Highly Scalable Parallel Parametrizable Architecture of the Motion Estimator

R. Cmar

Dpt. Microelectronics
FEI STU Bratislava, Slovakia

S. Vernalde

IMEC-VSDM
Kapeldreef 75, 3001 Leuven, Belgium

Abstract

In this paper a parametrizable architecture of a motion estimator (ME) is presented. The ME is designed as a generic full pixel calculation module which can be adopted for different video standards. The parameters by which the ME is described allow for a variety of architecture implementations. The parameters specify the level of parallelism reflected by multiple allocation of computational resources, and the use of configurable cache memories. The obtained VHDL description of the ME module is well suited for VLSI implementation.

1 Introduction

To achieve high compression rate for the coding of video sequences, the coding algorithm must be based on the exploitation of temporal redundancies, i.e. utilizing the information from previous frames. This is addressed by the motion estimation (ME) algorithm.

The motion estimation for hardware implementation is preferably realized using the full search block matching algorithm (FBMA). This is because of the inherent regularity that supports parallelism which is a basic technique to substantially increase throughput. The computational requirements of FBMA are very high especially when investigating larger search areas in a previous frame. The ME module is thus the key component in video encoder applications. Some relevant architectures of the ME [1-4] and video encoders [5] have been presented.

High speed requirements necessitate the implementation of critical parts of coding algorithms by a hardware accelerator. The parametrizable architecture of the accelerator is the solution to high performance combined with flexibility. An important issue is the memory since it drastically influences the power consumption of an application and has a major impact on the performance. In order to reduce the memory allocation and increase the memory bandwidth a dedicated and distributed memory is needed.

This paper will focus on the hardware implementation of the general full pixel motion estimator. In contrast to the architectures [1-4] we will concentrate on the flexibility of the scheme to cover a large application space while putting a special emphasis on the memory optimization. The architecture is parametrizable and particular instances are generated by chosing a specific parameter set. The parameter set determines the degree of parallelism and infers optimized cache memory configuration. The possible parallelism spans requirements for the processing performance of the H.263 standard up to HDTV.

The advantage of this approach is that a) it adapts the architecture to the speed requirements through the setting of the parameters; b) the generated VHDL high-level description of the ME is technology independent; c) it is extendible to the implementation of different video coding standards.

2 Architecture

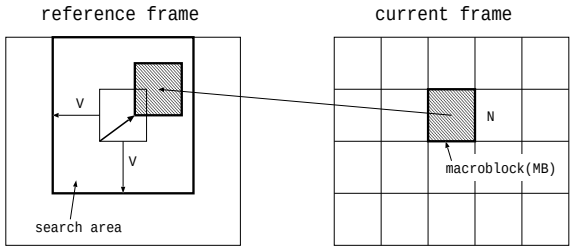


Figure 1: FBMA

The FBMA is based on the evaluation of the absolute difference error between a macroblock (MB) in the current picture frame and macroblocks in a search area in the reference picture frame (Figure 1). The error evaluation can be expressed as:

$$MAD(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |x(i,j) - y(i+m, j+n)| \qquad (1)$$

ED&TC '97 on CD-ROM

where x and y are the pixel values of a current and a reference picture frame, N is the macroblock size. The m, n coordinates with the minimum MAD value yields the motion vector (MV) of the encoded macroblock. The displacement calculation is restricted to a search range V such that $-V \leq m, n \leq V - 1$.

2.1 Macroblock parallelism

To access two frame memories directly would cause a processing limitation, therefore intermediate cache memories for the current (*CW cache memory*) and the reference (*SW cache memory*) frames are needed. Moreover we will try to reduce the size of the search window memory since a SW cache memory can be very large when dealing with possible motion vectors up to 64 pixels. The objective is to achieve this without imposing limitations on the processing speed.

