NoC Design of a Video Encoder in a Multiprocessor System on Chip Solution

Antoni Portero, Ramon Pla, Aitor Rodriguez, Jordi Carrabina
Universitat Autònoma de Barcelona
Edifici Q, ETSE,
08193 Bellaterra, Spain

Abstract - **Future multiprocessor system-on-a-chip (MPSoC) will need high performance and low power requirements due to user demand and limited battery life. In this paper, we take advantage of the architecture flexibility allowed by Network-on-Chip (NoC) to build a parameterizable MPEG Compressor. MPEG compressor System has been developed in synthetizable behavioural SystemC, as a flexible system divided in different tiles. Each tile can be configured as a functional block with different performance parameters in terms of power/speed/area. NoC design implements a 2D mesh with a parameterizable router implemented in high level TLM SystemC. A QoS module takes decisions on packet routing depending on macro-block encoded data (corresponding to I, P and B frames) in real time in terms of power or performance requirements.**

1. INTRODUCTION

Currently, one of the most challenging issues in the telecom market segment is to manage the growing complexity of embedded multimedia systems and to reduce their design productivity gap. These elements stimulate a continuous development of industrial methodologies and design flows exploiting reuse, extended verification, and high level-synthesis.
SystemC [20] is a system-level design language that allows modelling and verification of complex HW/SW components. Starting from some tool-dependent HW coding styles (e.g. Forte Design Systems Cynthesizer, Synopsys SystemC Compiler,...), it is possible to reach the silicon platforms out of an industrial design flow.
Furthermore, the emergence of high capacity reconfigurable devices is starting a revolution in its use for general-purpose computing applications. Many coarse-grain reconfigurable architectures appeared as reconfigurable coprocessors, structures ASICs, etc. considerably relieving the burden from the main processor in many multimedia applications due to their very high degree of parallelism. In addition, they generally have wider flexibility than an application specific circuit. These facts contribute to making them better alternatives to traditionally used DSPs or ASICs.
Coarse-grain reconfigurable architectures are an example of reconfigurable systems. They have identical Processing Elements (PEs) richly connected through programmable interconnections. The PE functionality and connections is configured through context words stored in an internal memory to allow dynamic reconfiguration. Examples of such architectures are MATRIX [1], MorphoSys [2] and REMARC [3]. Multimedia applications are fast becoming one of the dominating workloads for reconfigurable system. Many interactive virtual reality applications such as 3D Games, virtual museum or virtual shop applications have become feasible on reconfigurable systems [4].

1.1. Context and Motivation

Behavioural SystemC description permits a rapid system development. Parts of the systems are locally synchronized by a clock but globally there is a handshaking or a shared-bus communication. When we try to synthesize we do not know exactly the number of resources and clock timing of each tile. Hence, GALS philosophy (Globally Asynchronous Locally Synchronous) independences computation and communication. Any tile of the system can be connected to another if they have the same protocol. One-way handshake protocol is implemented in the example developed here.
Computational systems dispose of higher computation capacities usually limited due to the communication restrictions. To solve this problem, scientific community is beaten for communication based in packet switched taking the idea from the platform based communication between computers. Following SoC evolution driven by the increase in integration density, NoCs (Networks on a Chip) [12] are being used as a communication infrastructure between tiles containing IPs (Intellectual Properties blocks). OCP-IP organization is leading a community trying to standardize the communication protocols and promote the development of tools to automate the NoC strategy for multiprocessor System-on-Chip.

This paper shows an efficient use of NoCs to interconnect several replicated tiles with different computation requirements. Whatever would be system requirements; there will be a PE that can realize the operation in the best way. This is possible, due to the fact that different tiles will be available to realize the

0-7803-9262-0/05/$20.00©2005 IEEE

same operation. The system will have a manager that will decide the task to tile assignment according to performance requirements, basically execution time versus power consumption.

Paper is divided in seven parts; first section is an introduction, a motivation. Second section is about our example, an MPEG Video Encoder developed in behavioural SystemC and their parts (Texture Coding and Motion Coding). Third section explains SIMD architectures. Fourth section, presents our power consumption approximation. Fifth section is related to communication in a mesh-NoC. Sixth section explains our complete SoC. Finally, section seven there are some results and conclusions.

