\} = \{11\_1101_2,0\_0011_2\} = 3A3_{16}$.

Read Operation

For read operation, the address is assembled from the `TBPage` signal from the MMU, and `AddressTB` signal coming from the Traceback Unit. → `Address = {TBPage, AddressTB}`

Upon an `Init` signal, the value of `TBPage` is loaded with the value of `ACSPage`–1. The resulting Address, the combination of `TBPage` and `AddressTB`, will then be used to read the survivor data. `DataRAM`, will then be sent into the Traceback Unit.

The Traceback Unit will provide the next AddressTB signal. By decreasing the TBPage one on every falling edge of Clock2, we will be able to read the survivor on level $t - 1$ node.