SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

STi3220
MOTION ESTIMATION PROCESSOR CODEC

By : Marc QUEROL

SUMMARY **Page**

This application note, based around three different examples, gives an overview of architectures providing the motion compensation function. More than a collection of schematic diagrams (that would not fit exactly to the user's application), it is more an explanation of what kind of architecture can fit to what kind of application, what precautions must be taken and what kind of components can be used or not.

All the architectures are based around the STi3220 chip developped by SGS-THOMSON microelectronics that provides the motion estimation function. The chip functionnalities will not be detailed here (refer to the STi3220 data sheet for more information): the application note concentrates on the way of providing the good informations to the chip and not on the way of writing or reading those informations into the chip. For that purpose the main part of the note is dedicated to the frame buffer choice and managing.

I - QUICK FEED-BACK ON COMPRESSION TECHNIQUES

Compression techniques trying to reduce the amount of information for the transmission or the storage of a moving picture, are mainly based around the property of pixel's correlation for natural images.

The first technique exploits the spatial correlation of pixels: a pixel in the image has a very high probability of having a value very close to its neighbours average value. The most popular tool using this property is the **Discrete Cosine Transform** which transforms a block of pixels from the original picture into a block of non correlated coefficients. Each coefficient represents a spatial frequency of the original block (the top left one being the average value). Due to high spatial correlation between pixels, for most of the natural images the highest frequency coefficients are of little significance and hence can be reduced or suppressed without altering the picture quality. This is the role of the quantisation block after the DCT.

The second technique exploits the temporal correlation between a pixel in an image and the pixel being at the same position in the previous image. Except when an image and the previous one are totally different (sequence changes), the areas changing from an image to the following are very rare for natural moving pictures. Thus, if **a prediction** of the current image is made, by copying the previous one, the difference between both will most often be close to zero : the non-zero differences represent the moving parts of the picture and are the only information needed to describe the new image from the previous one.

In order to increase the efficiency of the prediction, the **motion estimation** technique is used that associates to a sub-block of the image (also called reference block) a motion vector giving the relative position of the most similar block in the previous image: this block is used as a prediction block. A motion compensation is implemented.

Illustration of the motion compensation technique:

	3 3 2 2
2 2	3 3 2 2
2 2	4 4 3 3
	4 4 3 3
reference block	area of the previous picture centered around the reference block position

If the prediction is made without motion compensation, the predicted block is:

3 2
4 3

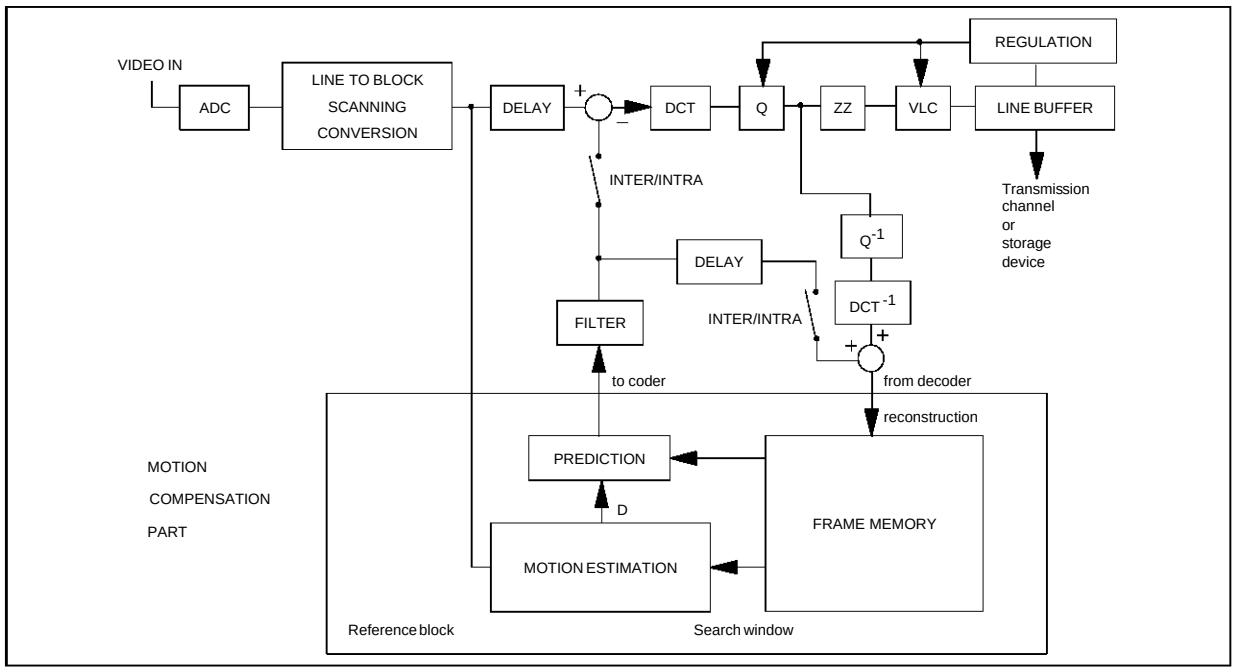
and the difference from the current block is:

1 0
2 1

Using motion estimation will deliver a motion vector equal to +1 in the horizontal direction (East) and -1 in the vertical direction (North). The prediction block is then the same than the reference one and the difference is zero: the only information needed to code the reference block is the motion vector.

The previously described compression techniques lead to the typical moving picture coder scheme shown in Figure 1. This application note only concentrates on the potential implementations of motion compensation.

Figure 1 : Typical Moving Picture Coder Scheme



II - MOTION COMPENSATION

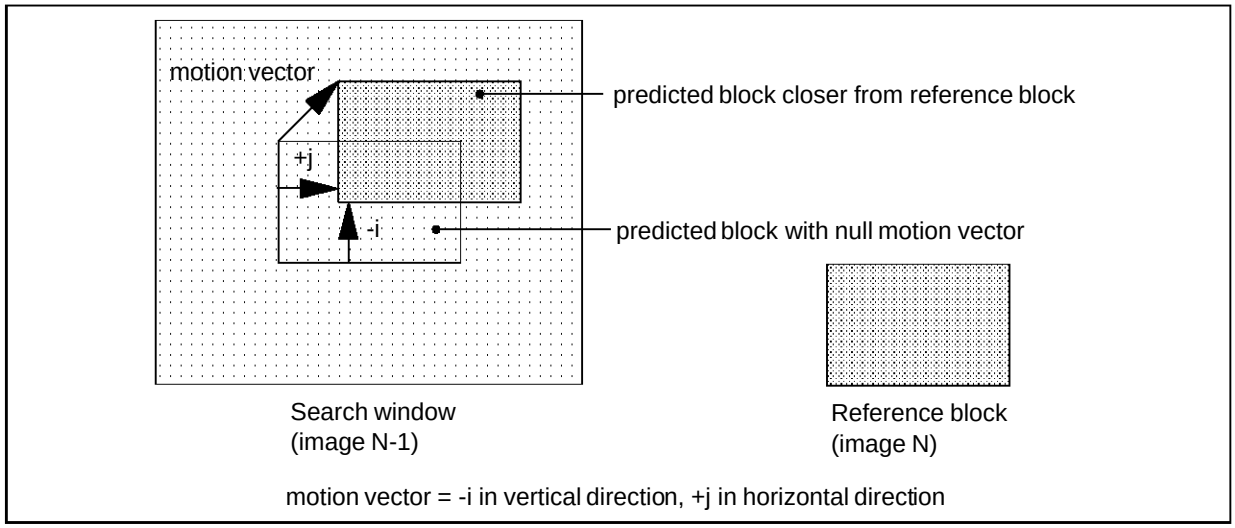
The motion compensation function is organised around three main functional blocks: a frame buffer for storage of the previous image, a motion estimator for research of the best motion vector and a predictor able to deliver to the coder the predicted block pointed to by the motion vector.

II.1 Motion estimation

One of the main devices, needing a lot of computation, is the motion estimator whose role is to find in a limited area of the previous image centered around the reference block position and called search window, the position of the block most similar to the reference block: this position is called the motion vector (see Figure 2).

In most cases, the picture is defined by the three components Y (luminance) U and V (chrominance), but the motion estimation is only made on the luminance component, the same resulting motion vector being applied to all components.

Figure 2



The comparison of the reference block with each possible block of the search window needs a lot of computation. An expression for the distance between the reference block and a predicted block is :

$$D(di,dj) = \sum_{m} \sum_{n} [Ref(m,n) - Pred(m+di,n+dj)]2$$

where :

Ref(m,n) = reference block pixel

Pred(m+di,n+dj) = predictor pixel with displacement di in vertical and dj in horizontal directions.

Computing a distance between two blocks 8x8 for instance needs 64 substractions, 64 multiplications and 64 accumulations. For a -8 to +7 possible displacement in both directions (horizontal and vertical), that is 256 possible predicted blocks, and implies 256 x 64 x 3 = 49152 operations to do during the 64 cycles of the reference block before being able to extract a motion vector.

The STi3220 motion estimation chip, developped by SGS-THOMSON, allows to find a motion vector in the range -8 to +7 in both directions, for reference block sizes of 8 x 4n (from 8x4 to 8x32) or 16 x 4n (from 16x4 to 16x16).

The formula used for computing the distance between the reference block and a candidate block of the search window is the Mean Absolute Error criterion defined by:

$$D(di,dj) = \sum_{m} \sum_{n} |Ref(m,n) - Pred(m+di,n+dj)|.$$

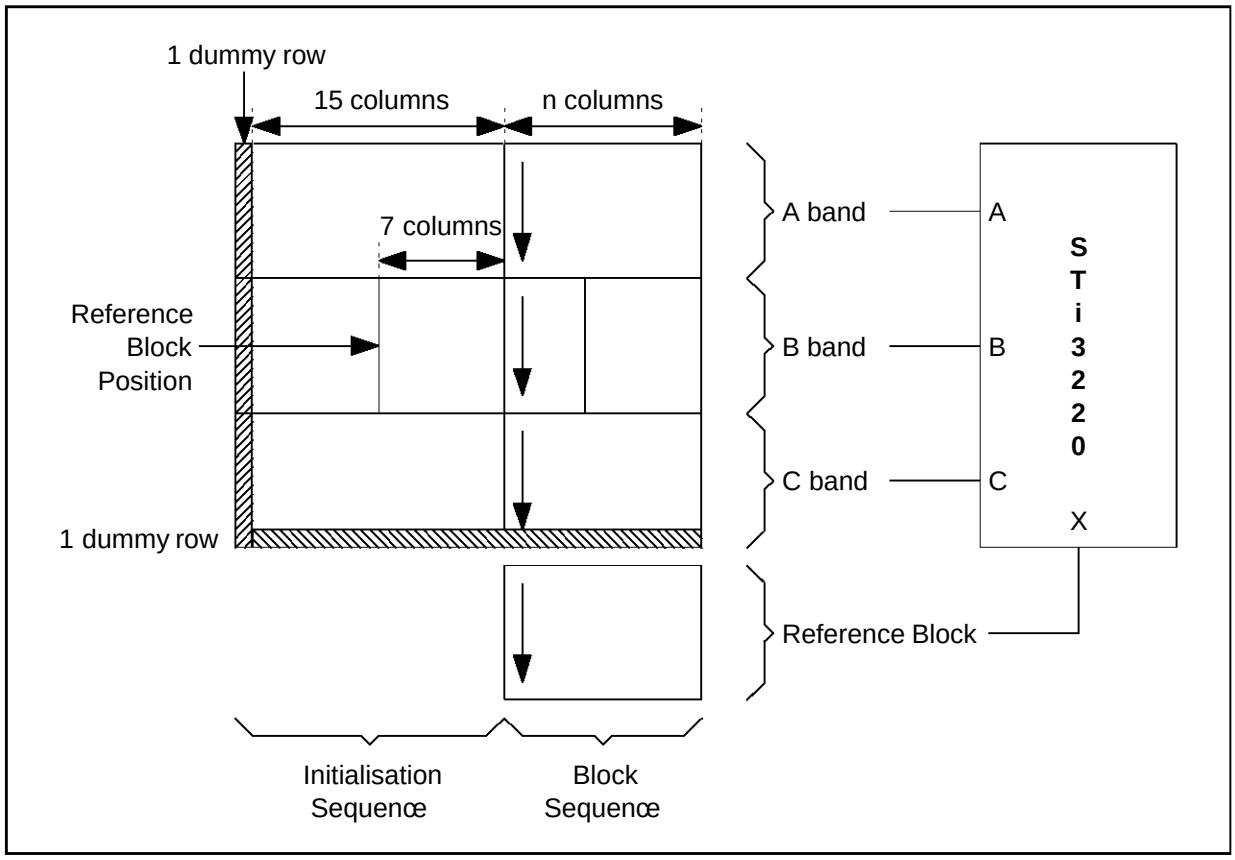
This distance is also called distortion.

The motion vector is the coordinates of the minimum distortion.

In order to compute the distortions, the STi3220 chip must be loaded with the search window and the reference block. For that purpose the chip has 4 input buses : one input for the reference block (X bus), one for the search window band above the reference block band (A bus), one for the search window band corresponding to the reference block one (B bus) and one input for the lower search window band (C bus).

Loading the chip is made column by column from top to bottom and left to right, a pixel being input on each of the four buses at each clock cycle.

Figure 3 : Search Window Input



Loading the chip is done in two phases (see figure 2.2):

- Initialisation sequence : Input of a dummy column of the search window used to initialise the internal pipeline architecture of the chip, followed by the 15 left-most columns of the search window. During that phase, the reference block input (X) is insignificant. If the reference block is 8-pixel high the initialisation phase costs 8x16=128 cycles. If the height is 16 pixels, the initialisation phase is 256 cycles long.
- Block sequence : During this phase the reference block and the end of the search window are input simultaneously into the chip. The last column of the reference block will exactly correspond to the last column of the search window. The chip will deliver the motion vector 38 (8x4n blocks) or 46 (16x4n blocks) cycles after the last pixel of the reference block, on a specific 8-bit bus IOB.

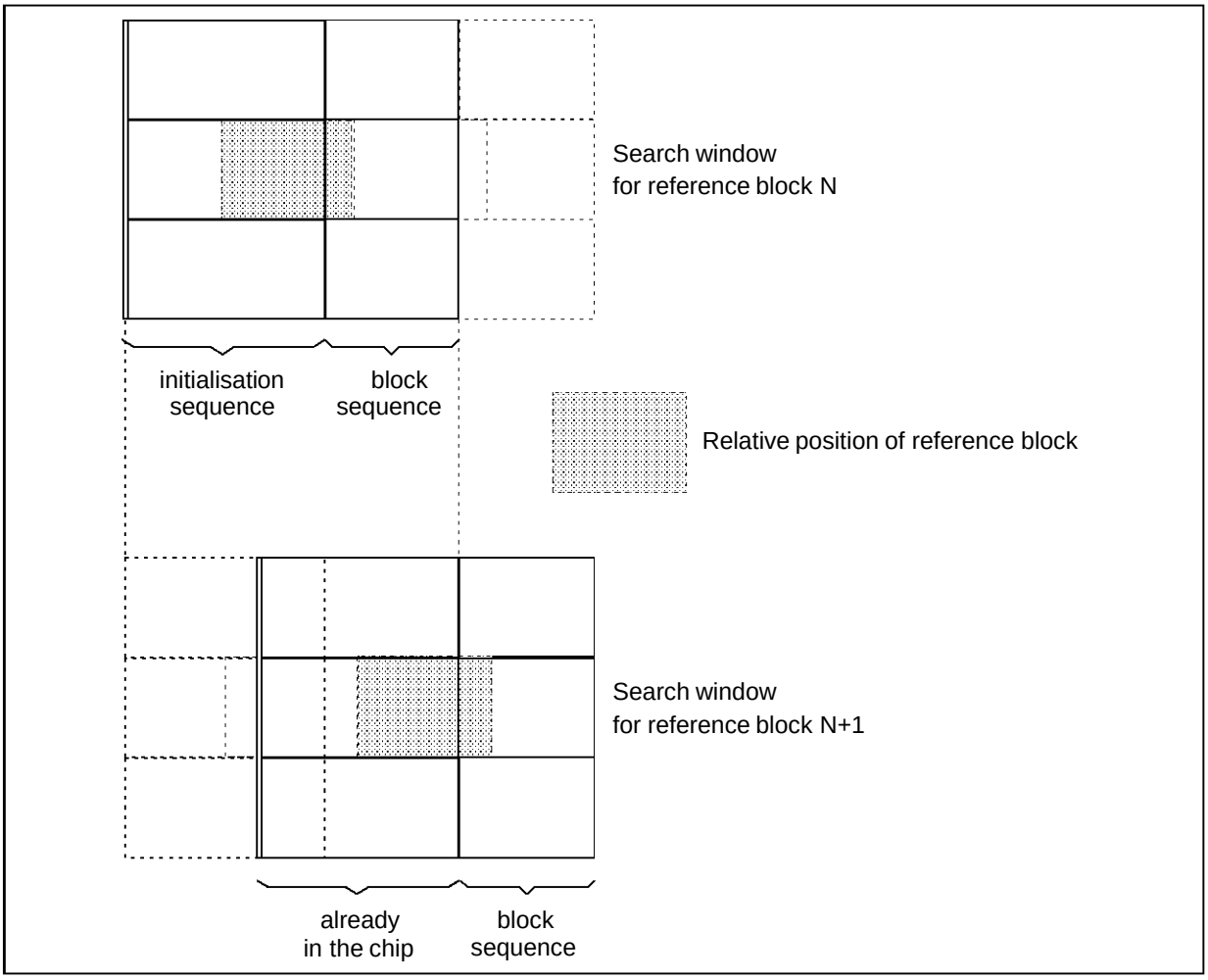
If the picture is computed in a band by band fashion, then the chip always contains the leftmost part of the search window that will be used for the next reference block and that is a subset of the current reference block's search window. That means that after the very first initialisation phase, a pipeline mode can be implemented: it is only mandatory to input the right part of the search window with the reference block, the left-most part being already in the chip (see Figure 4).

