DATA-REUSE EXPLORATION OF MULTIMEDIA APPLICATIONS ON PROGRAMMABLE PROCESSOR CORES[1]

A. Argyriou, M. Dasigenis, K. Tatas, N. Zervas*, and D. Soudris

VLSI Design and Testing Center, Dept. of Electrical and Computer Eng.
Democritus University of Thrace, 67 100 Xanthi, Greece

*VLSI Lab Dept. of Electrical and Computer Eng.
University of Patras, 26500, Patras, Greece

Abstract

The effect of the data-reuse transformations on the power dissipation but also on area and performance of multimedia applications implemented on single and multiple embedded programmable processor cores, is explored. The considered target architecture consists of, in the general case, multiple processor cores each of which has its own instruction memory and data memory hierarchy. Before coping with data-reuse exploration problem, we have to map a given multimedia application on the multiprocessor environment. For that purpose, the multimedia application is partitioned by employing LSGP partitioning scheme. The proposed methodology is illustrated through a widely-used algorithmic kernel named full search motion estimation. The plethora of experimental results show that there exist close relation among the number of processor cores, and the certain data-reuse transformation from one hand, the power, the area and the performance from the other hand. Significant improvements in terms of data memory power consumption and performance. Furthermore, it is proved that the designer for the total power budget should take into account not only the power consumption related with the data memory but also with the instruction memory.

1. Introduction

The number of multimedia systems used for exchanging information is rapidly increasing nowadays. Portable multimedia applications, such as video phones, multimedia terminals and video cameras, are available. Power consumption has become the dominant factor during the design of such systems, not only due to portability, but as well as cooling and packaging. For this reason efficient power reduction techniques should be applied during the design of such a system [1].

High-level power optimizing strategies are the most promising ones for large gains in power consumption [1]. However, these applications require increased processing power for manipulating large amounts of data in real time. This demand can be met by the use either custom hardware architectures or a number of embedded programmable processors. The first approach may be power and area efficient but it lacks of flexibility, since only a specific algorithm can be executed. On the other hand, the use of programmable processors is less efficient for power and area than the previous approach but it allow us to implement multiple algorithms on the same target platform. Thus, it implements a broader class of applications, within reduced design cycle. In other words, such processor platforms allows to the designer meeting faster the tight time-to-market constraints.

In multimedia applications the dominant factor in power consumption is the one related to memory data transfers. This fact is true only for custom-processor platforms. In contrary, the programmable processors are characterized by one additional and critical power component related to the power consumption of memory fetches from instruction memory. However, the related work that combines partitioning of the algorithm and techniques for reducing memory transfers is relatively small [2],[3],[4],[5],[6]. More specifically, a systematic methodology for the reduction of memory power consumption was presented in [2][4]. However, the proposed methodology targets only custom hardware architectures and did not tackle with the problem when programmable processors are used. Another research

[1] This work was supported by the project PENED '99 funded by G.S.R.T of Greek Ministry of Development

work which makes the use of data-reuse transformations for reducing memory transfers is presented in [3]. However, this approach has as target architecture a single processor-based system. Both the two previous research works cannot apply on to a multiprocessor environment One partitioning approach attempting to improve memory utilization is presented in [6]. However, this approach is limited by the two-level memory hierarchy, while the class of algorithms expressed in Weak Single Assignment Code form [7].

In this paper, we perform an exhaustive exploration of data-reuse transformations in terms of power, area and performance for multimedia application, considering the programmable multiple processor core architecture called Distributed data memory. To execute a multimedia application we choose the LSGP partitioning scheme [8]. Since we assume programmable platforms, we explore the impact of transformations not only on data memory consumption, but also on instruction memory. As we will prove, the influence of the instruction power consumption on the total power budget is very critical.