In Figure 2a the search areas of the MBs are divided into vertical strips. Each MB's search area contains $S = (2V + N)/B$ strips. Let's allocate $Q + 1$ strips for the SW cache memory where $Q + 1 < S$. Generally the minimum number of required strips (Q+1) is given by $Q = N/B + 1$, while the strip size is given by $B * (N + 2 * V - 1)$. B is the architectural parameter which specifies the width of the memory strip. Partitioning of the SW cache memory into strips allows the *parallelism for the MAD calculation and the memory loading processes.*

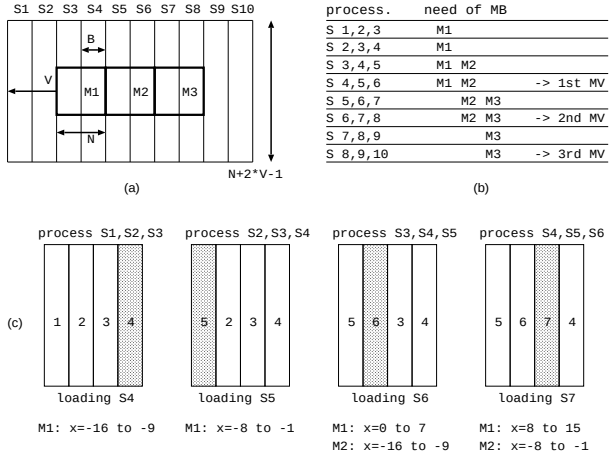


Figure 2: Macroblock parallelism, example with N=16, V=16, B=8, Q=3, P=2

In the example shown in Figure 2c Q strips $(S1, S2, S3)$ are processed by *the full pixel calculation estimator module* (F-estimator), while 1 extra strip $(S4)$ is loaded from the frame memory. In the next processing step $Q - 1$ strips $(S2, S3)$ of the previous memory content can be reused while, note, one additional strip $(S4)$ has been loaded. This processing step thus performs the calculation with the strips $(S2, S3, S4)$ while loading the strip $S5$ at the position of $S1$, etc. The processing of Q strips means a MAD evaluation of all possible MV displacements (henceforth only displacements) for the actual SW cache memory content (so called *strip processing*).

The reduction of the SW cache memory causes that at each processing step several successive MBs must be available in the CW cache memory, except for the beginning and the end of the row. Figure 2b shows that two CW cache memory units (1 unit = 1 MB) are needed to store two successive macroblocks. For example, at the step where $S3, S4, S5$ are processed, the same SW cache memory content is used (i.e. the same pixel usage) for the MAD calculation of displacements for different x-slices for the macroblocks M1 and M2 (shown in Fig. 2c). The number of multiple allocation of the CW cache memory units (P) required is given by $2 * V/N$. It means that the MAD calculation for the same SW cache memory content is performed for P successive macroblocks stored in CW cache memory at the same time (*macroblock-parallelism*). This means a high utilization of the SW cache memory resulting in a reduced number of memory accesses.

Finally by allocating one more CW cache unit we can take advantage of the preloading mechanism to avoid encountering wait states for the loading of the CW cache memories at transitions. A special mapping is needed to correctly distribute P+1 data supplied by the CW cache memories among P processing units inside of the F-estimator module. The remaining data can be used by the half pixel calculation if provisioned.

In the example (Figure 2b) the MAD calculation for one MB is spread over 4 processing steps while in each step different x-slice displacements are evaluated. The final MV for the MB is issued after executing the last processing step. The lifetimes for the calculation of MVs for successive MBs overlap and the MVs are generated in a pipelined fashion.

2.2 Strip parallelism

The strip parallelism is built on the macroblock parallelism. It can be illustrated on the snapshot of macroblock parallelism (strip processing from Figure 2c) considering one CW cache memory unit and the SW cache memory (Figure 3). The goal is to investigate all displacements of a MB stored in the CW cache memory within a portion of the search area represented by the SW cache memory content.

Several passes through the CW and SW cache memories are necessary. The simple solution would be to

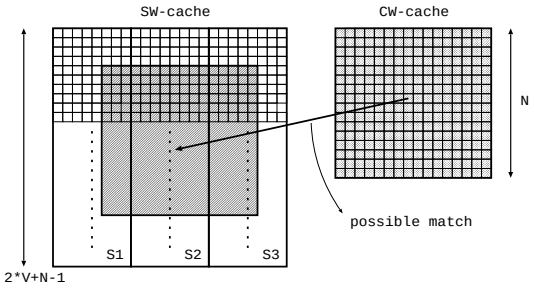


Figure 3: Strip parallelism

investigate one displacement per pass. However the strip parallelism suggests to investigate several displacements. Having considered the sizes of the CW and SW cache memories at the x-direction, N and $N+B$ respectively, the parallelism of size B can be chosen. In other words the row of $N+B-1$ pixels of the SW cache memory is processed against the row of N pixels of the CW cache memory, thus B displacements per MB can be investigated in one pass. The systolic array structures are suitable to implement this kind of parallelism (Figure 4).