Chapters III and IV show examples of pipeline mode usage with two block sizes (8x8 and 16x16).

For motion vectors outside of the range -8 to +7, the search window must be cut into sub-windows supported by the chip (i.e. -8 to +7 max in both directions around the reference block size). The computations must be done:

- by several chips in parallel each one dedicated to a particular sub search window.
- by only one chip computing the motion vector on the different sub windows in several passes.

Figure 4 : Pipeline Mode between Two Consecutive Blocks



Chapter V shows an implementation of -16 to +15 research using one chip.

II.2 Prediction

- After computation of the motion vector for the Y component, it is necessary to extract, for each of the three components, a predicted block that will be subtracted from the reference block for coding. There are two main ways of delivering the predicted blocks :
- the first one consists in accessing directly into the reconstructed frame buffer (see Figure 5). This implies that we have enough bandwith available on the frame buffer ports.
- the second one, is used if the frame buffer bandwith is not sufficient and consists in storing in an additional RAM the search window as it is sent to the STi3220 and to access that RAM for delivery of the predicted block (see Figure 6). In order to be able to deliver the U and V predicted blocks, their corresponding search windows should also be sent to the additional RAM (while the STi3220 chip is disabled: EN input = 1). This implies that the STi3220 needs to be run at a higher speed.
- An alternative could be to split the frame buffer into two fields : one for Y and another for U and V. Y frame buffer is read out for search window delivery (prediction on additional RAM) while U/V frame buffer is accessed for prediction output (see Figure 7).

II.3 Frame buffer

The frame buffer is shared by several resources:

a) write of the reconstructed picture after decoding (necessary on Y, U and V components). As the decoder works on blocks of the image (for DCT) the write will be made in a block by block fashion. If the blocks are delivered by the decoder in a column order they can be directly used in the way they have been stored for the search window delivery.

b) read of the search window for motion estimation (only necessary on Y component). Reading the search window can be made in two ways :

- accessing 3 consecutive times to the frame buffer during one input cycle in order to deliver a pixel for the upper, middle and lower band of search window (see Figure 8). If F is the frequency of the input samples delivered by the line to block scanning, than the frame memory must be read at 3xF for search window access.
- accessing only one time to the frame buffer (one block after each other) and using two delay lines in order to provide the STi3220 with the three necessary search window bands (see Figure 9). The lower band of the search window is directly the output of the frame buffer, the middle band is the output of the first delay line and the upper band is the output of the second delay line. The delay line length is one line of block.

Figure 5 : Prediction access into the frame memory

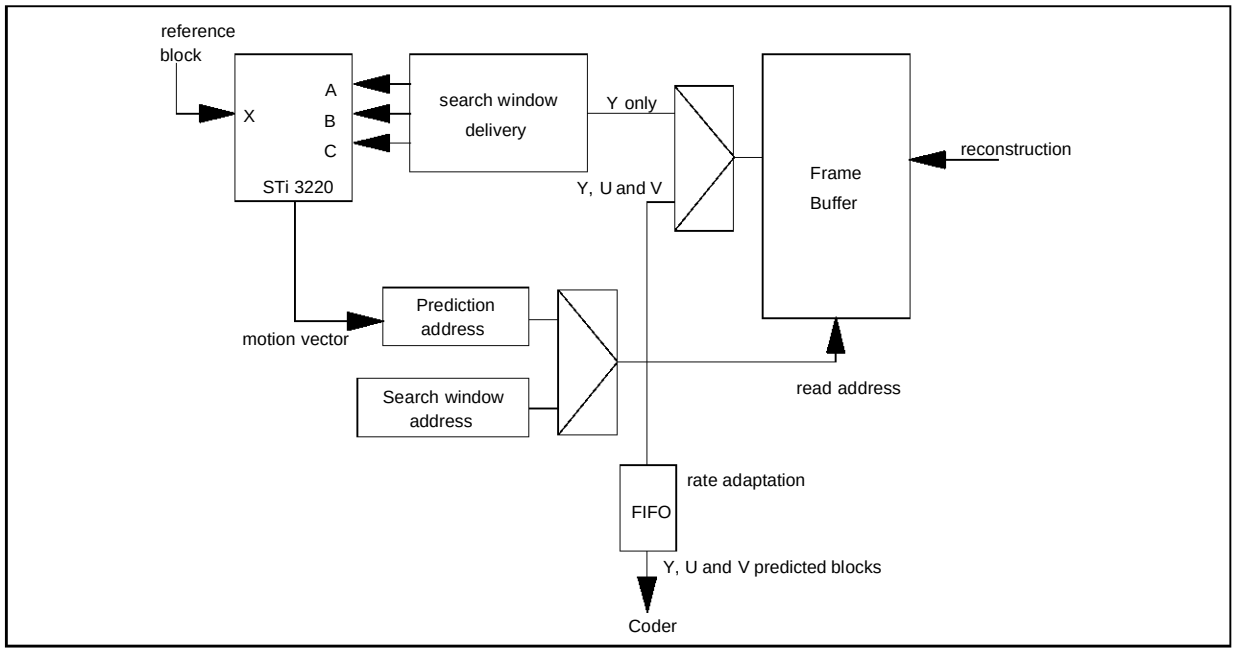


Figure 6 : Predictor Access into Additionnal Ram

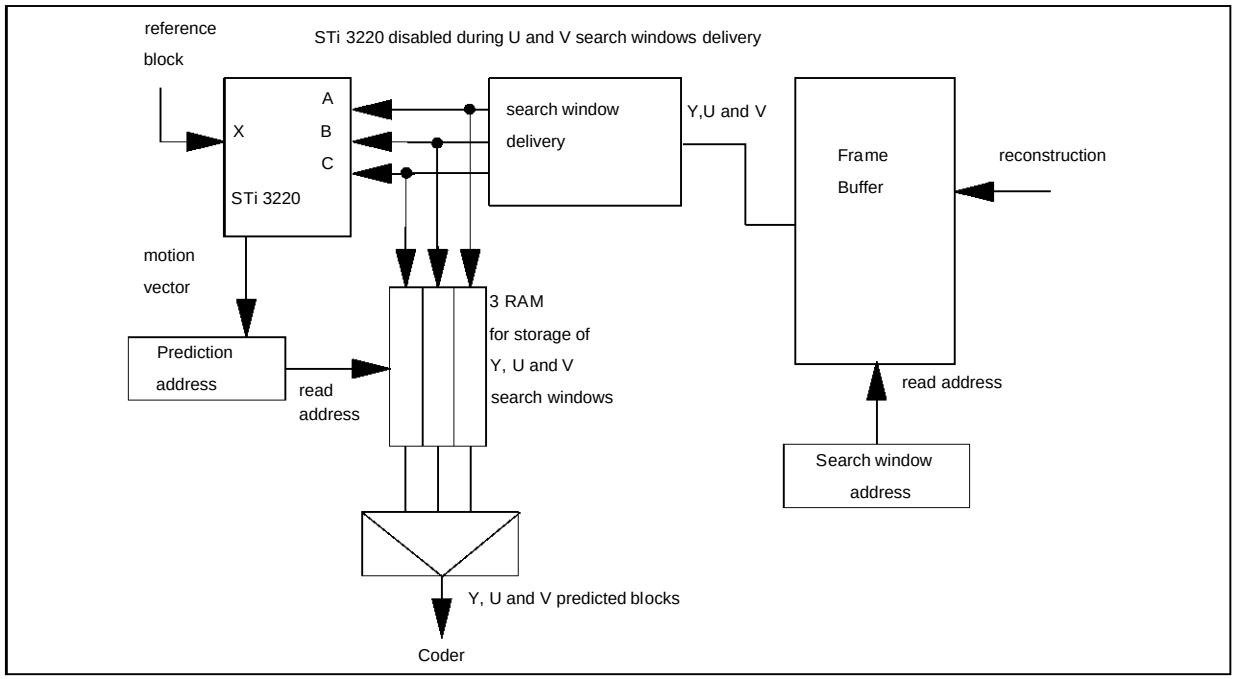
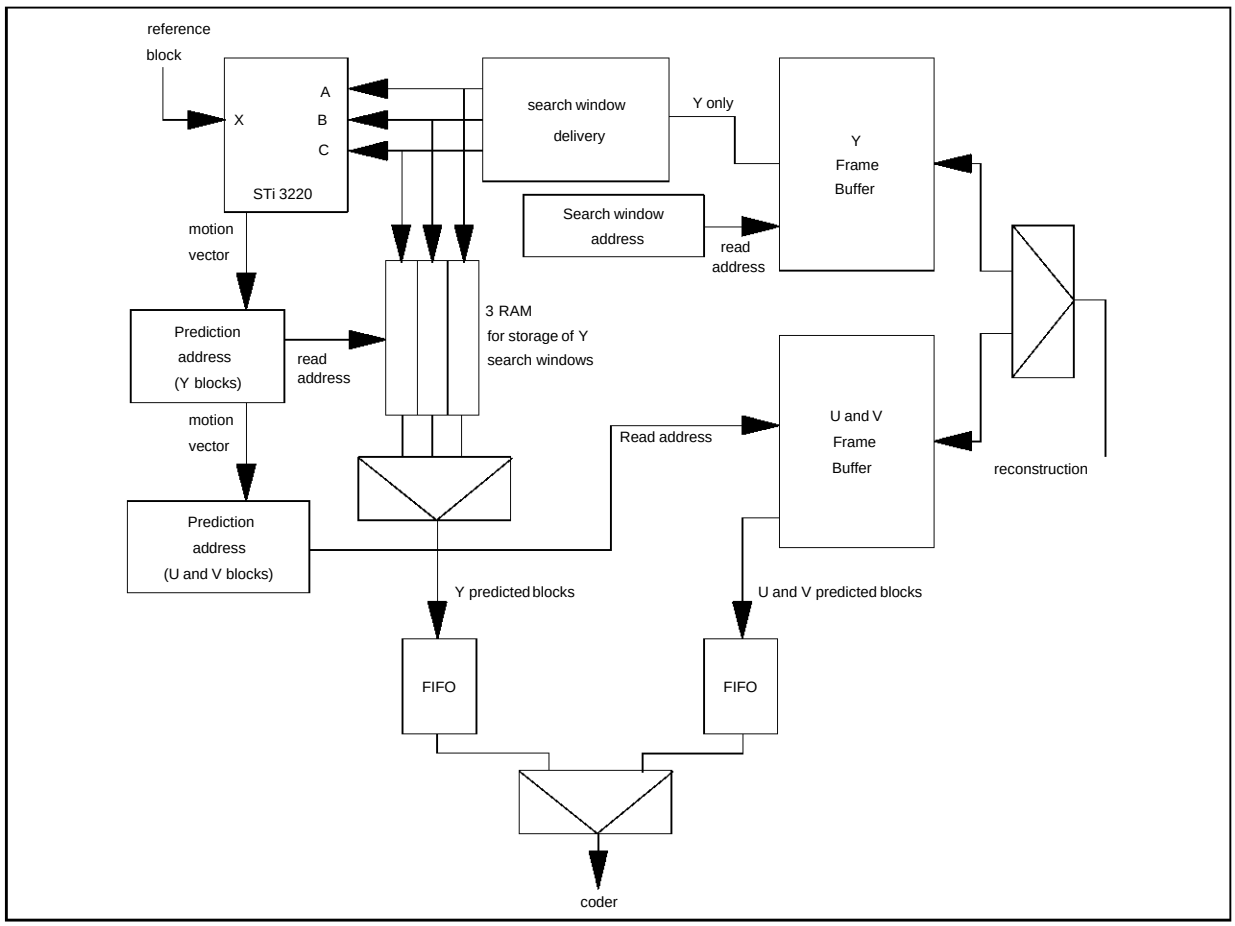


Figure 7 : Predictor Access into AdditionnaL RAM (Y) and Frame Buffer (U and V)



c) If the bandwith is sufficient, the frame buffer may be also accessed for delivery of the predicted Y, U and V blocks. If the bandwith is not high enough, the prediction cannot be made directly into the frame buffer and will be realised in an additional RAM (as explained in paragraph II.2).

The three previous parameters (frame reconstruction, search window and prediction) are the principle ones that will be taken into account in this note, but depending on the application (frame size, Y U V format ...) they can share the frame buffer with other resources. For instance an area of the memory can be reserved for implementation of the line to block conversion. Or the user may want to visualise the content of the frame buffer: in that case at least two frame stores are necessary, one for reconstruction, the other for visualisation. No doubt that other resources of the coder would require memory space and that the frame memory (that is not only used for frame) could provide free areas to use. The main problem will rapidly be the limitation on the memory access times that forces to duplicate the number of memory chips.

Figure 8 : 3 Accesses in Frame Buffer for Search Window

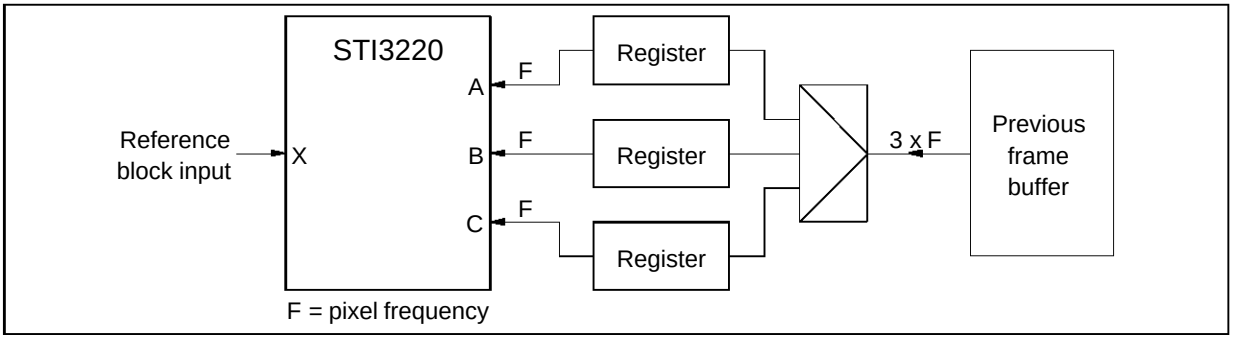
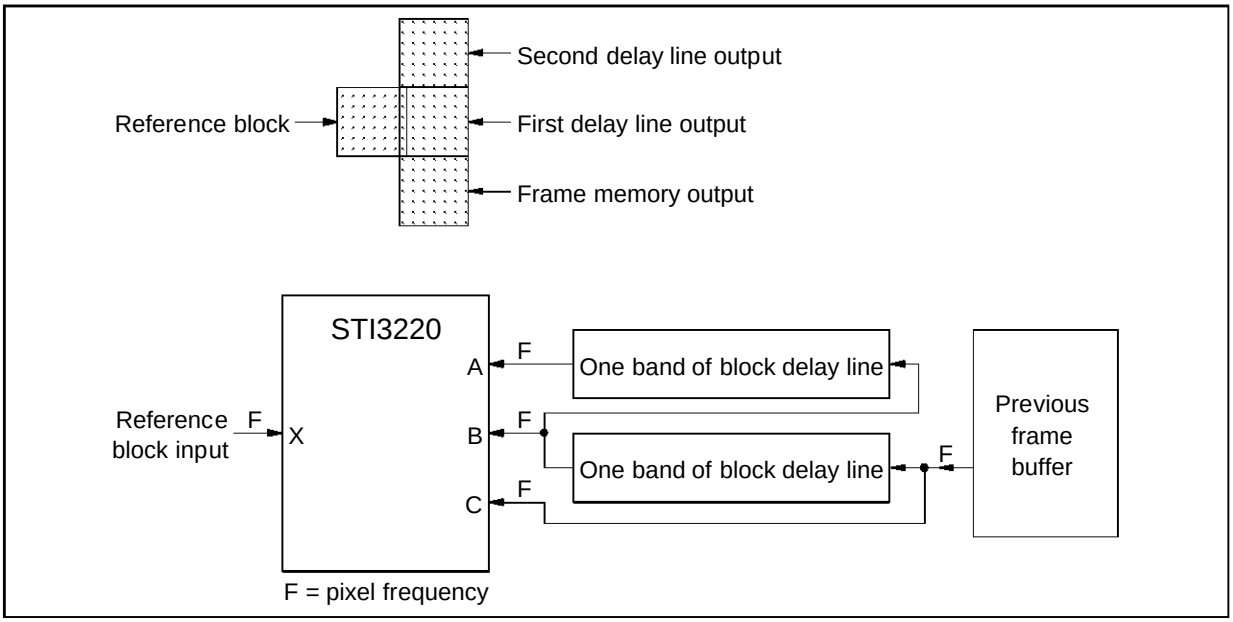


Figure 9 : Delay Lines for Search Window Delivery



III - EXAMPLE 1 : 720 X 576 X 25 HZ PICTURE, 8X8 BLOCKS, -8 TO +7 SEARCH WINDOW

III - 1 Introduction

The first example of a motion compensation implementation is based around a TV picture format as defined by the CCIR 601 recommendation, i.e. :

- picture size: 720 x 576 pixels. The example considers a 25 pictures per second application.
- 4.2.2. format : Picture defined by the three 8-bit components Y (luminance) U and V (chrominance). In 4.2.2. format each pixel is described

by two components : one sample of Y and one sample of U or V. The U and V components are sub-sampled (2:1) in the horizontal direction.