2. MPEG 4-VOB Example

The reference model is based in the behavioural SystemC specification of a MPEG 4 VOB encoder. It can process any king of GOP (Group of Pictures). In our case we are focused on IBBP type of GOPS since they include all kind of coding styles (intra-picture and inter-picture)[13].

We structured our System in different tiles. Each tile has asynchronous communications because behavioural synthesis does not permit to know exactly the number of clocks needed to realize a task. This feature permits orthogonalize communication from computation.

Two basic tiles that can be implemented in this system are: Texture Coding (TC) tile and Motion Coding (MC) tile. TC module is based on the implementation of the Discrete Cosinus Transform (DCT), quantization and zig-zag modules that work on pixels structured as blocks or macro blocks. There is also an inverse Texture Coding (iTC) module that allows retrieving the original image without having to fetch data again from external memory (L2 memory in figure 1). These computed pixel data are re-used to obtain P and B frames. The Motion Coding algorithm implements the inter frame compressor part. MC produces the image differences between frames that will be later codified using a TC module. Data is then passed through a tile that implements the Huffman algorithm for Variable Length Coding (VLC) and finally a video stream is produced according to the MPEG standard [7][8]. There is also a module that controls the whole system function and is the responsible for producing any kind of GOP. This module can change functional parameters of the different tiles in real time.

Figure 1 shows a basic structure containing the functional blocks that will later be mapped to tiles in an interconnected structure.

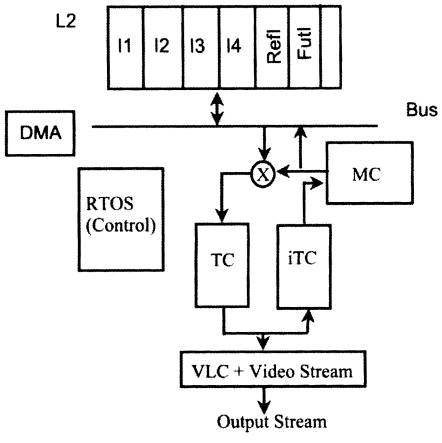

Figure.1 Schema of an MPEG encoder

2.1 Texture Coding Computation

The basic computation for Texture Coding (TC) of a Video Object (VOB) is the DCT that it is the transformation of a NxM image block from the spatial domain to the real part of the frequencial domain. For the image compression standards, usual values are N =M= 8 or 16.

The DCT is an orthogonal transformation. The DCT output can be represented as $Y=TXT^t$. The transformation matrix T (commonly referred to as the forward DCT or simply DCT) can be developed as:

$$(1)\ DCT_{2D}(u,v)=\frac{C(u)C(v)}{4}\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

u, v = 0, 1, ... 7 and $\quad C(u),\ C(v) = \begin{cases} 1/\sqrt{2} & u, v = 0 \\ 1 & \text{otherwise} \end{cases}$

An important property of the 2D-DCT transform is its separability. So, the 1D-DCT can be computed as

$$(2)\ DCT_{1D}(u)=\frac{C(u)}{2}\sum_{x=0}^{7}f(x)\cos\frac{(2x+1)u\pi}{16},\quad u=0,1,...7$$

$$C(u) = \begin{cases} 1/\sqrt{2} & u=0 \\ 1 & otherwise \end{cases}$$

This equation can also be expressed in vector – matrix form as $Z = TX^t$, where T is an NxM matrix whose elements are the cosine function values defined in (2), x = {x0,x1, ..., x7} is a row vector, and Z is a column vector. From (1), the output of the 2-D DCT can be expressed as (3):

$$(3)\ DCT_{2D}(u,v)=\frac{C(u)}{2}\sum_{x=0}^{7}\left[\frac{C(v)}{2}\sum_{y=0}^{7}f(x,y)\cos\frac{(2y+1)v\pi}{16}\right]\cos\frac{(2x+1)u\pi}{16}$$

$$(4)\ DCT_{1D}(x,v)=\frac{C(v)}{2}\sum_{y=0}^{7}f(x,y)\cos\frac{(2y+1)v\pi}{16},\quad x=0,1,...,$$

The equation (4) denotes the output of the 1D-DCT of the rows of f(x,y). The above equations imply that the 2D-DCT can be obtained by performing first the 1D-DCT of the rows of f(x,y), followed by 1D-DCT of the columns of Z(x,v). The output of the DCT is then quantized by a matrix of values Q and then a scan-zigzag is realized (both defined in the MPEG standard).