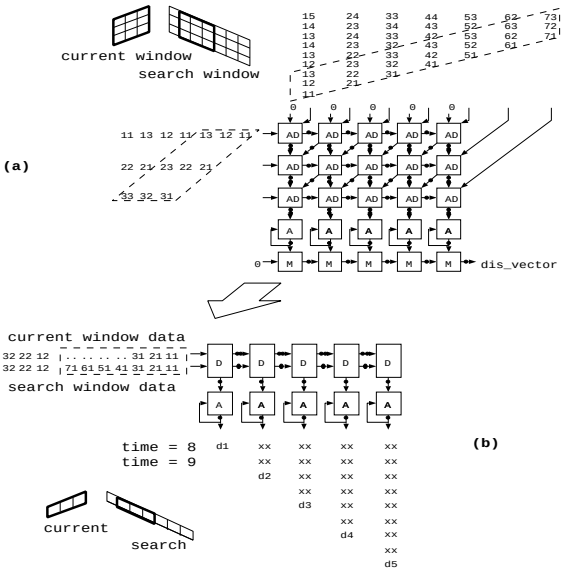


Figure 4: Systolic array implementations

Using the 3D dataflow representation of FBMA, several systolic array architectures were developed by [1]. In our case the systolic array architecture which performs the displacement calculation in the x-direction is needed. Generally this is to perform FBMA of the MB ($N*N$) within the search area of the size *N * (N+B-1)*. The corresponding structure ($3*3$ within $3*7$) is shown in Figure 4a. The preliminary synthesis results showed this is a highly area consuming structure when using realistic parameters. By another projection of 'matrix-shape' systolic array structure the 'chain-shape' structure can be obtained (Figure 4b), here for the particular case ($1*3$ within $1*7$). This 'chain-shape' systolic array processing unit (SAP) is the basic functional unit in the design.

The advantage of this line-against-line approach is that the input to the array structure is serial thus posing traditional requirements on a memory access. Further on this array structure allows for possible stacking of SAP elements (supports scalability) thus achieving higher processing performance tailored to the specific needs. Finally the systolicity of the output is also of great advantage as will be shown later.

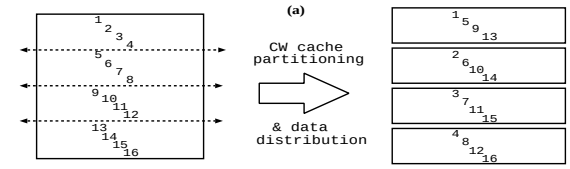
2.3 Intrablock parallelism

This parallelism in the architecture is based on stacking of several SAP elements which operate on different parts of an image, i.e. within MB. This is because of the commutativity of the FBMA calculation where the investigation of a displacement means the sum of the pixel differences. This way we may divide the MB area into several equal parts while each of these parts is processed by a unique SAP element. If the division is defined by R, any displacement calculation is accelerated by the factor R, too. At the end the partial error sums are summed up to get a MV evaluation. To make it possible we need to have multiple access to the CW and SW cache memories to feed R SAP units by two sets of R data sets.

Now we have to consider a distribution of data from the CW and SW cache memories for multiple passes through these memories in order to evaluate y-direction displacements, i.e. N rows of MB against the search area of the size $2*V+N-1$ rows (Fig. 3). This means that at the first pass the rows 1 to N (1..N) of the SW cache memory are compared with the CW cache memory, in the second pass the rows 2..N+1, in the third one 3..N+2, etc. As an example, if R = 4, we allocate 4 SAP units while each SAP must process N/4 rows of both memories in each pass. The obvious requirement is that these SAPs must work in parallel.

Since each SAP accesses different data from memory and the requirement is to have a single memory access, the memory must logically be divided into 4 (R) memory submodules. To reach efficient parallelism we have to ensure that at any time each SAP fetches the data from a different memory submodule. The solution is to provide the special data distribution (needs special loading mechanism) for both cache memories according to Fig. 5a (the numbers inside of the memories indicate the original pixel row position).

Previously we assigned the calculation of B displacements in the row manner to the SAP unit. So generally spoken having R SAP units causes acceleration of the MB pass by factor R while investigating B displacements in one pass.