Note : If the picture is originally divided between two interlaced frames, it must be used in a non interlaced form for efficient motion estimation.

The study will concentrate on the motion compensation function considering that the samples are delivered in input in an 8 by 8 block's form (scanned column by column from top to bottom and from left to right) with an alternance of one block of Y and one block of U or V. After the computation delay, the motion compensation function delivers for the coder a predicted block corresponding to the input reference block and waits back from the decoder for the reconstructed block.

The study is made throught two architectures choices :

- the first one is the simplest one. It allows to point out some of the key points of such an application : frame memory choice, delay lines realisation, frame memory organisation and predictor access. The result is a frame address generator easy to realise but a solution that implies the use of very fast memories.
- the second one is a solution where the use of the same amount of frame memory is more optimised and the number of external components is reduced. On the other hand, the address generator is much more complicated to realise.

The reader who doesn't want to follow the different steps of the study and enter into details can directly refer to the architectures block diagrams in Figure 10 and 14 and jump to the conclusion paragraph III.4.

III-2 First architecture proposal

a) Frame memory choice

Pixel rate : 720 x 576 x 25 = 10.368 Mpixel/s. Let's name "f" that frequency. With two components per pixel, the input byte frequency is 2f = 20.736 Mbytes/s.

As the motion estimation is only made on the Y component the samples' rate on the STi3220 can be set to f = 10.368MHz. The U and V samples are directly sent to the coder.

Due to the high samples' rate it seems difficult to read the 3 search window bands directly into the frame buffer at 3f speed. The solution with two delay lines, that allows to reduce the number of accesses on the frame buffer, will be kept for that reason (refer to chapter II.3 for principle).

Frame buffer size = 720 x 576 x 2 = 414,720 x 2 = 829,440 bytes

Necessary bandwith on the frame buffer:

- "2f" for the picture reconstruction (Y and U/V components).
- "f" for reading the search window on Y component.
- "2f" for extracting the predicted blocks on Y and U or V components.

This is a total frequency of "5f" (roughtly 50MHz).

This frequency is too high for a high capacity static RAM or for a classical dynamic RAM.

Dual port dynamic RAM (also called video RAM) **fit well to the application.**

serial port: fast sequential access for reconstruction or search window delivery (regular block by block fashion).

random port: random access for predicted blocks read. But 2 accesses still remain to be done on the random port i.e. a frequency of 20.736MHz : this is too high for the existing components even if using the fast page mode : for instance a 256K x 4 video RAM, with access time of 100ns, has a 55ns read cycle when using fast page mode.

It becomes necessary to **split the RAM into two fields :**

one field for Y and one field for U and V (see Figure 10).

Each field is 411,720 bytes long constituted by four 256Kx4 VRAM. In that way it becomes possible to read in parallel the predicted blocks on Y and U/V frame buffers at "f" frequency. The rate adaptation, delivering alternatively the Y or U/V predicted blocks to the coder at "2f" frequency, is made thanks to two output FIFOs (one FIFO is only 64 bytes long).

The reconstructed blocks are sent back by the decoder to the frame buffer at "2f" frequency and are alternatively Y or U/V blocks. They will be written on the serial ports of each RAM: "2f" frequency is a cycle time of 48.22ns and the minimum write cycle time on the serial port of a 100ns video RAM is 30ns.

An optimisation consists in **writing a block only on one row of the memory** and not among two rows: in that way only one transfer is necessary between the Serial Access Memory (SAM) and the RAM for one block written.

As the reconstructed blocks are written alternatively on Y or U/V frame buffer, each serial port is only used one half of the time. When not used for block reconstruction, the serial port of Y buffer can be used for outputting a block of the search window. This output must be done at "2f" frequency and a rate adaptation is necessary for the STi3220 working at "f" frequency : this is done with a 64-byte FIFO.

b) Memory organisation

Four 256K x 4 video RAMs are used for storing the 414,720 bytes of one frame buffer (Y or U/V). They are organised as a 512x512 nibble array. In order to optimise the use of their serial ports, each block must be stored on only one row of the memory.

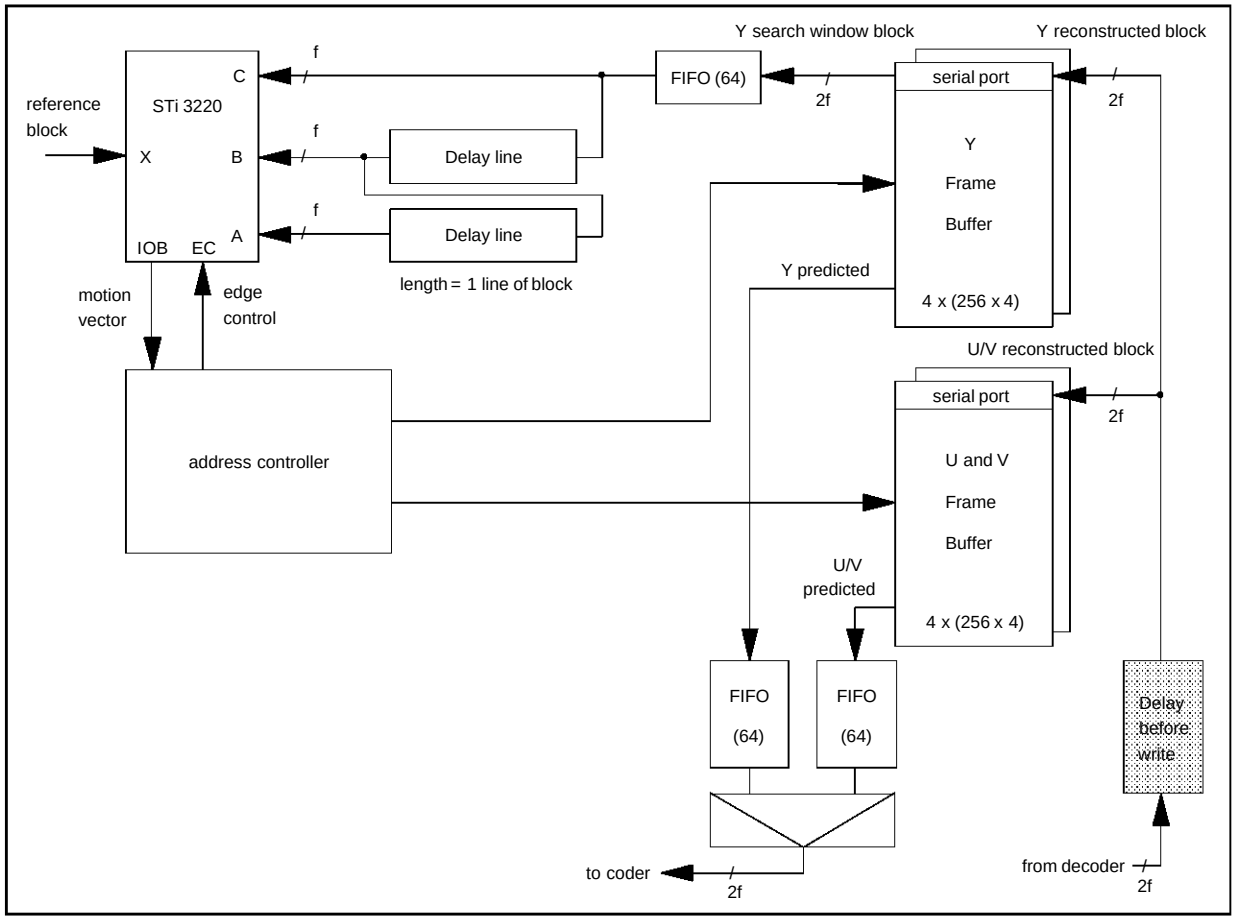
The memory space organisation becomes:

picture						
8	**8**					
	1	2	3	...	89	90
	91	92	93	...	179	180

frame memory								
	64							
row 0	1	2	3	4	5	6	7	8
row 1	9	10	11	12	13	14	15	16
.	.							
row 10	81	82	83	84	85	86	87	88
row 11	89	90						
row 12	91	92	93	94	...			

The picture format is 720x576 that means 90x72 blocks 8x8. As the 64 samples of a block are all stored on the same row, one row of 512 bytes is able to store 8 consecutive blocks. One line of block is stored on 12 consecutive rows, the last one containing only 2 blocks. With 72 lines of block in an image, the number of rows used in each frame buffer will be 72 x 12 = 864 rows (512 x 2 = 1024 rows available)

Figure 10 : First Architecture Proposal



c) Delay lines

Delay lines are necessary on two different points of this motion compensation architecture :

- a first set of two delay lines, already mentionned, is necessary for providing the STi3220 with the search window. Each delay line is one line of block long i.e. 720x8 = 5760 bytes long. The delay line can be made with an 8Kx8 SRAM associated with an address counter (see Figure 11). Each byte of the SRAM pointed to by the counter is first read out and then written with the new value to input in the delay line (read modify write). When the counter (in fact decrementing) reaches 0 it is re-loaded with the line length value. As two accesses to the RAM must be done during one cycle (1/10.368MHz = 96.45ns) the RAM cycle time must be at least 40ns.
- a second delay line is necessary to delay the reconstruction of the picture into the frame memory. As a matter of fact, the reconstruction of the new picture must not destroy pixels of the old picture that are still necessary, in particular for prediction.

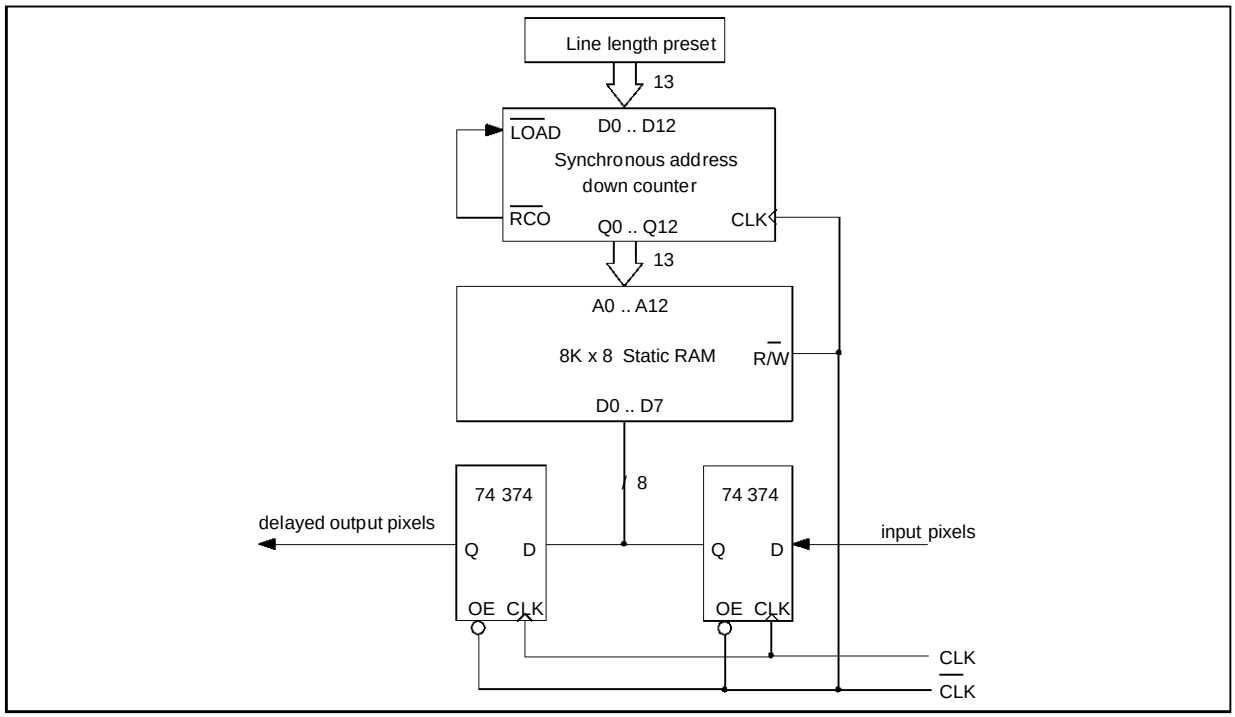
...	B1-1	B1-2	B1-3	B1-4	...
...	B2-1	B2-2	B2-3	B2-4	...
...	B3-1	B3-2	B3-3	B3-4	...

For instance, when the reconstructed block corresponding to reference block position B1-2 is delivered by the decoder to the frame buffer, it is not possible to write this block in the place occupied by B1-2 as long as B1-2 can be used. The last use of B1-2 may happen when delivering the predicted block corresponding to reference block B2-3 as B1-2 is in the search window centered around B2-3.

Therefore the delay between the time a predictor is extracted from the frame memory and the time when it is possible to write the reconstructed block into the memory is one line of blocks plus two blocks i.e. (720 + 2x8) x 8 = 5888 pixels. As each pixel is equivalent to two bytes (Y and U/V), this delay line length must be 11776 bytes (minus the cycles lost outside the motion compensation part for computation in the coder and the decoder).

The delay line can be made with the same principle than the previous ones, but taking care of the fact that the working frequency is ”2f”, i.e. a cycle time of 48.2ns. To cope with such a rate it could be possible to use two sets of two 16K x 4 static RAM in flip-flop: one set is written while the other is read. This solution is quite expensive.