2. Memory Architecture

We assume a data memory architecture with single or multiple processor cores each of which has its individual on-chip instruction memory. We name this scheme, application specific instruction memory (ASIM). The algorithm structure parameters such as the number of loop indices, number of iterations, and the format of a statement affects strongly the generated code size, and eventually, the instruction memory size of each processor. Depending on the running application, instruction memory size of the processors may be different. However, in the case of the motion-estimation-like algorithms [9], the code size is constant. Concerning the data-memory organization application specific data memory hierarchy (ASDMH) is assumed [2,10,3]. Since we focus on parallel processing architectures, we explore ASDMH in combination with a well-established, data-memory architecture model: called *distributed data-memory architecture* (DMA) with one and multiple programmable processor cores, *N*, with *N*=1,2,3,… Fig. 1 shows the target model DMA for multiple processors each of which has its separate data-memory hierarchy. Apparently, this model can be used a single processor. All memories modules of the memory hierarchy are single ported but also area overhead are possible in cases of large amount of common data to be processed by the *Í* processors. Thus because, in such cases duplicates of the same data may to exist at the same time in different data-memory hierarchies. The on-chip memories may have different sizes and connect directly to the corresponding processor. Since, we deal with motion-estimation-like algorithms, the off-chip memory consists of *2N* pairs of equal sized memories. Each pair includes data from the current and previous frame. The assumption that *#off_chip_memories=2N*, can be seen as a memory optimisation step. For experimental purposes, we considered target models with *N*=1, 2, and 3 without any restriction about memory hierarchy levels.

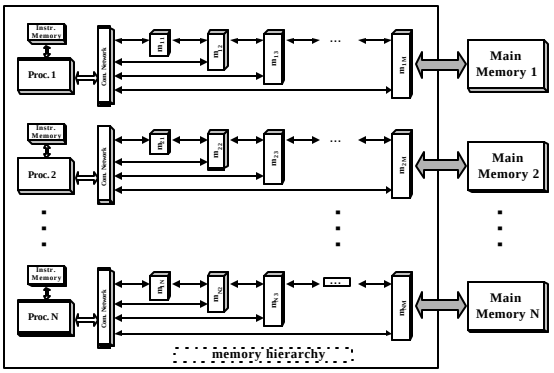

Figure 1. The distributed memory data-memory architecture model with multiple processor cores.

Apart from DMA, there exist two also well established models, namely shared data memory architecture (SMA) and shared-distributed data memory architecture (SDMA), which will be used for power exploration of data reuse transformations for future work.

3. Data Reuse Transformations

The fact that in multimedia applications the power related to memory transfers is the dominant factor in total power cost, motivate us to find an efficient method to reduce them. This goal can be done by efficient manipulation techniques of memory data transfers. For that purpose, we performed an exhaustive data reuse exploration of the application's data for different number of processors. Employing data reuse transformations, we determine the certain data sets, which are used very often in a short period of time. The re-used data can be stored in smaller on-chip memories, which require less power per access. In this way, redundant accesses to large off-chip memories are removed, resulting into a less power consumption. Of course, data reuse exploration has to decide which data sets are appropriate to be placed in separate memory. Otherwise, we will need a lot of different memories for each data set resulting into a significant area penalty. In the case of multiple cores, the data reuse transformations should be applied after the partitioning process on all processors resulting into a memory hierarchy closely related to the target architecture.

Since our target architecture consists of programmable processor(s), we should take into consideration the power dissipation due to instruction fetching. We will prove that the power component

related with the instruction memory is significant and should taken into account with the consumption coming from data memory transfers. Furthermore, the instruction-memory-related power depends on both number of executed instructions and the size of the application code. Particularly, the number of executed instructions determines how many times the instruction memory is accessed, while the code size determines the memory size. The cost function used for our data reuse exploration on the three target architectures is evaluated in terms of power, performance, and area. Power and area costs are related both on data memories and instruction memories. The cost function for power is given by:

$$Power\_cost = \sum_{i=1}^{N} power\_cost\_i \quad (1)$$

where N is the number of processors and the i-th power estimate, $power\_cost\_i$ is:

$$power\_cost\_i = \sum_{c \in CT} [P_r(word\_length(c), \#words, f_{read}(c)) + P_w(word\_length(c), \#words, f_{write}(c))] + P_i(instr\_word\_length, code\_size, f)$$

where c is a member of the copy tree (CT) [10], $P_r(\cdot)$, $P_w(\cdot)$, and $P_i(\cdot)$ is the power consumption estimate for read operation, write operation, and instruction fetch, respectively.

The total delay cost function is obtained by:

$$Delay\_cost = \max_i \{\#cycles\_processor_i\} \quad (2)$$

where $\#cycles\_processor_i$ denotes the number of the executed cycles of the i-th processor (i=1,2,...,N). Also, the maximum number of cycles specify the performance of the system. Here, for experimental reasons we use the ARMulator [11].