2.2. Motion Coding Computation

The computation to get P-VOBs or B-VOBs frames is based on two algorithms: Motion Estimation and Motion Compensation. The search algorithms that we have developed for block matching between images are FSME, logarithmic Motion Estimation (logME), and Spiral Motion Estimation [9].

A. Full Search Motion Estimation FSME

In FSME each block of pixels is shifted the whole search area for comparison, as shown figure 2, a. For full-search block-matching, the current MB is shifted to every integer-pel motion vector position. Fast motion estimation techniques, however, employ search strategies which performs a sub-sampling of the motion displacement space. Generally, these search strategies have to cope with the problem of eventually falling into local minimum of the distance criteria function. Search strategies can be regarded as a hierarchical level above the distance calculation.

B. 2DLog: 2D logarithmic Search. TSS: Three Step Search

Fast motion estimation techniques such 2D logarithmic Search (2DLOG) or TDL (two-dimensional logarithmic search) was

proposed by [9]. 2DLOG is a fast search algorithm based on minimum distortion, in which the distortion metric MSE (Mean Square Error Function) is only calculated for a sparse sub sampling of the full search area. The step size of the search area is reduced by n/2 with every search step. TSS was proposed by [9] using a similar structure of 2DLOG, but with the use of MAD (Mean Absolute Difference) instead of the MSE. It can be seen in figure 2b.

C. SpiralSME: Spiral Search Motion Estimation

This is an adaptive data dependent algorithm. The search moves in spiral in all position similar to FSME and stops when the present threshold is passed (Dmin < Threshold value). This algorithm takes into account that solution is near the centre where it starts searching (see fig 2c).

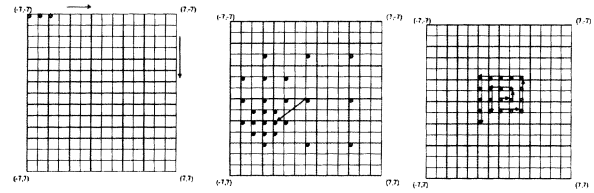

Fig. 2: a) FSME b) TSS(3 Step Search) c)SpiralSME

Definition : MAD- Mean Absolute Difference.

Matching Criterion

$$MAD(i,j) = 1/MN \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} |CP[x+k][y+l]-RP[x+i+k][y+j+l]|$$

$$-p \le i \le p, \quad -p \le j \le p,$$

MV: Motion Vector (u,v) = Minimum MAE

These algorithms permit to obtain the motion vectors and the blocks with the pixel difference values can be compensated with the motion compensation algorithm.

3. SIMD ARCHITECTURES

Concurrent components are those with the ability to process several data in the same way or in different manners at a time. It is convenient to distinguish between the two. The parallelism with elements that treat data with distinct processes is explicitly resolved in the very specification, relying on the SystemC semantics. On the other hand, the parallelism with components that include several copies of the same PE can be described in a more efficient form without explicit detail in the component specification, thus making the refinement procedure simpler. As a result, in this section we shall deal about the specification and synthesis of single instruction, multiple data (SIMD) architectures.

Generally, it is quite easy to deduce a SIMD structure from a ***for*** loop. It is possible to use the behavioural synthesizer capacity of behavioural synthesis tools to get a parallel architecture from an unrolled loop. Taking into account the SIMD model, input and output data must be stored in vector arrays whose access would be controlled by the control loop variables. Note that, in this case, there are two nested loops and there is also a carry variable (***s_reg***) that is directly transmitted to successor PE in the unrolled version.

Directive in unrolling loops permit to get a PE with different performance. For example unrolling two inner loops permit to create a parallel structure with 64 MAC (multiplier adders). Putting sequentially ***wait()*** statements after each loop; the structure is completely different with just 2 MACs.

4. Power Consumption

Power consumption can be divided in three kinds of power: switching power, short circuit power and leakage power. In our approximation, as usual when considering functional design approaches, just the switching power of a CMOS is take into account. Although leakage power can not be neglected for deep-submicron technologies, it is a constant contribution independent on switching activity of the system. Switching power of a CMOS gate : $P = \frac{1}{2} C V_{dd}^2 f \eta$ where VDD is power supplied, f clock frequency, C output load capacitance and η switching activity factor.