Figure 11 : Example of Delay Line Implementation

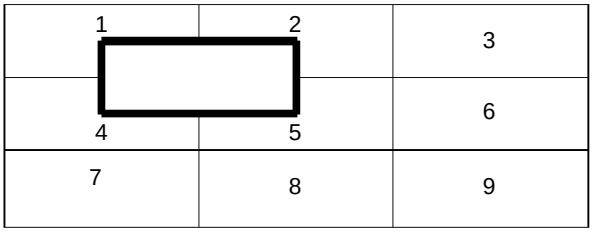


To avoid those external SRAM, the new reconstructed frame (let's call it frame 2) can be stored in the free address just after the last address of the current frame used for prediction (let's call it frame 1) as it can be seen in Figure 12. The only condition to implement such a structure is to have a free area in the memory higher than the requested delay. When the read reaches the end of frame 1, the address generation only continues its increase for starting the read of frame 2: the address generation is cyclic over all the memory space.

d) Predictor access

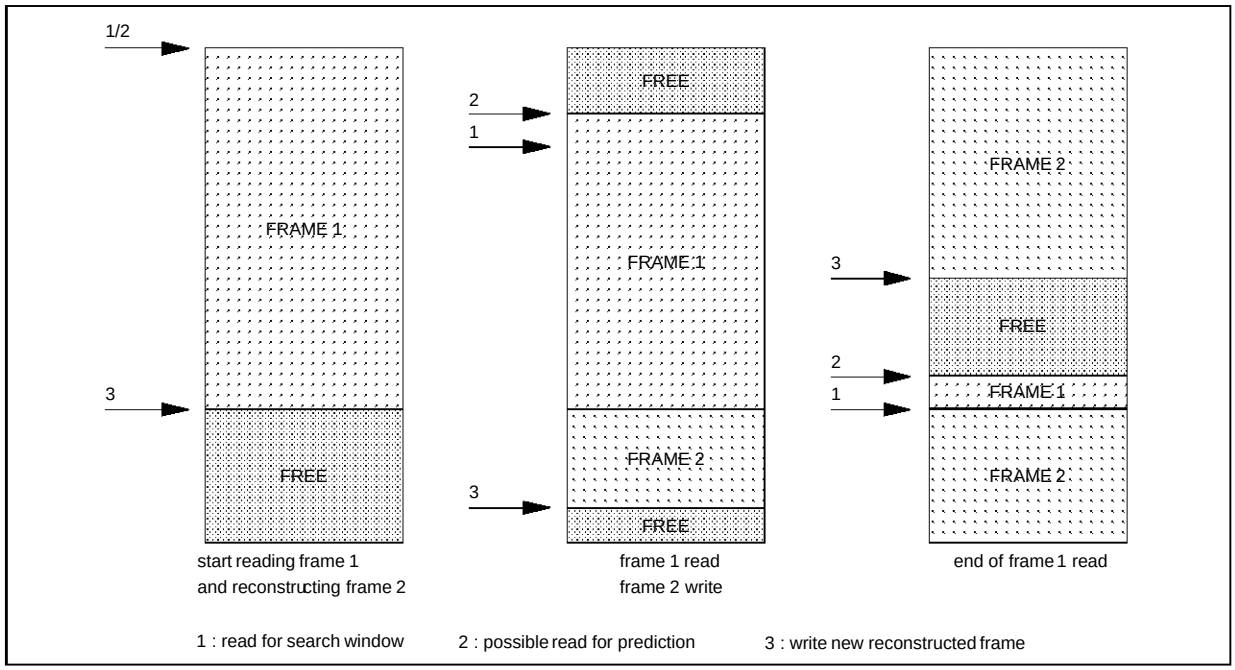
As seen before, reading a predicted block must be done at "f" frequency, i.e. a cycle time of 96.45 ns. However the access time on the random port of a 100ns video RAM is only 190ns. In order to cope with an average access time of 96ns, the page mode access of the video RAM must be used as much as possible : as a matter of fact, the row and column addresses (RAS and CAS selection) must be preset for each random access while for a page mode the row is only selected once (RAS), all the following accesses being done on the same row (also called page) with only a selection of the good column address (CAS). The fast page mode access is only 55ns for a 100ns video RAM like the HITACHI's 256Kx4 HM534251.

The addresses for the predicted block must be calculated depending on the motion vector delivered by the motion estimation chip STi3220. This motion vector is delivered by the STi3220 on the 36th cycle after the end of the reference block (in pipeline mode, it is equivalent to the 36th cycle after the beginning of the next reference block). As said before one block of the picture is not written among two pages of the memory. Nevertheless, except for a null motion vector, the predicted block involves several blocks of the frame memory i.e. several pages. In the example below the predicted block involves pixels from blocks # 1, 2, 4 and 5.



In order to output the predicted block in a column by column order, it is necessary to select a new page twice during each column (access to block #1 followed by block #4 in order to output the first column of the predicted block in the example above). Thus two RAS are necessary for 8 CAS: 2 random accesses and 6 fast page mode accesses. With a 100ns VRAM, reading 8 bytes of a column is 2x190 + 6x55 = 710ns long, i.e. for reading a predicted block : 710ns x 8 = 5.68µs.

Figure 12 : Illustration of Frame Stores in the Memory Space



While reading a predicted block, it is also necessary to manage the serial port of the dual port RAM. A maximum of three transfers between RAM and SAM is necessary :

- one transfer of the reconstructed block written in the SAM to the RAM.
- Two transfers of RAM blocks needed on the SAM for delivery of the search window (the search window blocks are not in the same position than the reference frame blocks but shifted seven columns right (refer to chapter 2), i.e. that they cover 2 frame block's positions. Hence an additional RAM to SAM transfer may be necessary if the two consecutive blocks are not stored on the same page).

The time for reading the 64 samples of a predicted block becomes:

710x8 + 3x190 = 6.25µs. This is equivalent to 6.25/64 = 97.65ns per sample, i.e. a frequency of 10.24 Msample/s that must be compared to f = 10.368MHz.

The margin is not very safe with the components existing on the market at the time this note was written. But there is no doubt that new faster video RAM are or will be available at the time the reader is looking to those lines. For that reason, this solution must be kept in mind.

III-3 Second architecture proposal

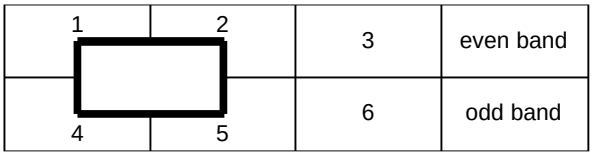
a) prediction

The speed limitation with the previous solution is mainly due to the fact that the page mode of the video RAM cannot be used efficiently when reading the predictor.

As the predicted block most often uses information on 4 blocks of the frame memory, it could be possible to read out all the samples needed for prediction in one of the 4 blocks, then all the needed samples on another block ... The number of random accesses is limited to 4 page changes. But the address generation would be very difficult to manage (no continuity) and an additional output RAM would be necessary in order to rearrange the samples and deliver them to the coder in a column by column order and not in the way they have been extracted from the frame buffer.

Therefore it appears as **a necessity to split again the frame memory in two new parts.**

An interesting cut of the frame memory consists in storing **all the even lines of block in one memory field and all the odd lines in another field.** The predicted block always straddles one even and one odd band of block.



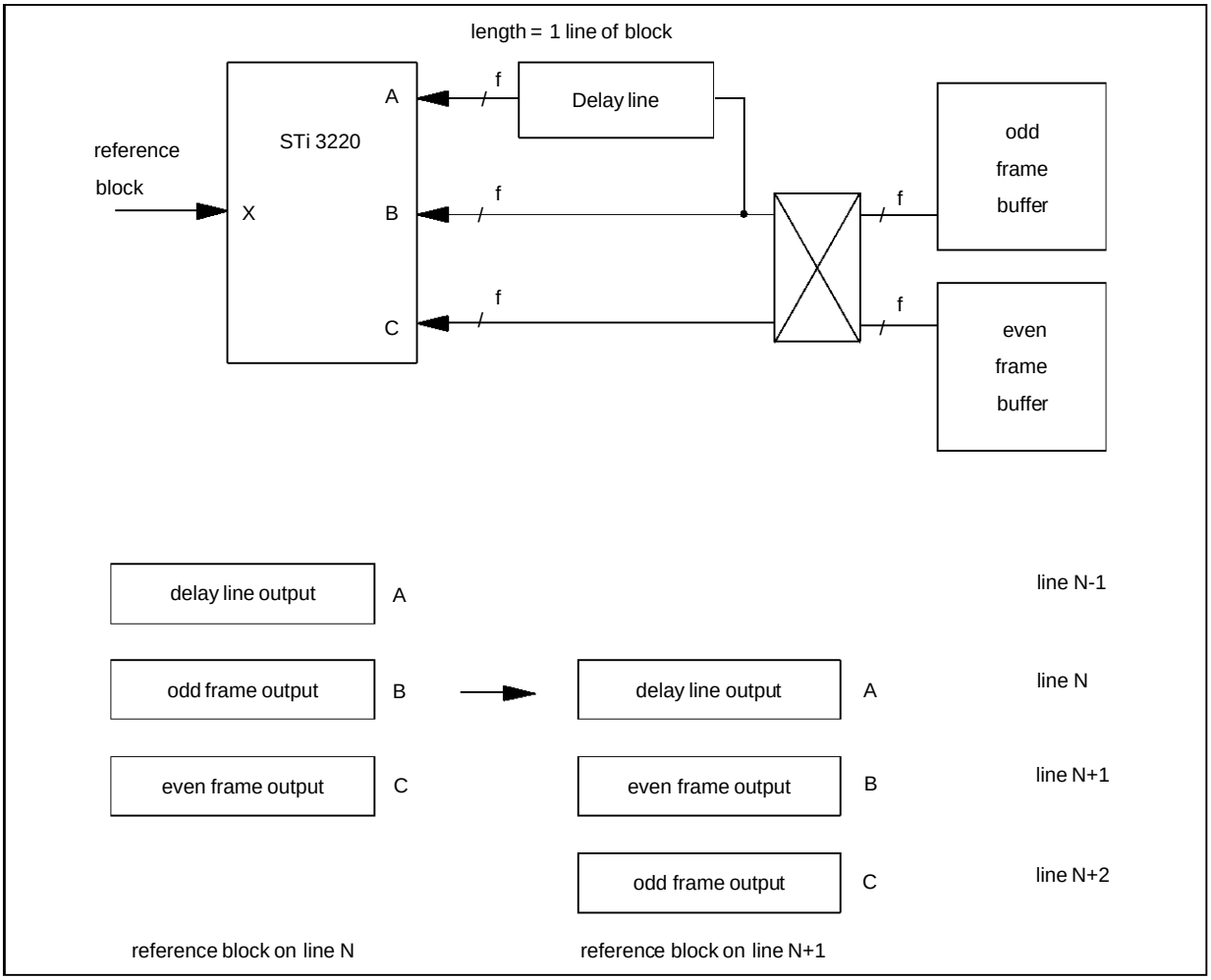
In the example above, block #1 is on even memory field and block #4 on odd memory field. The first columns of the predicted block are selected by a multiplexer between odd and even fields. On each field the selected page doesn't change during all the predicted block output, except if blocks #2 and #5 are not on the same page than blocks #1 and #4 respectively. For reading a predicted block, the maximum number of page changes is limited to two.

Each RAM field becomes an half frame buffer i.e. 207,360 bytes implemented on two 256Kx8 video RAM. The remaining 57,784 bytes of each field is used to begin the storage of the reconstructed picture just after the current one as explained before in "delay lines" chapter.

b) Search window delivery

An interesting consideration of that architecture is that the search window uses simultaneously the odd and even fields of the memory. It is then possible to provide the STi3220 on B and C inputs with the ouptut of the odd and even fields and to generate the A input thanks to a delay line supplied with the B input samples (see Figure 13). This allows to suppress one more delay line replaced by a simple data multiplexer: during one line of blocks, the odd field is connected to B input while the even field is connected to C input. During the following line of blocks, the odd field is connected to C input and the even one to B input.

Figure 13 : STi3220 Supply from 2 Frame Buffers and One Delay Line (example on two consecutive lines)



c) Second architecture timings

Let's verify the reliability of the second architecture, shown in Figure 14 :

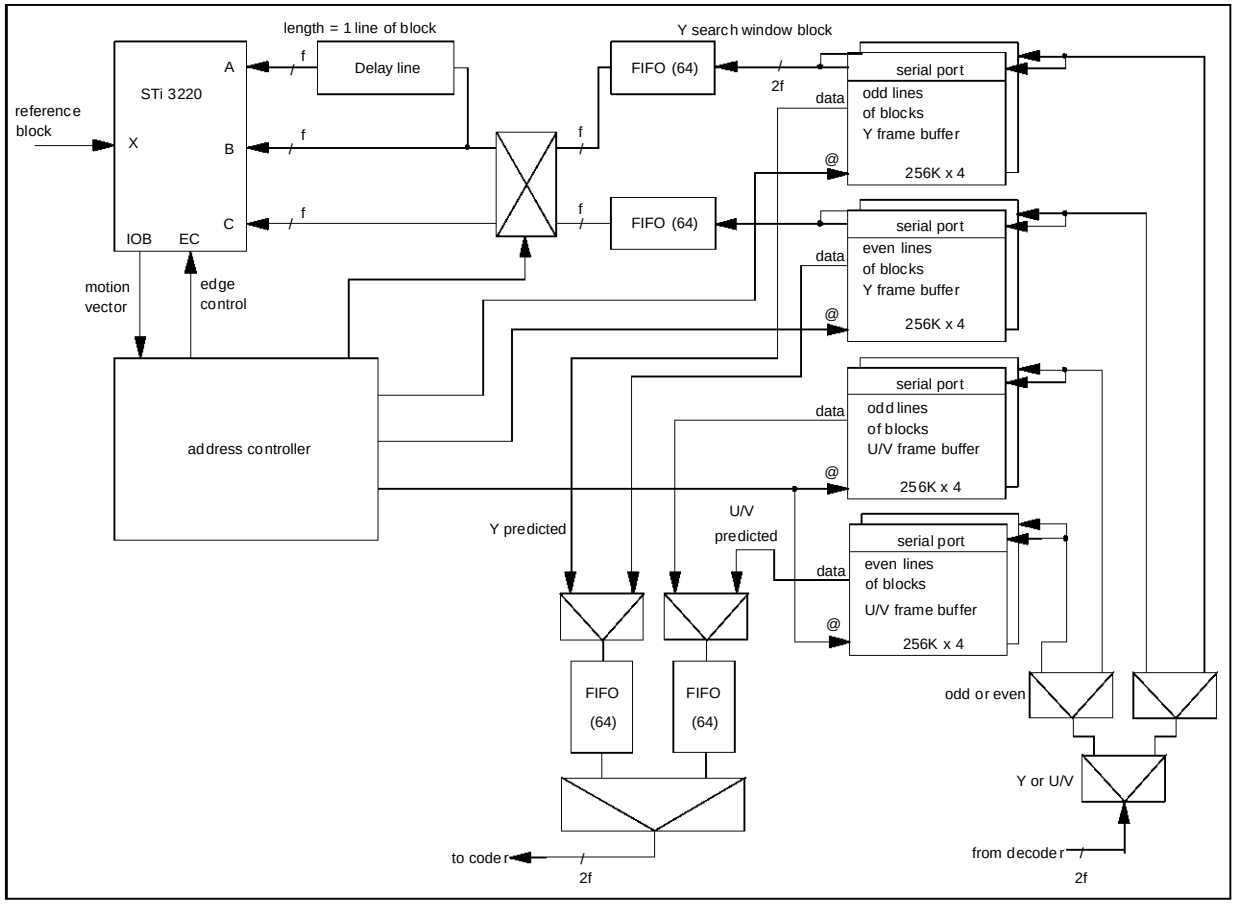
Each memory is accessed on the serial port alternatively for writing a block of the new reconstructed picture (this needs one transfer from SAM to RAM for each block) or for reading the search window in the previous picture (this needs one or two transfers from RAM to SAM). Each transfer between RAM and SAM may cost two random cycles : one for waiting for a current random access on the RAM to be finished and one for transferring the desired row between RAM and SAM. The maximum time lost for two blocks on the serial port is 6 random cycles. Due to transfer cycles, read or write on the serial port must be made at a frequency higher than "2f" .

The random port is accessed for reading the predicted blocks at "f" average frequency. Due to the seperation odd/even lines of blocks, we saw that only two page selections are necessary on each RAM. But it is also necessary to manage the serial memory exchanges with the RAM : as seen before this is a maximum of 3 transfers to do, each transfer followed by a new selection of the page currently read. A total of 8 random accesses may be lost for 5 samples read out, the remaining 59 samples being accessed in fast page mode.

With a random access = 190ns and a fast page mode access = 55ns, the global time for reading a predicted block on the random port is : 8 x 190 + 59 x 55 = 4.765µs. The margin with the time allowed (64/f = 6.17µs) is very safe in that case.

For instance if the fast page mode cycle is fixed to 70ns and the random access to 210 ns (3 x 70) then the total time for reading a predicted block is 8x210 + 59x70 = 5.81µs.

Figure 14 : Second Architecture Proposal



With such clocks, the serial clock cycle can be fixed to 35ns (70ns / 2). The total time for writing a predicted block and reading a search window block is: 128 x 35 + 6 x 210 = 5.74µs compared to the allowed time 6.17µs.

There is no more problem of speed with this solution. Only small FIFOs (64 bytes) are necessary for rate adaptation between the variable samples rate around the frame memory and the fixed computation rate of the coder or the decoder. Of course synchronisation signals indicating the beginning of the blocks will also be necessary.

The time left after the end of a data burst (a block) on the video RAM and the beginning of the next burst will be used for RAM refresh cycles. One refresh can be done simultanously on each video RAM, after each predicted block read sequence, that means one refresh every 6.17µs. This is a total time of 3.16ms for the 512 rows of the video RAMs, compatible with the maximum refresh cycle time of 4ms.

d) Address controller

The address controller, that can be an ASIC or developped around EPLD and PAL devices, must manage several functions:

- SAM : Write of the reconstructed block on the good row and from the good column.
- SAM : Read of the good block for the search window : for instance if the reference block position begins at address n in the even field than the search window access must be done from address n + 7 columns = n + 56 in even field (B input of STI3220) and at address n + 56 (or n + 1 line of block + 56) in odd field (C input of STI3220).