The corresponding area cost function is:

$$Area\_cost = \sum_{i=1}^{N} area\_cost\_i \quad (3)$$

with

$$area\_cost_i = \sum_{c \in CT} [area\_length(c), \#words(c)] + area(instr\_word\_length, code\_size).$$

4. High-Level Estimation Model

High level description of an application, e.g. algorithm-level, the design trajectory from the higher to lower design levels imposes making ccurate estimation (as much as possible) of certain design parameters such as power, area, and performance. In other words, we need a high-level estimation model to characterize an application through the calculation of critical factors. Below, we provide the model for power, area, and performance. As it was previously stated, in embedded multimedia applications the memory data transfers and storage dominate the total power consumption. However, we must also calculate the power related to instruction transfers, since our target models include programmable processors. Consequently, the power model should take into account the power components related to data and instruction memory accesses. These accesses take place either on-chip and off-chip for data memories and on-chip for instruction memories. Thus, our formula for one processor memory subsystem becomes:

$$P_{MEM} = \sum_i P_{ON-CHIP\_i} + \sum_j P_{OFF-CHIP\_j} + P_{I-MEM} \quad (5)$$

memories, respectively. The on-chip power is considered to be equal to the power related to the memory access itself by ignoring the power related to the on-chip bus activity because it is very small. For these calculations Landman's memory model is used [12]. According to this model on-chip power can be obtained by:

$$P_{ON-CHIP} = P_{ACCESS} = F_{ACCESS} \cdot F(word\_length, \#words, V_{dd}) \quad (6)$$

However, the off-chip power depends not only on the memory accesses itself but also in the wiring the bus drivers and I/O pins. Detailed calculation of these power components is very difficult. However, typical values can be set for the above factors in order to be able to make comparisons. Thus capacitance for bus drivers, I/O pins and wiring is assumed to be 24pF. It is also assumed that in average half of the bus (bit) lines make a transition per off chip memory access. Consequently, the formula for off-chip power calculation becomes:

$$P_{OFF-CHIP} = F_{ACCESS} \cdot [word\_length \cdot \frac{24 \cdot 10^{-12}}{2} \cdot V_{dd}^2) + F(word\_length, \#words, V_{dd})] \quad (7)$$

In order to estimate the performance of a particular application, we use the number of executed cycles resulting from the considered processor core simulation environment. Here, for experimental reasons we will use the ARMulator [11]. For the area occupied by the memories, Mulder's model is used [13].

5. Experimental Results-Comparative Study

We perform extensive comparative study of the relation between data-reuse transformations and different number of processors, taking into account the distributed architecture model. We begin with the description of our test vehicle and through its partitioning scheme, we will provide the experimental results after the application of the data-reuse transformations for all target architectures, in terms of power performance and area.

5.1 Demonstrator Application and Partitioning

Our demonstrator application was selected to be the full search motion estimation algorithm [14]. It was chosen this algorithm because it is used in a great number of image and video processing applications. Our experiments were carried out the luminance components

of QCIF frame (144x176) format. Reference window was selected to include 15x15 candidate blocks, while blocks of 16x16 pixels were considered. The algorithm structure is described in Figure 2. This algorithm has three double nested loops. A block of the current frame (outer loop) is compared to a number of candidate blocks (middle loop). In the inner loop, a distortion criterion is computed to perform the comparison.

```
for (x = 0; x < N /B; x++)
 for (y = 0; y < M/B}; y++)
  {
   for (i = -p; i < p+1; i++)
    for (j = -p; j < p+1; j++)
    {
     for (k = 0; k < B; k++)
      for (l = 0; l < B; l++)
       {
      if((B*x+i+k) < 0 || (B*x+i+k) > N-1 || (B*y+j+l) < 0 ||
     (B*y+j+l) > M-1)
\*conditional statement for the pixel of candidate block *\
```

Figure 2. The full search motion estimation algorithm.

In order to run full search motion estimation algorithm on multiple processor cores, we use as partitioning scheme the Locally-Sequential Globally-Parallel (LSGP) technique [8]. In the LSGP scheme, one partition is mapped to a processor, which executes sequentially the corresponding iterations of each partition. The number of processors is equal to the partitions, which perform the assigned iterations in parallel fashion. By applying this technique to a generalized for-loop structure, while assuming p partitions, the form of the partitioned algorithm becomes as shown in Figure 3.