(b)

	step	SAP(1) CW - SW	SAP(2) CW - SW	SAP(3) CW - SW	SAP(4) CW - SW
y-vector 0	1	1(1) - 1(1)	2(2) - 2(2)	3(3) - 3(3)	4(4) - 4(4)
	2	5(1) - 5(1)	6(2) - 6(2)	7(3) - 7(3)	8(4) - 8(4)
	3	9(1) - 9(1)	10(2) - 10(2)	11(3) - 11(3)	12(4) - 12(4)
	4	13(1) - 13(1)	14(2) - 14(2)	15(3) - 15(3)	16(4) - 16(4)
y-vector 1	1	1(1) - 2(2)	2(2) - 3(3)	3(3) - 4(4)	4(4) - 5(1)
	2	5(1) - 6(2)	6(2) - 7(3)	7(3) - 8(4)	8(4) - 9(1)
	3	9(1) - 10(2)	10(2) - 11(3)	11(3) - 12(4)	12(4) - 13(1)
	4	13(1) - 14(2)	14(2) - 15(3)	15(3) - 16(4)	16(4) - 17(1)
y-vector 2	1	1(1) - 3(3)	2(2) - 4(4)	3(3) - 5(1)	4(4) - 6(2)
	2	5(1) - 7(3)	6(2) - 8(4)	7(3) - 9(1)	8(4) - 10(2)
	3	9(1) - 11(3)	10(2) - 12(4)	11(3) - 13(1)	12(4) - 14(2)
	4	13(1) - 15(3)	14(2) - 16(4)	15(3) - 17(1)	16(4) - 18(2)
⋮	⋮	⋮	⋮	⋮	⋮

Figure 5: Memory partitioning and data distribution

Figure 5b shows the calculation of different y-direction displacements and corresponding requirements of memory row distribution for SAP units. The row position corresponds to a different step of the SAP processing and the column position represents the processing of the particular SAP unit. The important issue is that in each row of the table the different memory submodules (the numbers in brackets) are accessed by each SAP which allows the efficient usage of single port memories only.

2.4 Parallelism in the design

The product of the degree of the *macroblock, strip* and *intrablock parallelism* $P * B * R$ represents the overall architectural parallelism. The degree of parallelism is proportional to the amount of allocated SAP units. The allocation of the SAP calculation units (that is a major area contribution) can be reduced using the hierarchical mode (D parameter), that is by a factor D. Instead of evaluating B displacements per MB within the SAP units simultaneously, using the so called x-direction interleaving, only B/D displacements are calculated, thus reducing the SAP area by almost the factor D. The interleaving is also performed in the y-direction by handling the row counter. D = 1 results in an exhaustive full pixel search. D = 2 evaluates the full MV with the precision given by 2 pixels, etc. The coarse speed-up can thus be characterized by $B * P * R * D$.

2.5 Global architecture

The main parameters which describe an architecture are the following ones.

- N macroblock size
- V maximum motion vector
- B width of the search memory strip
- D step of the hierarchical search
- R intrablock parallelism

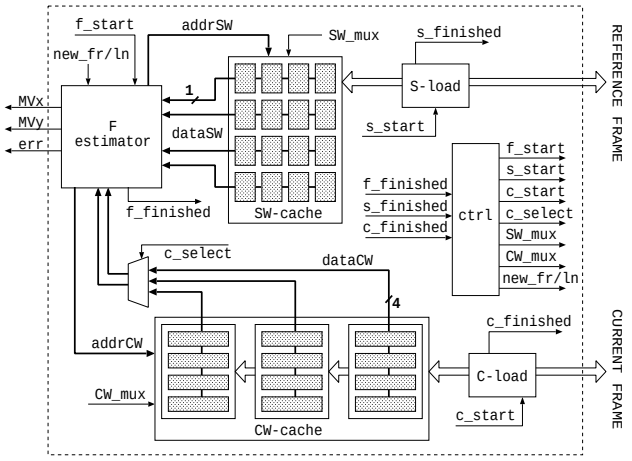


Figure 6: Architecture example

The architecture example for the parameter set N = 8, V = 8, B = 4, R = 4 is shown in Figure 6. The F-estimator module is designed such that it generates the addresses for a pixel fetch from the cache memories. In the next cycle the data are expected to be ready at the data input ports. The loading of the cache memories is provided by data transfer modules (S-loader, C-loader). The processing of the F-estimator is synchronized with these cache loading processes by the controller which fires these processing modules and steers some additional signals. It makes sure the memories are filled with the proper data and keeps the order of the processing. The architecture requires single port memories only.