For both read and write the address generation only consists in a counter incrementation. Even or odd field addresses are equal modulo one line of block.

- RAM : read of the predicted block depending on the motion vector delivered by the STI3220 chip on the 36th cycle after the end of the reference block. From the start address of the predicted

block, the address generation consists in a simple counter incrementation plus a selection of the even or the odd field output data.
- generation of the control signals for the RAM chips (including refresh cycles), for the multiplexer selections...

III - 4 Conclusion

This first example has been studied through two architectures quite similar. In both cases eight video RAM chips 256Kx4 are used that allow to separate the sequential access on the serial port from the random access on the parallel port:
- Read search window block on the serial port.
- Write of the reconstructed blocks on the serial port. One block must be stored on one row only to avoid time expensive page changes. The reconstructed frame is stored just after the previous one in order to avoid an external delay line.
- Read of the predicted blocks on the random port. The fast page mode must be used as much as possible to reduce access time. The second solution is much more optimised for reaching that goal.

For such an application the main parameter is the high pixel rate. That was the reason for the choice of search window delivery through delay lines in order to limit the access time on the frame memory. However, if the components are not fast enough, it becomes necessary to split the frame memory into several fields : in that way it becomes possible to read or write simultaneously several informations on several fields at the same time thus reducing the apparent speed in the same ratio. The consequence is that the more the number of fields, the more the number of associated data multiplexers and the more complicated the address generator.

For all those reasons, the first solution, with a memory cut in two fields (Y and U/V frames) is the simplest one (the address generator could be done with counters and PAL devices) but also the solution needing high speed memories. The solution has not enough time margin with the existing components on the market (100ns VRAM), but has been exposed here because the components will go faster and faster or because it can be used for slower pixel rate applications.

The second solution, with a memory cut in four fields (two fields Y and U/V, each one seperated into odd lines of blocks and even lines of blocks) is a better optimisation of the memory use and fits well with the existing components. An intrusting feature of the frame cut allows to suppress one delay line and to replace it by a multiplexer. On the other hand the address generator is more complicated but for a high volume application it can be realised with an ASIC thus providing a solution with a good components' speed/cost tradeoff.

IV - EXAMPLE 2 : CSIF PICTURE, 16X16 BLOCKS, -8 TO +7 SEARCH WINDOW.

IV-1 Introduction

The second example of architecture study is made around a CIF 30Hz format :

frame size = 352 x 288 pixels.
Pixel rate = 3.0413 Mpixel/s.

4.1.1. Format :

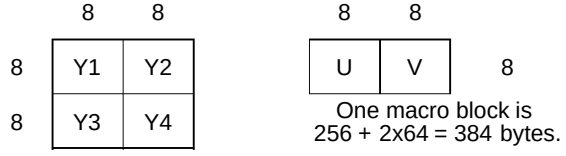
The chrominance components U and V are subsampled (2:1) in both vertical and horizontal directions. Four orthogonal pixels of the picture are defined by four samples of luminance (Y) one sample of chrominance U and one other sample of chrominance V.

pixels			
Pix O	Pix O	Pix O	Pix O
Pix O	Pix O	Pix O	Pix O
Pix O	Pix O	Pix O	Pix O

associated samples			
YUV O	Y O	YUV O	Y O
Y O	Y O	Y O	Y O
YUV O	Y O	YUV O	Y O

The motion estimation is made on 16x16 blocks of Y, while the coding, and hence the prediction is considered to be made on 8x8 blocks for the three components of the image. Due to sub-sampling ratio, each block 16x16 of Y is associated to one block 8x8 of U component and one block 8x8 of V. The combination of those 3 blocks is also called a macro-block. A CIF picture is composed by 22 x 18 macro blocks.

macro block composition :



One macro block is
256 + 2x64 = 384 bytes.

The samples are considered to be sent to the motion compensation part in the macro block fashion i.e. one block 16x16 of Y scanned column by column (one column is 16 bytes) followed by two blocks 8x8 of U and V also scanned column by column (one column is 8 bytes).

However the predicted blocks must be 8x8 blocks, scanned column by column and sent to the coder in the order:

Y1 Y2 Y3 Y4 U V.

Those blocks after decoding are input back in the same order for the frame reconstruction.

Four pixels of the picture being associated to 4 bytes of luminance and 2 bytes of chromimance, the total frame size is 352 x 288 x 1.5 = 152,064 bytes and the byte rate is 1.5 times higher than the pixel rate i.e. 4.56 Mbyte/s (219ns for one byte). This is the working frequency of coder and decoder.

Considering the slow byte rate of this example, the architecture is chosen with two imperatives:

- **use of classical Dynamic RAM** for frame buffer (e.g. a 100ns access time DRAM has a cycle time of 190ns).
- **access for search window directly in the frame buffer** to supress the external delay lines.

It seams also interesting, for simplification purposes, to manipulate Y and U/V samples in the same flow (no differenciation for search window).

As in the previous example, people in a hurry can directly refer to the architecture in Figure 18 and to the summary in the conclusion chapter IV.3.

IV-2 Architecture choice

a) Search window delivery

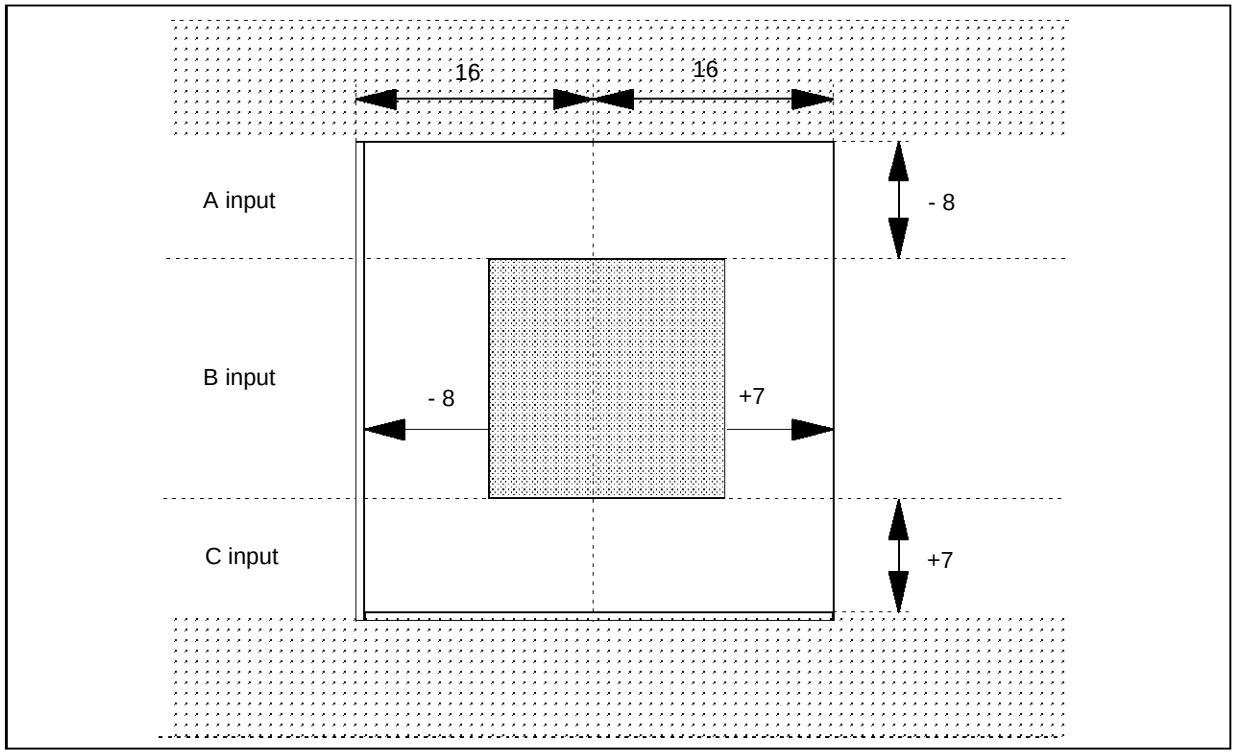
When doing motion estimation on 16x16 blocks with displacement vector in the range -8 to +7, only one half of the upper band of the search window is used (8 lower pixels) and only one half of the lower band is used (7 upper pixels) as shown in Figure 15.

It becomes possible to connect together inputs A and C of the STi3220 as they are not used at the same time.

Sharing the resources between A and C inputs doesn't make the use of two delay lines possible. The search window delivery must directly be made into the frame buffer with two accesses for each reference block sample input.

In order not to double the access frequency on the frame memory, it is possible to split the RAM into two fields : **one field for storage of even lines of macro blocks and one field for storage of odd lines of macro blocks.** In that way only one access is necessary on each field for providing the two search window samples. A data multiplexer will allow the connection of either the even or the odd field to either the B or A+C inputs of the STi3220.

Figure 15 : Search Window and Reference Block Position



b) Prediction

The predictor access will be made in an additionnal static RAM where the search window must be stored on the flight. In order to simplify the system, the U and V components will also be read in the frame buffer for delivery of the search window althought they are not used for motion estimation. When delivering the U or V samples on the A,B and C inputs of the STi3220 it is necessary to disable the chip for not taking care of those information: the EN pin must then be tired to high level. Due to their sub-sampling ratio, the motion vector for U and V components will be half the motion vector of Y component.

The total area necessary for storage of the search window is (see Figure 16):

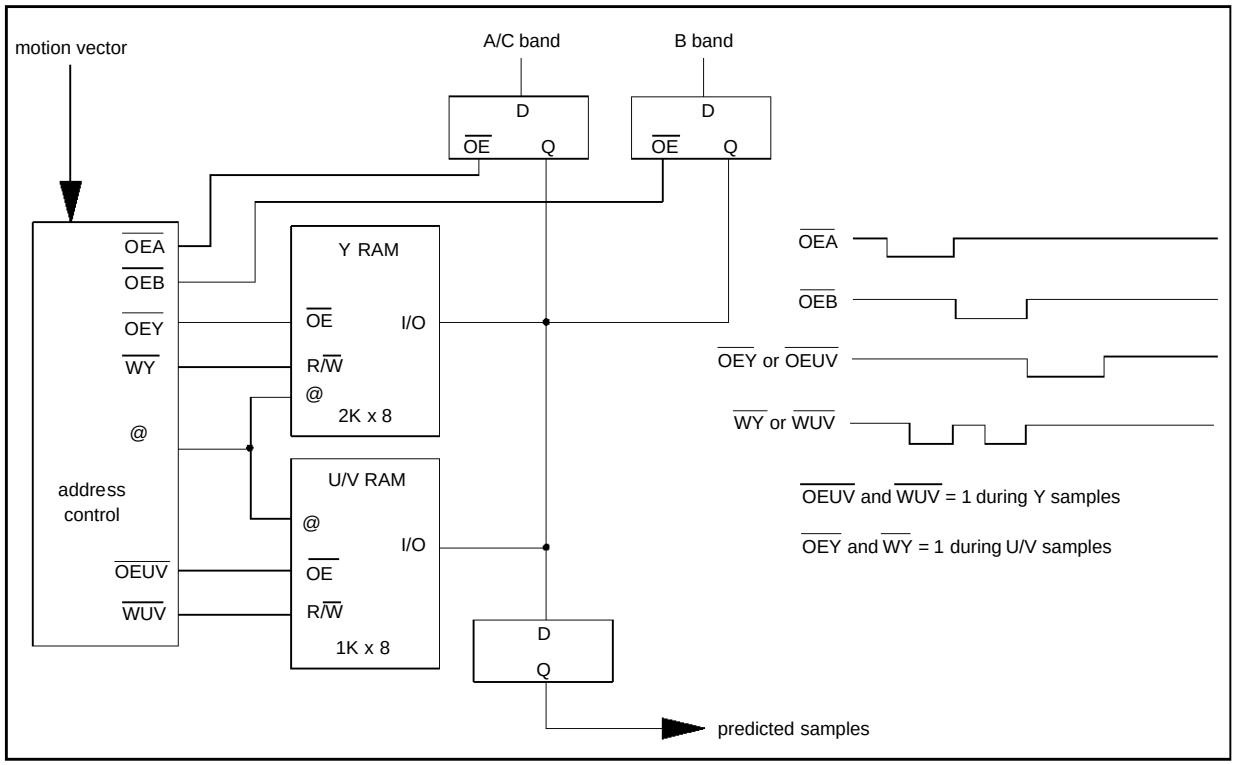
Figure 16 : organisation of the prediction SRAM.

A1	A2	A3	A4	half macro block height
B1	B2	B3	B4	one macro block height
C1	C2	C3	C4	half macro block height

- Four macro blocks for the current search window: 2 macro blocks for B band (B1 and B2) and 2 half macro blocks on either A or C band (respectively A1, A2 and C1, C2).
- One additional macro block on B band (B3) and two half macro blocks on A and C bands (A3 and C3): those macro blocks will be used in conjunction with A2,B2 and C2 for the next search window (let's say the second search window). During the time they are stored, the motion vector corresponding to the first search window and reference block is output by the STi3220 on the 46th cycle. We can consider that the motion vector can be changed by the external system and that it will be taken into account for prediction only with the first sample of the next macro block: that leaves 384 - 46 = 338 cycles for the external system to decide which vector must be used.
- One additionnal macro block on each band (A4,B4,C4) necessary to avoid destroying the first search window read out for prediction with the information of the third search window that must be stored in the SRAM (third search window = A3, A4, B3, B4, C3, C4).

Of course when the predictor of the first search window has been output, locations A1, B1, C1 are free for storage of the last blocks of the fourth search window (fourth search window = A4, A1, B4, B1, C4, C1).

Figure 17 : predictor organisation



The total space needed in the prediction SRAM must be 8 macro blocks i.e. 3072 bytes, that can be divided between one 2Kx8 SRAM for Y component and one 1Kx8 SRAM for U and V (see Figure 17). The SRAM are associated with two latches that temporarely store the samples on each band.

The SRAM is accessed 3 times: one time for storing a sample of one band (B for instance), one time for storing a sample of the other band (C if first half of a column, A if second half) and one time for reading one sample of the predicted block. With a sample rate equal to 219ns, the access time on the SRAM must be at least 73ns.

c) Frame memory choice

The last point to consider is the frame memory that must be shared between frame reconstruction and search window read. As the cycle time of the memory (190ns for a 100ns DRAM) is not far from the byte cycle time (219ns) it is not possible to do two accesses during one cycle : the frame buffer must again be split into two new parts. The samples will be read or written by pair, one memory dedicated to odd samples the other to even samples.

This leads to 4 memory spaces : two fields for storing odd or even lines of blocks and each field constituted by two memories for storing 2 samples in parallel. Each memory must be 38,016 bytes long. The nearest existing DRAM implies the use of 8 DRAM 64K x 4 with access time not higher than 100ns to cope with the 219ns cycle time.

The proposed architecture is shown in Figure 18.

The frame memories are associated to 6 latches. As a matter of fact four samples are read out from the two fields of two memories during one cycle (needing four latches for temporal storage) and two samples are written in one field of two memories during the following cycle (needing two latches plus a multiplexer for selection of the good field).

d) Memory organisation and refresh

The 64Kx8 DRAM are organised as a 256 x 256 array.

One macro block is 384 bytes shared between two DRAM : so each half macro block of 192 bytes is stored over 2 DRAM.

The picture contains 22 x 18 = 396 macro blocks shared between two fields of DRAM , so each DRAM contains 198 half macro blocks of 192 bytes.

If each half macro block is written on a column of the DRAW (198 columns used in each RAM), each time a macro block is accessed (and there is always a macro block read in each DRAM for search window delivery), the corresponding 192 rows are addressed and hence refreshed. The total time needed for reading a macro block is 384 x 219ns = 84.096µs. This is also the cycle time for refreshing all the useful rows of the memory : it is compatible with the maximum refresh cycle times of 4ms defined for the 64K x 8 DRAM.