Do in parallel:
Begin

for (x=0; $x < \lceil N/pB \rceil$; x++) {sub-algorithm}

for ($x < \lceil N/pB \rceil$; $x < \lceil 2N/pB \rceil$; x++) {sub-algorithm}

• • •

for ($x < \lceil (p-1)N/pB \rceil$; $x < \lceil N/B \rceil$; x++) { sub-algorithm}

End

Figure 3. The partitioned full search algorithm.

The semantic "*Do in parallel*" imposes the parallel (concurrent) execution of p nested loops (i.e. sub-algorithm). From this above-code, it is apparent that the outermost loop is broken into p partitions, each of which is mapped to processor. The p processors execute the same algorithmic structure for different values of loop index x, i.e. different current blocks. Due to the inherent property of algorithm, a set of data should be used by two consecutive sub-algorithms.

Here it should be stressed that the LSGP scheme partitions the number of iterations, but not the corresponding code being executed by each processor. In other words, the instruction memories have the same size and contents. Simply, the processors may execute different number of iterations (operations). Our experiments were carried out assuming p=2 and 3, meaning two and three partitions. Therefore, the loop index x has a range of nine. Due to QCIF format (144x176), the outermost index ranges from 0 to 8. The first, second, and third processor executes the algorithm in parallel fashion, for loop index x ranging from 0 to 2, from 3 to 5, and from 6 to 8, respectively.

We examined the impact of the different number of processors combined with 21 data reuse transformations [3] on power, performance, and area. In the case, of multiple cores the transformations were applied after the partitioning process was finished in accordance with the previous section. They involved the insertion of memories for a line of current blocks (CB line), a current block (CB), a line of candidate blocks (PB line), a candidate block (PB), a line of reference windows (RW line) and a reference window (RW). In Figure 4, the copy tree of the full search motion estimation algorithm is identical for processor 1, 2, and 3, where the dashed lines show the memory levels. Each rectangle contains three labels, where the number determines the applied data reuse transformations associated to memory hierarchy level. The remaining two labels determine the size of an PB and CB line or block, RW line or reference window.

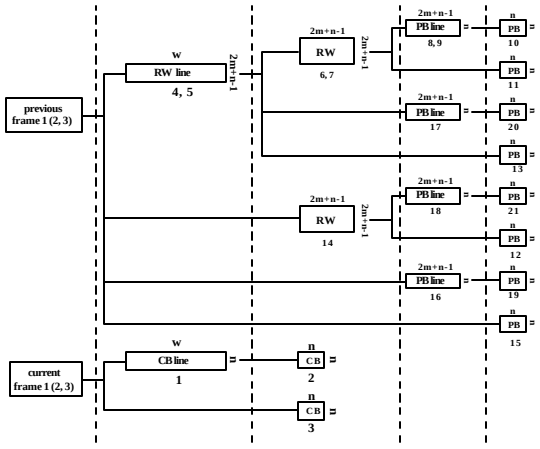

Figure 4. The copy tree structure of the full search estimation.

5.2 Experimental Results

Comparisons among the three processor configurations with application specific instruction memory in terms of data memory power consumption, instruction power consumption, performance and area in terms of data-reuse transformations are shown in Fig. (5),(6),(7),(8), and (9), respectively.

In Fig. 5 it can be seen that the data-reuse transformations have large effect on the power consumption of data memory for all processor configurations, where almost all transformations have identical effect. Also, the increased processor number leads to more power efficient designs. Although the increased number of processors reduce the power consumption of data memory, the rate of power reduction goes down, since the existence of multiple processors can be considered as a memory optimisation technique which has applied before a data reuse transformation. Although the multiple processors on-chip memory hierarchy remains unchanged, the number of memory accesses decrease, as the number of processors increase. In addition, the partition of off-chip memory is another power optimisation technique. In contrary, the increased number of processors increases the power consumption of instruction memory for several transformations, while the remaining ones (i.e. 1,2,3,4,5,14, and 15) do not affect the power consumption, as shown in Fig. 6. Comparing the absolute values of Fig. 5 and Fig. 6, it can easily be deduced that the power consumption of instruction memory has a very significant role and cannot be ignored in embedded programmable processor cores. The increased power consumption of instruction memory comes mainly from the increased code size, which depends on the number of processors, the *N*-tuple copies of code, and the increased control operations for the inter-processor connection.