A more detailed look at the F-estimator module is shown in Figure 7. The DISTR module takes care of the reading of the data from the cache memories and distributing them among the SAP units. The SAPS module consisting of multiple SAP units performs the FBMA calculation in a way that each SAP produces B/D partial error sums generated at different time steps. The ACCBUFF module sums and stores these partial error sums and finally evaluates MV. Thanks to the systolicity of the output from the SAP units the ACCBUFF module exhibits the high degree of

N	V	B	R	D	Parallelism (B*P*R)	Frame size	Frame Rate /sec	Area (mm2)	SAP area (%)
16	16	8	4	1	64	CCIR	5.1	18.5	81
16	16	8	16	1	256	CCIR	20.5	65.0	87
16	16	16	16	1	512	CCIR	30.4	118.3	92
16	16	16	16	1	512	HDTV	5.2	118.3	92

Table 1: Performance analysis and synthesis results (obtained by Synopsys)

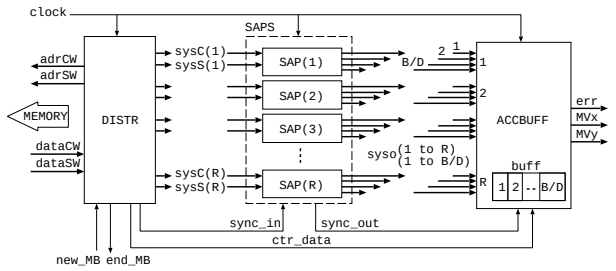


Figure 7: F-estimator module

resource sharing. Generally there are P allocations of the SAPS and the ACCBUFF modules while the DISTR module is shared.

3 Results

The processing performance evaluation is dependent on the parameter set. Assuming the size of a frame is $I * J$ macroblocks, the number of strip processings N_{sp} within one row of MBs is $N_{sp} = N/B * (J-1) + V/B$. The number of cycles for a strip processing can be expressed as $C_{sp} = N_{pass} * C_{mb}$ where N_{pass} is the number of passes within a strip processing defined by $N_{pass} = 2V$ and C_{mb} is the number of cycles for one pass through the macroblock defined by $C_{mb} = (N+B-1)*N/R$. Finally the number of cycles for the processing of the whole frame is expressed by $N_{sp} * C_{sp} * I$.

A customized ME architecture is generated according to the specified set of parameters. The VHDL generic constructs allow to describe the configurable architecture. Thus the input parameters decide upon the number of resource allocations and are also responsible for a functional modification of some resources.

The table shows synthesis results (the F-estimator module only, the interconnection area considered to be 0) of some architectural implementations along with the throughput. The results were obtained using MIETEC technology 5V, 0.7um, assuming a ME processing clock cycle of 20 ns (50 MHz). The percentage of the SAP units' area in the design shows the very high hardware resource utilization since the auxiliary circuitry area is minimal.

4 Conclusion

A parametrizable scalable parallel architecture for a motion estimator was developed which allows for a variety of configurations. The architectural parametrizability of the module leads to an efficient customized implementation.

The parameter settings along with the hardware specifications (memory access time, processing clock cycle) determine the speed and the area characteristics of the ME module. Different speed requirements are satisfied architecturally through the choice of the degree of parallelism and are tuned further by high-level synthesis tools. The hardware optimization is possible thanks to the high level description which is technology independent.

The key issue in this approach is an optimization of the ME module taking into account its memory organization. The efficient cache memory strategy considerably relaxes frame memory bandwidth limitations, thus makes the ME module suitable for different video encoder architectures.

References

[1] T.Komarek, P.Pirsch, "Array Architectures for Block Matching Algorithms", IEEE Trans. on Circ. and Systems, Vol.36, No.10, October 1989

[2] K.M.Yang, M.T.Sun, L.Wu, "Family of VLSI Designs for the Motion Compensation Block-Matching Algorithm", IEEE Trans. on Circ. and Systems, Vol.36, No.10, Oct. 1989

[3] L.D.Vos, M.Stegherr, "Parametrizable VLSI architectures for the Full-Search Block-Matching Algorithm", IEEE Trans. on Circ. and Systems, Vol.36, No.10, October 1989

[4] Y.S.Jehng, L.G.Chen, T.D.Chiueh, "An Efficient and Simple VLSI Tree Architecture for Motion Estimator Algorithms", IEEE Trans. on Sig.Proc., Vol.41, No.2, Feb. 1993

[5] P.Pirsch, N. Demassieux, W.Gehrke, "VLSI Architectures for Video Compression", Proc. of the IEEE, Vol. 83, No. 2, February 1995