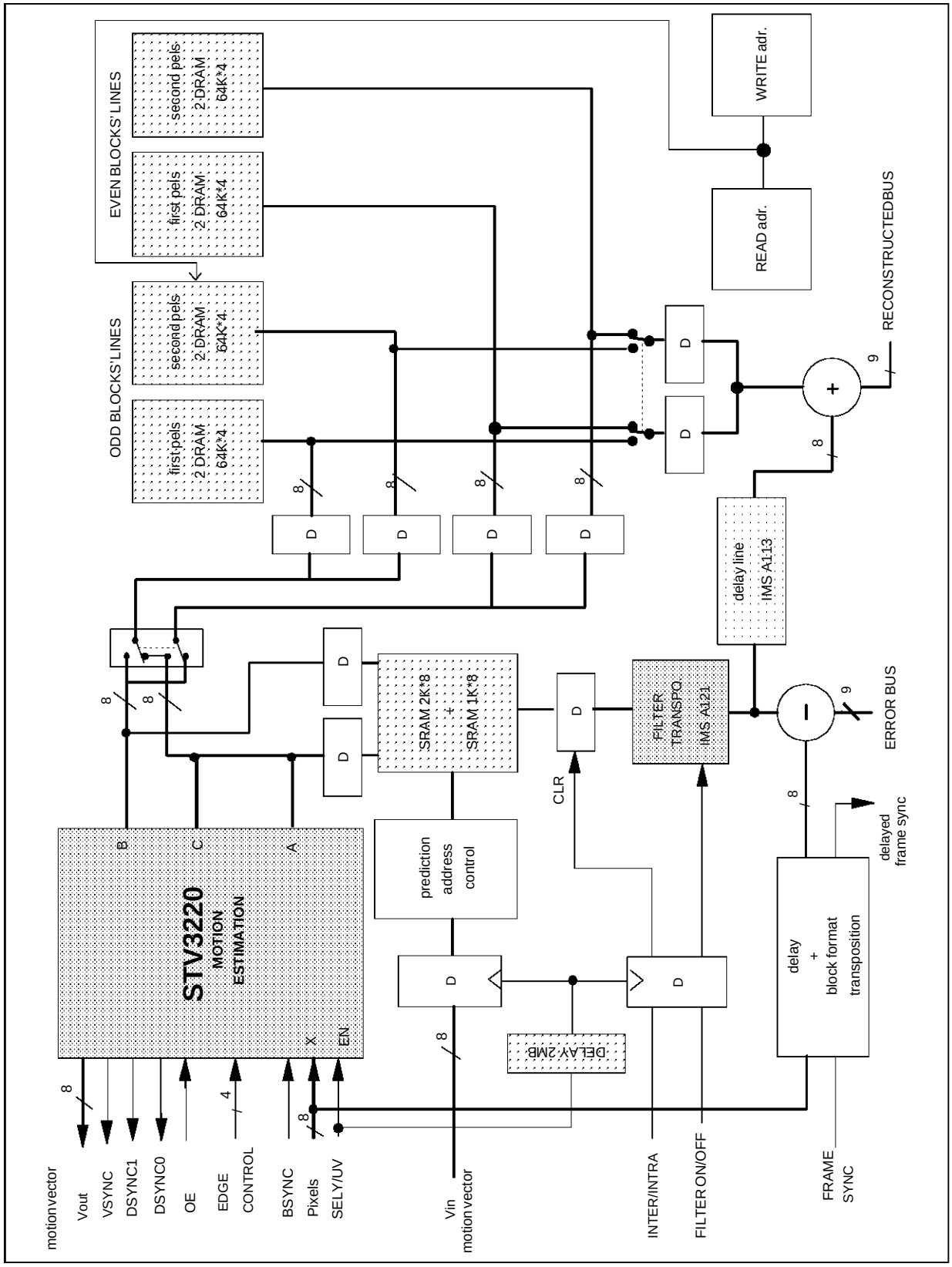
Macro block n :

y0	y1	y3	...	y240
y1	y17	y33	...	y241
y2	y18			
y3	y19			.
y4	y20			.
y5	y21			.
...				.
y14	y30			
y15	y31		...	y255

u0	u8	...	u56
u1	u9	...	u57
u2			
u3			.
u4			.
u5			.
u6			
u7	u15	...	u63

v0	v8	...	v56
v1	v9	...	v57
v2			
v3			
v4			
v5			.
v6			.
v7		...	v63

Figure 18 : Example 2 General Architecture



Memory organisation for storage of macro block n in odd or even field :

	DRAM1 - column n	DRAM2 - column n
row 0	y0	y1
row 1	y2	y
row 2	y4	y5
...		
row 7	y14	y15
row 8	y16	...
...		
row 127	Y254	y255
row 128	u0	u1
row 129	u2	u3
...		
row 159	u62	u63
row 160	v0	v1
row 161	V2	v3
...		
row 191	v62	v63

Rows 192 to 255 are not used. Macro block n+1 is stored on column n+1.

Note: The two memories in each field are always accessed with the same address.

e) Address control

In order to simplify read , the write address generation should take care of the fact that the Y samples are sent back by the decoder in 8x8 blocks (after inverse DCT) and that they will be read in 16x16 blocks for motion estimation: the samples should be stored in a way such as the read only consists in increasing the row addresses. Of course the U and V 8x8 blocks are stored in the way they are coming from the decoder just after the Y block in the memory.

When a memory field is dedicated to the middle band of the search window (B input), the column address doesn't change during one macro block. When the field is dedicated to lower and upper bands (A and C inputs) the 8 last bytes of a column are read out before the 8 first ones (for Y) so it is necessary to change the column address every 8 samples (for Y) or 4 samples (for U or V) but with a row address always increasing from the beginning to the end of the macro block.

f) Filter

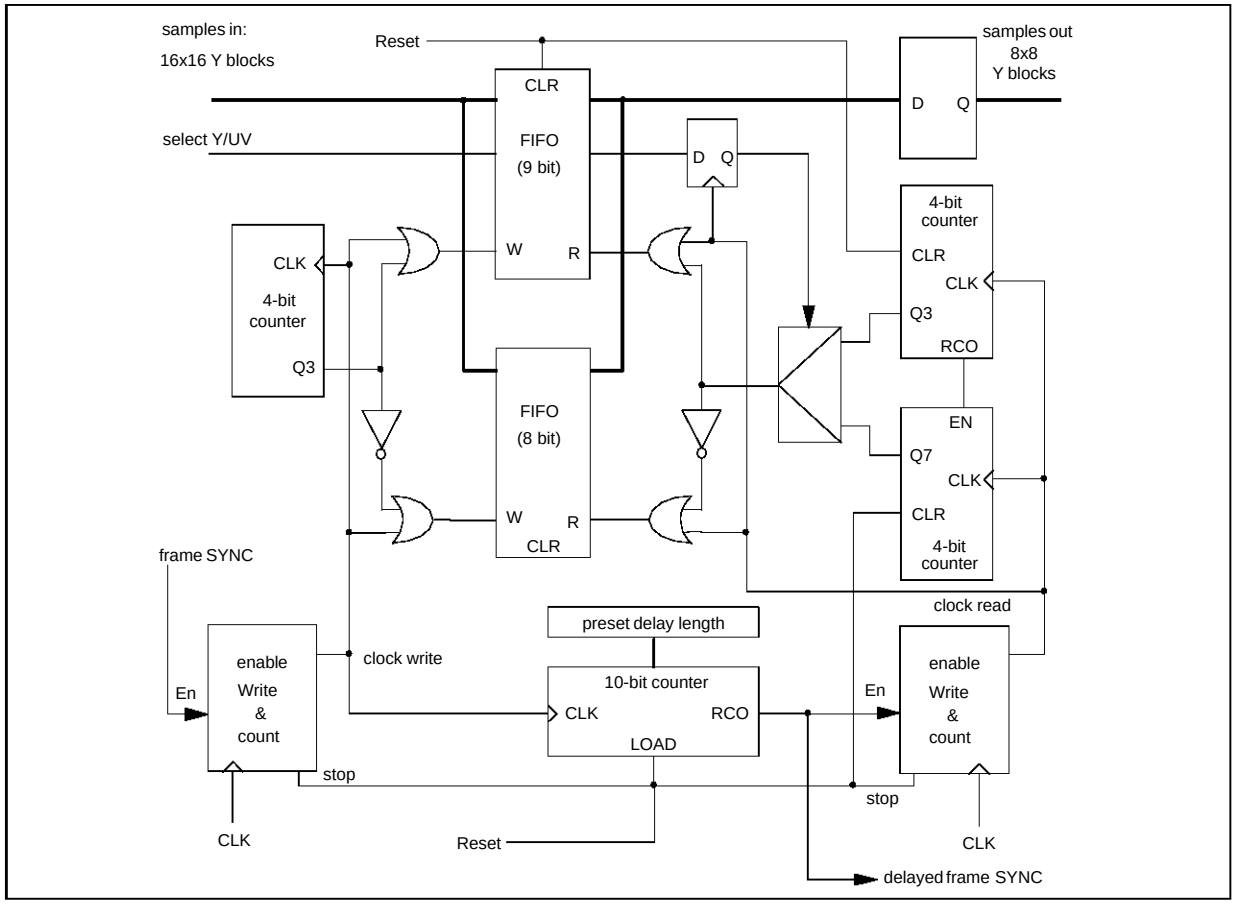
In the architecture scheme shown in Figure 18 there is a filter on the predictor output : this filter can be used to increase the efficiency of the prediction by reducing artefacts due to high frequencies. It is realised with an IMS121 device from SGS-THOMSON that is able to perform either Discrete Cosine Transform, Inverse DCT, filtering or matrix transposition. In that case the filter is used that associates with the input predicted block a low-pass filtered output block delivered in a transposed order. Even when the filter is not used in the loop, the IMS121 must be used to do transposition of the predicted blocks in order that they can always remain with the same scanning order in the output stage.

Note: the filtering is done on 8x8 blocks and not 16x16 : that is one of the reasons why the prediction must be made on 8x8 blocks. The delay for the filter or transposition function is 128 cycles.

g) Error blocks = reference - prediction

The filtered predicted blocks must be subtracted from their corresponding reference block to provide the error blocks, sampled over 9 bits, that will be computed by the coder (for instance throught DCT). Those blocks must be 8x8 blocks. The transformation from 16x16 to 8x8 blocks is already done for the predicted blocks when reading the search window RAM, but it must be made for the reference block that was sent to the STi3220 in 16x16 form.
The delay between the time a reference block is input in the STi3220 chip and the time it is subtracted from the predicted block is 2 macro blocks for reading the predicted blocks and 128 cycles for filtering i.e. 896 cycles. Of course this is an approximation as the number of cycles depends on the final schematic diagram.
The proposed architecture for delaying the input reference blocks and providing the block format conversion is shown in Figure 19.

Figure 19 : Block Delay + Format Change



When starting, the write on the FIFOs is disabled until the first frame synchronisation appears (with the first pixel). Read on the FIFOs is prohibited until the delayed frame sync. (generated by the counters preloaded to the delay length = 896 cycles) is not delivered.

In write mode the first 8 samples of a column are written on the first FIFO, the 8 following on the second FIFO : the first FIFO will always contain the upper 8x8 blocks of a 16x16 block and the second FIFO the lower ones. In read mode the first FIFO is read 128 consecutive cycles for delivery of the two first 8x8 blocks, and than the second FIFO during 128 cycles for the two following blocks. For U and V blocks (already in 8x8 format) the FIFOs are read in the way they are written.

h) delay predicted blocks

The filtered predicted blocks must not be lost after delivery of the error blocks as they will be used for frame reconstruction in addition with the reconstructed blocks (after decoding). For that purpose they must be delayed during the time the error blocks are computed in the coder and sent back to the board by the decoder. This delay can be made using the SGS-THOMSON A113 chip: this device is a delay line whose length can be programmed from 8 to 1320 steps.

IV-3 Conclusion

The most interesting feature of this architecture is the use of very popular (and hence cheap) components like the 64Kx4 DRAM or the small 1Kx8 or 2Kx8 SRAMs.

Like in the previous example, in order that the DRAM cycle time (190ns for a 100ns DRAM) fit with the input byte cycle time (219 ns) a split of the frame buffer in several fields is necessary:

- one odd lines of blocks field and one even lines of blocks field. On each cycle, one sample is read out from each of the two fields providing the two samples necessary for the search window supply.
- each of the two fields is divided in two parts: in that way, during one cycle two bytes are read out in parallel from each field (one byte in each

sub-part) and during the following cycle two bytes are written in parallel in one of the fields, for frame reconstruction.

The total frame memmory is composed by eight 64Kx4 DRAM.

The frame buffer is only used for search window delivery and for frame reconstruction. The prediction is made in an additional RAM where the search window is stored on the flight as it is sent to the STi3220 inputs. In order to simplify data control, the U and V samples are computed in the same flow than the Y samples. The additional RAM is a 2Kx8 SRAM for Y search window and 1Kx8 for U/V search window.

As the prediction is seperated from the frame buffer, the address generator of the frame memory is greatly simplified and could be realised with several counters and PAL devices.

This architecture is the simplest and the cheapest of the three examples of this application note. Consequently, it is also the architecture that is the fastest to implement with standard components for prototyping a codec. For high volumes, the memory address generator and the predictor could be integrated into a ASIC device leading to a block diagram with the STi3220, the DRAMs and the control ASIC as main devices.

V - EXAMPLE 3 : CIF PICTURE, 16x16 BLOCKS, -16 TO +15 SEARCH WINDOW.

V-1 Introduction

The picture format considered in this example is exactly the same than the previous one :

30 frames/s, 352 x 288 pixels/frame, 12bit/pixel (4.1.1 format): frame size = 152,064 bytes.

But in this example we want to compute a motion vector in the range -16 to +15 in both directions.

The study first explains how to compute such a search window with only one STi3220 able to cover a -8 to +7 range, then a proposal of architecture is made (refer to figure 5.2) followed by an explanation of the search window supply mechanism. Finally, like in the other examples the study ends with a verification of the different accesses that must share the frmae memory bandwith. A conclusion chapter (5-6) is available for people who just want a quick overview.

V-2 Search window computation

a) Search window definition

As the STi3220 chip is only able to compute motion vectors in the range -8 to +7, four different computations over four sub search windows are necessary to cover the range -16 to +15. The sub windows selection is shown in Figure 20.

The four resulting motion vectors are partial ones. In order to determine what will be the final motion vector for the total search window, the minimum distortions, corresponding to each partial motion vector for the total search window, the minimum distortions, corresponding to each partial motion vector, must all be compared. No particular algorithm will be given here to determine what must be the final motion vector as it is a part of the coding policy.

The motion vector can be directly associated to the minimum of the four distortions : for instance, if the minimum is obtained in the sub search window 2, then the resulting motion vector (MV) is the partial motion vector 2 (MV2) plus 8 in horizontal direction and minus 8 in vertical direction.

b) How to compute sub-search windows

The four different calculations for each sub search window can be implemented in two different ways :

- using four STi3220 chips in parallel each one working on a different sub search window. After an initialisation sequence and a block sequence on each STi3220 (i.e. 512 cycles), the computation can be done and the partial motion vectors obtained after 46 new cycles on each chip. This solution is only useful in high speed systems and is an heavy and expensive solution. In an application like the CIF format it is cheaper to use the second solution.
- using only one STi3220 doing all the partial computations on the four sub search windows. Each time a sub search window is computed, an initialisation sequence and a block sequence are necessary. It can be noticed on Figure 20 that the initialisation sequence of sub search windows 2 and 4 are the same as the block sequence of sub search windows 1 and 3 respectively. Hence the pipeline mode can be used between sub search windows 1 and 2 (or 3 and 4). Sending all the necessary data to the chip will cost 6 blocks 16x16 i.e. 1536 cycles. Each time the chip is in a block sequence, the 16x16 reference block must be input on the X bus for comparison with the sub search window. Of course the results of each sub search window, obtained on the 46th cycle following a block sequence, must be stored in order to be analysed at the end of the process (at least the motion vector and the minimum distortion must be kept).

All the sub window scanning must be done during the time of one input reference block. As six sequences are necessary the chip must work at a speed 6 times higher than the Y reference input rate.
Considering the pixel rate = 352x288x30 = 3.041MHz, the chip must run at 18.25MHz i.e. at its maximum working frequency. It implies that Y and U/V samples cannot be mixed on the same flow as the speed would be 1.5 times higher.