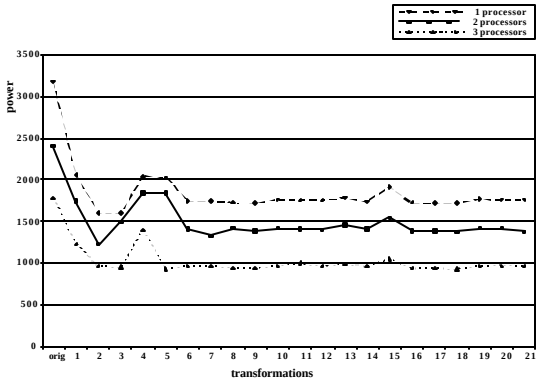

Figure 5.The effect of data-reuse transf. on power of data memory.

The dominant role of instruction memory power consumption should be decreased significantly. Otherwise, the data-reuse transformations are useless in embedded programmable processors. For that purpose, the use of appropriate instruction cache memory is compulsory. The issue of cache memory will be presented in future work.

Fig. 7 depicts the total power consumption. It can be seen that the most power efficient design approach is the combination of one and two processors and data-reuse transformations 2,3,14 and 16. Also, power efficient design is the combination of transformation 3 for all processor configurations. Studying carefully, Fig. 5,6, and 7, it can be easily inferred that total power consumption comes mainly from the power consumption of instruction memory, since the contribution of data memory is almost constant.

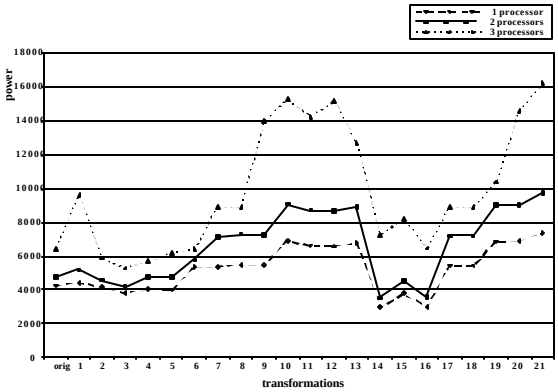

Figure 6. Comparison results for power dissipation of instruction memory.

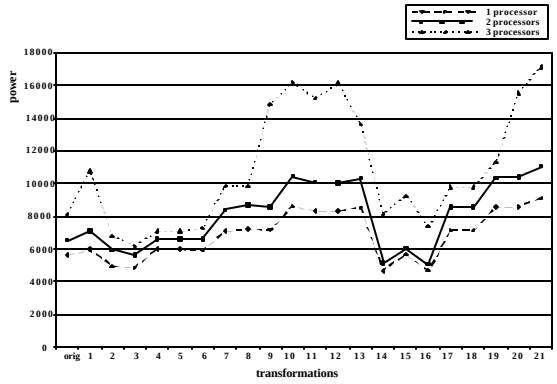

Figure 7. Comparison results for total power.

Fig. 8 shows that the configuration with three processors is the most efficient in terms of performance. Almost all data-reuse transformations lead to efficient designs. Increasing the number of processors, we the performance is improved due to reduced number of operations that must be executed by each processor. It can be noticed that increasing the number of processors, the large variation of the number of cycles between

successive number of processors decreases resulting into a kind of “saturation”. Therefore,

#cycles_variation%={delay_cost[Nth processor]-delay_cost[(N+1)th processor]}/delay_cost[Nth processor]?100% (8)

That is, the performance gains between successive processor configurations, becomes smaller, and thus the corresponding performance curves get closer.

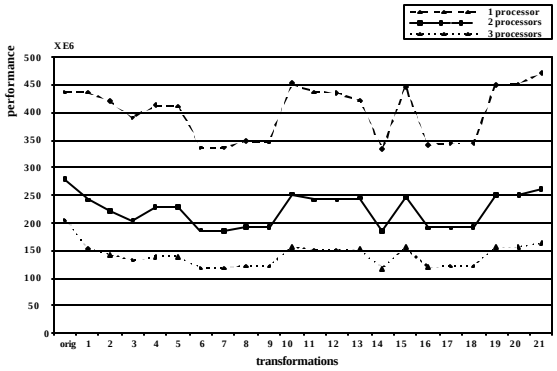

Figure 8. Performance comparison results.