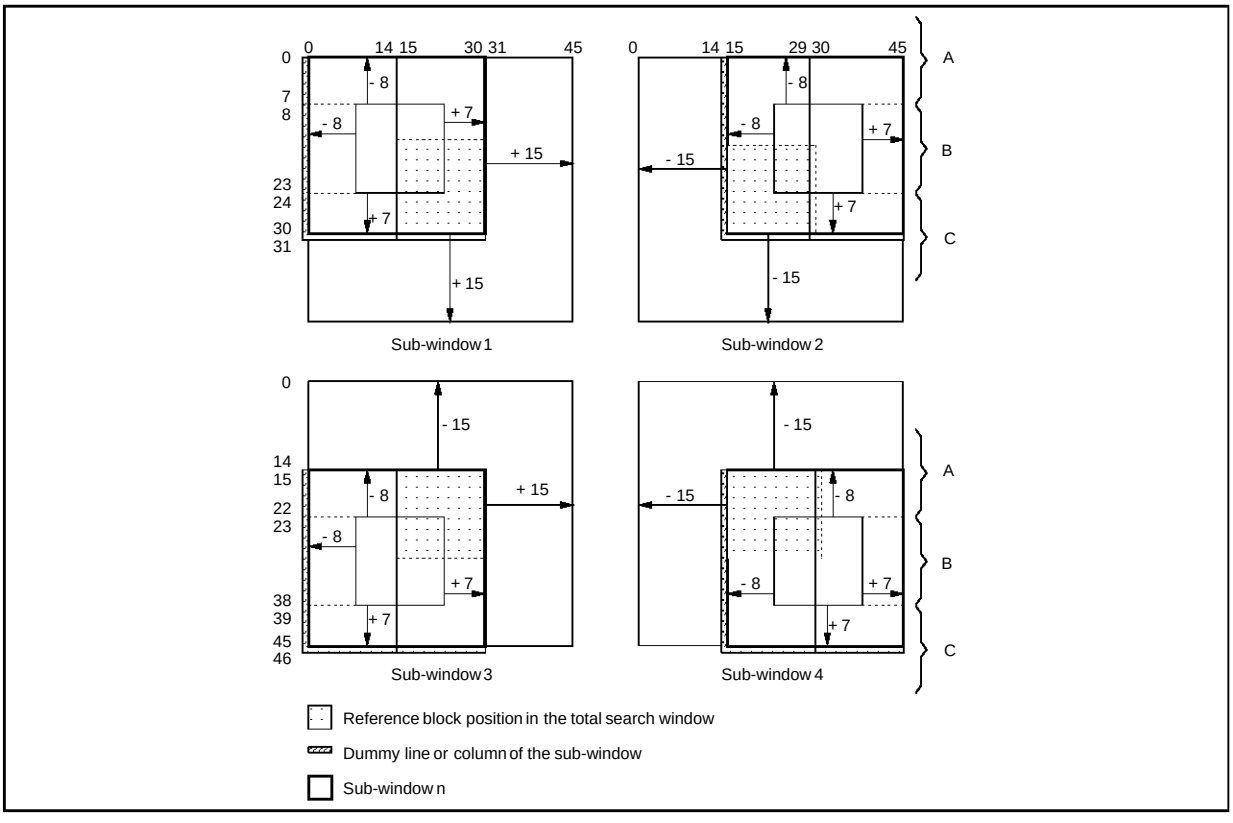
V-3 Architecture study

Due to the division of the search window in four different parts it is not possible to use the delay lines structure in this application.
It could be possible to split the frame memory in two parts (odd and even lines of blocks) in order to access in parallel to the two samples needed for the search window. This implies the use of video RAM in order to cope with the samples rate (about 55ns) but this would lead to memory sizes of 152,064 / 2 = 76,032 bytes which is not optimum with the existing component sizes (64K x 4 or 256K x 4). Hence we will consider a solution with only one frame buffer constituted by two 256K x 4 video RAM memories.

It is not possible to output from the frame buffer all the blocks needed for the search window during the time of only one reference block : the frequency would be 12 x f if f is the pixel's frequency.
A small static RAM will be added to the main frame buffer in order to store the search window : in that way, for each new reference block in the pipeline, only the three new right-most blocks of the search window are necessary (see Figure 20). As it is necessary to provide the search window inputs of the STi3220 with two samples on each cycle, the SRAM is in fact separated in two parts : one SRAM for storing the middle band's search window blocks and one SRAM for storing the upper and lower bands of the search window (as they are not accessed at the same time). The transfer of the new blocks from the frame buffer to the two SRAM will be made throught two FIFOs for rate adaptation.

The last point to consider is that all the SRAM data are read out in the same way they are written: this property induces to use FIFOs instead of SRAM in order to simplify the external control (no counters for address generation).
An architecture able to support this example is shown in Figure 21 and is going to be analysed in the following lines.

Figure 20 : Four Sub Search Window for -16 to +15 Displacement



V-4 Search window delivery

Note: For each block of the search window, the 8 last pixels of each column are sent to the STi3220 chip on B and C buses, followed by the 8 first columns sent on buses A and B (refer to Figure 20). In order to provide the search window blocks in the good order, the output of the frame memory is delayed by 16 cycles (refer to Figure 21): the 8 first samples of a column are first written into the 16 cycles delay device, then the 8 last samples of the column are read out from the memory for search window delivery (they are also written in the delay line). During the 8 following cycles the 8 first samples of the next column are read out from the memory and stored into the delay line while the 8 first samples of the current column are extracted from the delay line for search window.... The delay for delivering the search window is equivalent to 8 cycles.

Let's try to analyse the different phases of the search window delivery, with reference to Figure 22.

Note: The blocks noticed in Figure 22 have not the same borders than the blocks of the image: as a matter of fact their relative position is shifted one column left. The information will be extracted from the frame buffer blocks after blocks but it is easy, with the rate adaptation provided by FIFOS 1A and 1B, to read out the blocks for search window with a relative position anticipated by one column. For simplification purpose we will refer in the explanations to the search window blocks and not to the image blocks keeping in mind that one search window block = one column of a block image N and fifteen columns of block image N+1.

Figure 21 : Example 3 Architecture

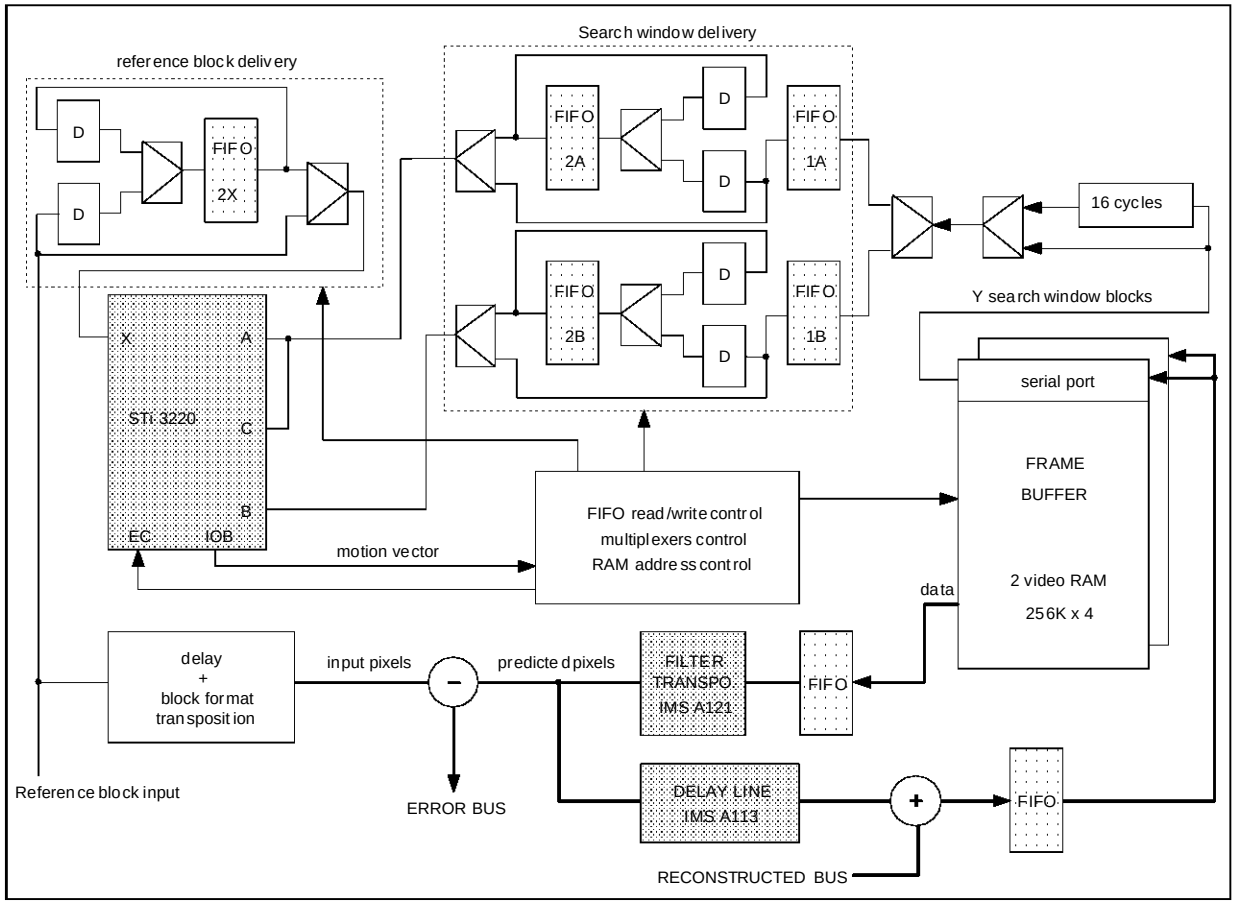
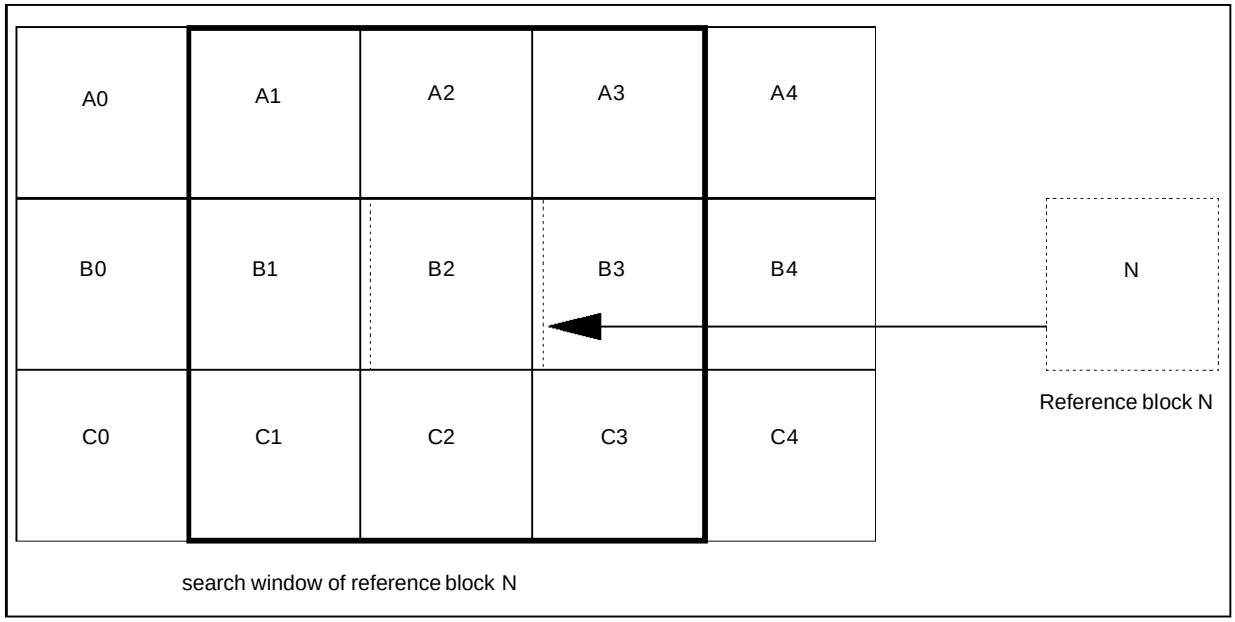


Figure 22 : Search Window Block Notation



The sequences on the STi3220 inputs for reference block N must be:

A/C input :	A1	A2	A3	C1	C2	C3
B input :	B1	B2	B3	B1	B2	B3
X input :	..	N	N	..	N	N

The input sequences for the next reference block N+1 will be :

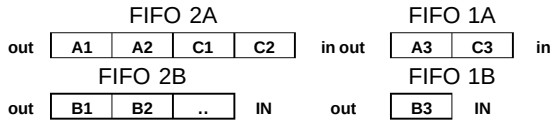
A/C input :	A2	A3	A4	C2	C3	C4
B input :	B2	B3	B4	B2	B3	B4
X input :	..	N+1	N+1	..	N+1	N+1

As it can be seen on those two reference block sequences some blocks of the search window are used several times either for the same search window or for the next one. Hence when outputting a block from the FIFOs it may be necessary to write it back into the same FIFO: the output of the FIFOs 2A and 2B may be connected to their own input.

Search window processing:

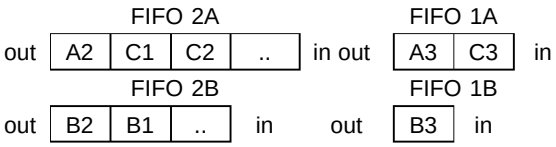
Note: The FIFOs name used in this description can be referred to in figure 5.2.

- At the beginning of the process, we suppose that FIFO 2A (upper and lower bands FIFO) already contains A1 A2 C1 and C2 and that FIFO 2B (middle band) contains B1 and B2. FIFOs 1A and 1B should contain at the beginning respectively A3 C3 and B3. This can be represented by :



There is a free area in FIFO 2B which size is one block.

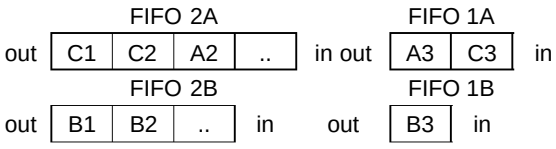
- The first phase is an initialisation phase of the STi3220 and consists in loading the first half part of the first sub search window into the chip. Blocks A1 and B1 are read out from FIFOs 2A and 2B. Block A1 will not be used in another phase so FIFO 2A is not written during that time. On the other hand, block B1 will be used later and hence the block read out from the FIFO 2B is written back into the same FIFO. The reference block input X on the STi3220 is irrelevant during that phase. The FIFOs content at the end of this phase is:



- During the second phase A2 and B2 are output from the FIFOs. This is a block sequence for the STi3220 during which the reference block must also be input. At the end of that phase the first

sub search window is input and the first motion vector computation is running. As blocks A2 and B2 will be used during later phases they must be written back in both FIFOS.

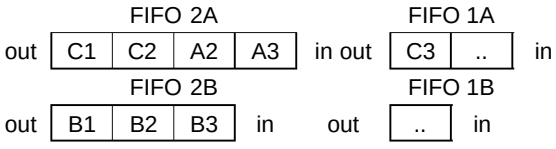
FIFOS content at the end of the phase :



the 3rd phase is also a block sequence for the STi3220 as the pipeline mode can be used between first sub-window and second one. The reference block must again be sent to the STi3220. During the same time blocks A3 and B3 are needed to finish the second search window: as those blocks are not yet into FIFOS 2A and 2B, they are read out from FIFOs 1A and 1B while they are also stored into FIFOs 2A and 2B in their free locations i.e. just after blocks A2 and B2. FIFOs 2A and 2B are not read during that phase.

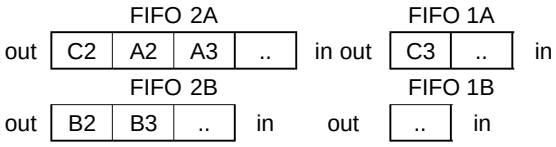
Note: on the 46th to 49th cycle of this sequence the first motion vector and minimum distortion are delivered on IOB bus of the STi3220 : they must be stored for future exploitation.

The FIFOs content at the end of the phase is:



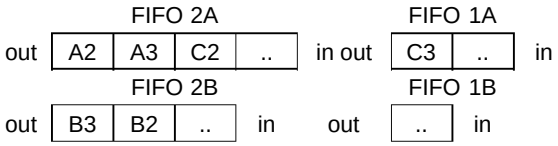
FIFO 1A has free location for writing the next search window block for the next process, i.e. A4. As this FIFO contains two blocks for search window, writing A4 can spend half an input block time (twice the pixel rate). In the same way B4 can be written into FIFO 1B during the time of one input block (at pixel rate).

- The 4th phase is again an initialisation sequence for the STi3220 during which the first part of the third sub search window must be loaded. During that phase blocks C1 and B1 are read out from FIFOs 2A and 2B. Both blocks will not be used anymore and hence are not written back into the FIFOs.

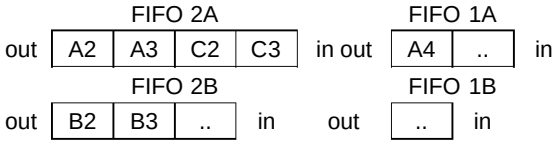


Note: The result of the second sub search window is available during that phase and must be stored.

- During the 5th phase, which is an STI3220's block sequence, blocks C2 and B2 are output from FIFOs 2A and 2B and must be written back into their original FIFO for future usage. Of course the same reference block is again input into the STI3220.



- The 6th is the last block sequence, pipelined with the previous one, during which the last part of the search window is loaded : block C3 is delivered by FIFO 1A and stored into 2A while block B3 is delivered by FIFO 2B.



At the end of that phase, FIFOs 2A and 2B are ready for starting the next reference block computation. The hypothesis of the FIFOs content at the beginning of the process is verified. For FIFOs 1A and 1B, we supposed at the beginning of the explanation that the blocks needed where already stored into the FIFOs. In fact blocks An and Bn are needed on the 3rd phase of the process, blocks Cn on the last phase. They can be loaded while they are not used and not necessarely since the beginning of a search window process.

Note : the result of the last sub search window is obtained after the 46th cycle of the first phase of the next search window delivery.

Note : The reference block must also be sent 4 times to the STi3220 chip. For that purpose the reference block is first sent to the board at 6f frequency and stored into a FIFO which output is sent back to its own input in the same way than the other FIFOs used for search window delivery.

V-5 Frame buffer accesses

The frame size is 352x288x1.5 = 152,064 bytes organised as 396 macro blocks each of 398 bytes. For serial port's use optimisation, each macro block is associated to one row of the memory. Only 396 rows over 512 and 398 columns over 512 are used in each of the two 256K x 4 video RAM of the frame buffer.

The serial port of the video RAM is dedicated to search window blocks reading and to block reconstruction : the blocks needed for search window are only Y component blocks (256 bytes) while the reconstructed block is a complete macro block (384 bytes).

When reconstructing a block into the frame memory, care must be taken in order not to scratch information still necessary for the search window (refer to example 1 for explanations): the delay before writing a new block into the frame memory after it has been computed in the STi3220 must be at least equal to 2 lines of blocks and 2 blocks = 46 macro blocks = 17664 cycles. This would be a lot of bytes to temporarely store in an additionnal memory. To avoid this, it is possible to write the reconstructed blocks directly into a free area of the frame memory (512 - 396 = 116 rows available) and to transfer after the desired delay, the macro blocks from the temporal additionnal area to their definitive location in the frame memory: this is one transfer from additionnal area in RAM to SAM and one transfer from SAM to frame area in RAM.

As the macro blocks are computed in the decoder in 8x8 size (in the order Y1 Y2 Y3 Y4 U and V as explained in chapter 4.1) it is necessary when writing the blocks into the frame memory to store them in the good order for a future macro block sequential output: first column of Y1 block must be followed by first column of Y3, followed by second column of Y1 .... That means that only 8 consecutive samples coming from the decoder can be written on consecutive addresses of the serial port. A transfer between SAM and RAM must be done every 8 pixels (except for U and V that are always 8x8 blocks stored after the last column of Y4). Writing a macro into the frame memory will need 32 transfers during Y samples block input and one final transfer after U and V blocks input.

The total number of accesses on the serial port during one block of pixels is:

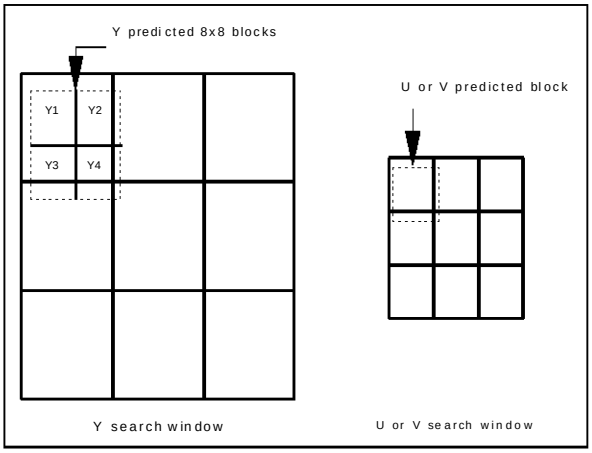
- read of 3 blocks of 256 bytes for search window. The maximum time for loading a block into serial memory consists in waiting for the current random access to be finished, followed by the transfer cycle from RAM to SAM. For 3 blocks this will be a total of 6 random cycles.
- write of the reconstructed macro block i.e. 384 serial cycles associated with 33 transfers and 33 ”wait” cycles.
- 2 transfers between RAM and SAM for storage of the macro blocks in the good final address, associated to one ”wait” cycle.

This is a total of 3x256 + 384 serial cycles + 6 + 66 + 3 random cycles. With a 100ns video RAM the minimum serial clock cycle is 30ns and the random one is 190ns leading to a global time of 48.810 us.

As the basic pixel frequency is 352 x 288 x 30Hz = 3.04128MHz = f, the total time allowed per block is 256/f = 84.175µs: the margin is high enough for the serial port accesses.

On the other hand, the random port is dedicated to the predicted macro block accesses.The predicted block must be delivered to the coder in a 8x8 form (Y1 Y2 Y3 Y4 U and V) (see Figure 23).

Figure 5.4 : Example of predictor position in the searc window



As it can be seen in the figure, for two 8x8 predicted blocks (Y1 and Y2 in the example above) it is possible to read 8 consecutive samples of a column on 8 consecutive addresses of a macro block and for two 8x8 predicted blocks (Y3 and Y4 in the example above) two different macro blocks access are necessary for delivering one column of 8 samples. In the same way for the U and V predicted blocks it is necessary to do two macro block changes every 8 samples. Each macro block change is a normal random cycle (190ns) while each consecutive address access can be done as a page mode cycle (55ns).

The total number of accesses for prediction is : 2 x (8 x (1 random + 7 page)) + 2 x (8 x (2 random + 6 page)) + 2 x (8 x (2 random + 6 page)) = 80 random + 304 page mode.

Those accesses will be interrupted by the control cycles for the serial port, i.e. 3 + 33 + 2 = 38 cycles (equivalent to a random access). As those accesses may happen during page mode sequences,

we consider that each access for the serial port replaces a page mode cycle by a random cycle.

The global number of accesses on the random port is:

(80 + 38) random + 38 transfers + (304 - 38) page mode = 156 random + 266 page mode = 44.27µs.

This time is also compatible with the total time of 84.175µs allowed for a complete macro block input.

If we take a basic cycle time of 50ns for the serial port, with a page mode cycle of 100ns and a random cycle of 300ns, the total times on each port become:

- serial port : 1152 serial + 75 random = 80.1µs
- random port : 156 random + 266 page = 73.4µs

The time margin on the frame memory accesses may allow to use slower (and hence cheaper) video RAM, or free spaces in the vidoe RAM can be used for additional functions that have not been considered here, for example the line to block conversion for the reference block. The more functions will share the frame memory space the more complicated will be the address controller. However this controller is most likely to be realised with EPLD devices or integrated into an ASIC.

V-6 Conclusion

This architecture presents the advantage of offering interesting solutions for computing a search window range that didn't seem originally easy to manage:

- Use of only one STi3220 chip running full speed for calculation of the whole range.
- Use of a small amount of video RAM for frame buffer (two 256Kx4 chips) shared between frame reconstruction, prediction and serach window delivery.
- Use of 5 additional FIFOs connected in a clever way, for delivery of the good search window sequences on the STi3220 inputs.

However, having said that, it must be considered that the RAMs, FIFOs and multiplexer controller is going to be very complex. An ASIC circuit will be the good choice in that case.

VI - MISCELLANEOUS

VI-1 -16/+15 displacement range with one chip at lower speed

In the third example, we saw that the STi3220 chip was used at 6f frequency if f is the pixel rate, due to the fact that two intialisation sequences are necessary for each search window scanning. The pipeline mode cannot be used efficiently in this case.

However it is possible to always use the pipeline mode as shown in Figure 24.

All the blocks of a complete row are first compared to their associated sub-search window 1 for instance (upper left sub search window): all the corresponding partial vectors and distortions must be stored. Then the same reference blocks are sent again to the STi3220 chip for comparison with sub search window 2 (upper right): the resulting distortions are compared to their corresponding distortions of sub search window 1 and the minimum of both is kept in memory with its associated motion vector. Of course the same operation is made again with sub-search windows 3 (lower left) and 4 (lower right). During all the computation the pipeline mode is never broken.

In this case the chip's working frequency is only four times the pixel's rate.

After a complete line of blocks has been computed, all the resulting motion vectors can be used for extraction of the predicted blocks.

The disadvantage of this solution is that it is necessary to work over a complete line of blocks (address generation more complicated) and that, consequently it needs the storage of all the minimum distortions and motion vectors of a line of block.

This solution could be useful for applications where the STi3220 working frequency is not high enough (chip running at 4f instead of 6f).

VI-2 Computing larger displacement ranges

There is no limitation on the size of the search window that can be used : the only condition is to split the global search window into sub-search windows compatible with the STi3220 chip search window (-8 to +7). Figure 25 shows an example of -32 to + 31 search window scanning in horizontal direction and -16 to +15 scanning in vertical direction with 16x16 blocks.

The larger the total search window, the higher the number of sub-search windows and the faster the chip's working frequency. In the example of Figure 25 the chip must work at 10 times the pixel's frequency. If the chip frequency becomes too high it is then necessary to use several chips as explained in chapter VI.3.

Figure 24 : -16/+15 Displacement Range with STI3220 at 4f Frequency

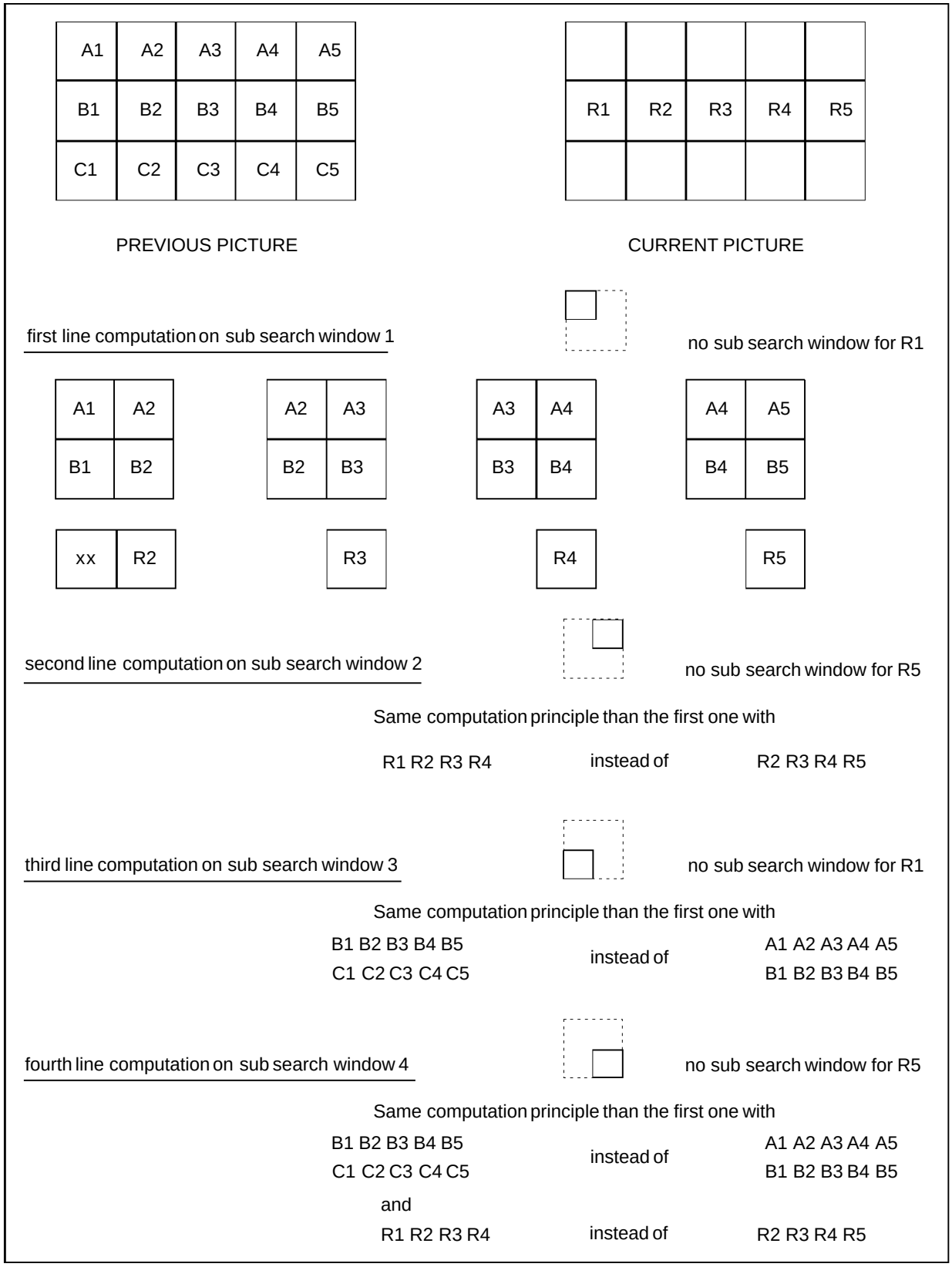


Figure 25 : Example of Search Window Range -32/+31 in Horizontal Direction -16/+15 In Vertical Direction

A1	A2	A3	A4	A5
B1	B2	B3	B4	B5
C1	C2	C3	C4	C5

search window of reference block centered on B3 position

sub search window 1	A1 A2	sub search window 5	B1 B2
1 initialisation + 1 block sequence	B1 B2	1 initialisation + 1 block sequence	C1 C2
sub search window 2	a2 A3	sub search window 6	b2 B3
1 block sequence	b2 B3	1 block sequence	c2 C3
sub search window 3	a3 A4	sub search window 7	b3 B4
1 block sequence	b3 B4	1 block sequence	c3 C4
sub search window 4	a4 A5	sub search window 8	b4 B5
1 block sequence	b4 B5	1 block sequence	c4 C5

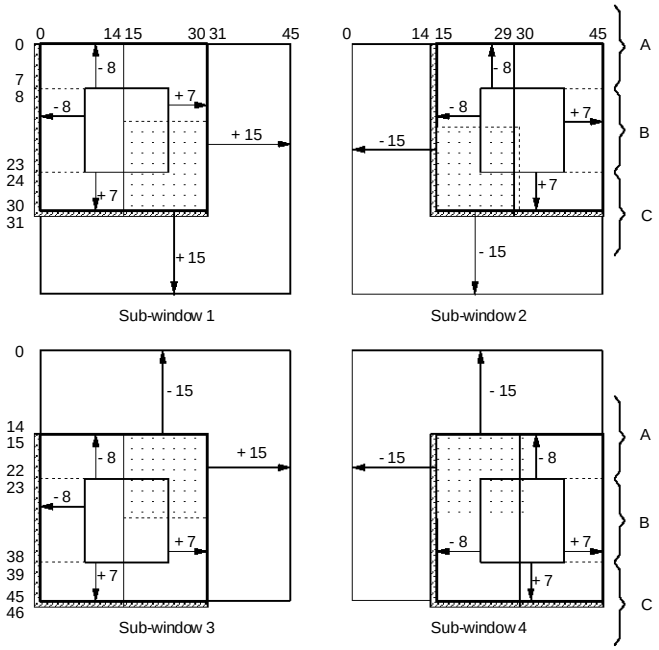
An, Bn or Cn : blocks sent to the STi 3220

an, bn, cn : blocks already in the STi 3220

Total time for scanning all the search window = 10 blocks

AN411-24.EPS

Figure 26 : -15/+15 Search Window



A411-25.EPS

The search window size is not necessarely defined between -(m x block_size) to (m x block_size)- 1 as all the examples that have been seen before. But if it is not, then the pipeline mode cannot be used efficiently on a line of blocks. The main typical case concerns the 16x16 blocks with -15 to +15 search window range. As it can be seen on Figure 26, the initialisation sequence of sub-search window 2 (or 4) is not equal to the blocks sequence of sub-search window 1 (or 3) but is shifted one column left: therefore the pipeline mode cannot be used between search windows 1 (3) and 2 (4).

The time needed to scan the complete search window is equivalent to 8 sequences (4 initialisations and 4 blocks) instead of the 6 sequences (2 intialisations and 4 blocks).

If the STi3220 motion estimation chip is not able to support the pixel rate implied by such an application, an alternative could be to compute the -16/+15 search window and to clamp the motion vectors that may be equal to -16 to the value -15: this is not critical as in that case the prediction is always made but perhaps with less efficiency.

VI-3 Using several STi3220

If the STi3220 working frequency is not high enough for the desired search window and pixels' frequency, it becomes necessary to use several chips for the computations. Two solutions are then possible:

- Each chip computes one or several sub search windows of each reference block.
- Each chip computes the complete search window but only for one reference block every N reference blocks (where N is the number of chips) as illustrated in the figure 6.4 here under with two STi3220.

In that example, it can be noticed that the frame buffer RAM working frequency is exactly the same as the STi3220 chips working frequency. Of course a small additional RAM is necessary around each chip for temporal storage of its search window. The motion vector is alternatively delivered by one chip or the other.

Figure 27 : Use of Two STi3220