In Fig. 9 the effect of data-reuse transformations on area is illustrated. From that it can be inferred that each transformation influences area in almost identical manner for all processor configurations. It is also clear that all data-reuse transformations for larger *N* increase slightly area, due to increased number of smaller memories. For a number of data-reuse transformations, the processor increase results into increased area (i.e. for data memory), since the transformations insert rather large buffers and eventually, multiple-level memories. In contrary, the transformations 3,12,14,15,16,18,19 and 20 provides efficient designs for all number of processors, since they insert small buffers.

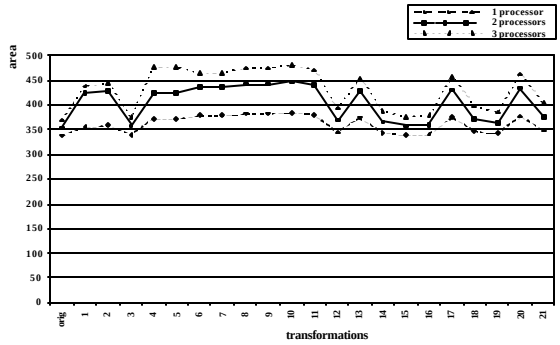

Figure 9. Area comparison results.

6. Conclusions

In this paper the data-reuse exploration is performed for the partitioned version of a real life application and for three alternative programmable processor cores configurations. In all cases application specific, data-memory hierarchy and instruction memory, as well as embedded programmable processing elements, are assumed. The experimental results prove that an effective solution either in terms of power, performance, and area, can be acquired from the right combination of processor core structure model and data-reuse transformation. It is proved that the application of the data-reuse transformations on data memory is independent from the number of processor cores, while it is not true for the instruction memory. Finally, once more, the critical influence of the instruction power consumption on the total power budget cannot be ignored.

References

[1] A. P. Chandrakasan, R. W. Brodersen, Low Power Digital CMOS Design, Kluwer Academic Publishers, Boston, 1998.

[2] F. Catthoor, S. Wuytack et al., Custom Memory Management Methodology, Kluwer Academic Publishers, Boston, 1998.

[3] N.D. Zervas, K. Masselos, C.E. Goutis, “Data-reuse exploration for low-power realization of multimedia applications on embedded cores”, in Proc. of PATMOS’99, October 1999, pp. 71-80.

[4] K. Masselos, F. Catthoor, C.E. Goutis, and "Strategy for Power Efficient Design of Parallel Systems", in IEEE Trans. on VLSI, No. 2, June 1999, pp. 258-265.

[5] S. Wuytack, J.-P. Diguet, F. Catthoor, D. Moolenaar and H. De Man "Formalized Methodology for Data Reuse Exploration for Low-Power Hierarchical Memory Mappings", in IEEE Trans. on VLSI Systems, Vol 6, No. 4, Dec. 1998, pp. 529-537.

[6] U. Eckhardt, R. Merker, "Hierarchical Algorithm Partitioning at System Level for an Improved Utilization of Memory Structures", in IEEE Trans. on CAD, Vol 18, No 1, pp. 14-23, January 1999.

[7] V. P. Roychowdhury, L. Thiele, S. K. Rao and T. Kailath, “On the localization of algorithms for VLSI processor arrays”, in Proc. Workshop VLSI Signal Processing III, Nov 1988, pp. 459-470.

[8] S.Y. Kung, "VLSI Array Processors", Prentice Hall, Eaglewood Cliffs, 1988.

[9] Peter Kuhn, “Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation”, Kluwer Academic Publishers, Boston 1999.

[10] L. Nachtergaele, *et al*, "System-level power optimizations of video codecs on embedded cores: a systematic approach", Journal VLSI Signal Processing Systems, Kluwer Academic Publishers, Boston, 1998.

[11] ARM software development toolkit, v2.11, Copyright 1996-7, Advanced RISC Machines.

[12] P. Landman, Low power architectural design methodologies, Ph.D Dissertation, U.C. Berkeley, Aug 1994.

[13] J.M. Mulder, N.T. Quach, M.J. Flynn,"An Area Model for On-Chip Memories and its Application", IEEE Journal of Solid-State Circuits, Vol. SC26, No.1, pp.98-105, Feb. 1991.

[14] V. Bhaskaran and K. Kostantinides, “Image and Video Compression Standards”, Kluwer Academic Publishers, Boston, 1998.