High level power estimation is based in how many times L1 memories are transmitting data to other memory structures or registers [17][18][19]. Taking into account that execution time to load and store data is where system spends more energy, it is possible to count the number of times that this processes are needed to carry out the algorithm. By executing the algorithm using a high level model (in this case SystemC) that includes the implemented concurrence, these numbers can be obtained and stored in a file. Afterwards, we use cacti 3.2 [10] to get estimation results about power consumption together with the execution time to process data. Cacti power estimations are not very accurate but we consider them good enough for a high level knowledge about the system power behaviour. Figure 3 shows the macro #sim_Power that obtains the number of load-store information in memories.

Another advantage of a SystemC description is that it is platform independent as is C++ code. The model can be changed to fit in different embedded PE and allow using it processor Instruction Set Simulators (ISS) to get information about execution time and other power models.

```
MAIN DATA STRUCTURES
sc_int <bus_length> dummy[BLOCK+1];
sc_int <bus_length> X[BLOCK][BLOCK];
sc_int <bus_length> DCT[BLOCK][BLOCK];
I
#define power_SIM 1
#define Synth 0
Algorithm
  j = 0;
  #ifdef Synth PIPELINE #endif
  DCT1j: while( j < N ) {
    DCT1i: for( i = 0; i < N; ++i ) {
    #ifdef Synth MACRO_FOR_UNROLL_LOOP1 #elseif
          wait();#endif
      s_reg = 0;
      MACS1: for( k = 0; k < N; ++k ) {
      #ifdef Synth MACRO_FOR_UNROLL_LOOP2 #elseif
          wait();#endif
        dummy[k] = X[i][k] * dct[j][k];
        s_reg = dummy[k] + s_reg;
                    #ifdef power_SIM
                    counter_Mult_X_dct ++;
                    counter_DCT_X_dct ++;
                    #endif
      } // for MACS1
    } // for DCT1i
    ++j;
  } // while DCT1j
Example 1 SIMD Architecture
```

Figure 3. 1-DIM DCT SIMD Architecture

4. COMMUNICATION

Since behavioural hardware descriptions use handshaking for the communication between modules, it is possible to insert a router

inside each tile of our system [12]. Hence, a 2D mesh network is implemented based on XY-Routing algorithm.

The router is based on the one proposed in SoCIN[5] and RaSoc[6]. It is a router composed by five channels: L (Local), N (North), S (South), E (East) and W (West), where each one of them has both an input and an output. Each channel is composed by two unidirectional opposite channels, each one with its channel data and two control signals (*val* and *ack*) for handshaking.

In our case, N, S, E and W are *sc_fifo* channels and L is a *sc_signal* channel. In spite of the fact that the *sc_fifo* channel is a not in the SystemC synthesisable subset, it let us to implement a fast simulation of the system and determine the First In First Out (FIFO) sizes for a future synthesis of the system. Sc_fifo is a SystemC channel that may contain any data type and it is used to manage data flow (therefore being a high-level software model).

The router has five SC_CTHREAD processes (clocked thread process), one for each input channel. SC_CTHREAD is the only process of the three that has SystemC (SC_METHOD, SC_THREAD and SC_CTHREAD) which let to use the *wait_until()* call. This instruction let us to implement the handshaking communication. At the system starts working, each thread is waiting for a *val* event, which means that there is some data in the channel to be read.

With five independent processes in the same router we can address simultaneously different information coming from different channels that goes to different destinations. For example, one router may address data that comes from S and goes to L and at the same time data that comes from E goes to N.

In cases N, S, E and W, when their respective threads are activated, the wrapper will check the routing information of the header of the incoming block and, if it is not the destination, it will update the RIB (Routing Information Bits) from the header (decreasing the Xmod or Ymod fields) and it will address it to the output channel. If the router that reads the block is at the destination address, the wrapper will extract the NxM block data information, but not the header, and send it to the resource connected to the L port, used by the tile.

The router has a *wrapper* that generates the header of the packet (block). This wrapper has the routing table depending on the **xy** coordinates of the router in the network. Table I shows the general routing table.

The packet data contains one header and one NxM (normally N=M = 8 or 16) pixel data block. The image type determines the block image size, that is composed of an integer multiple of NxM. Consequently, according to the image type, it may be necessary to do α transition of packet structures with image blocks of NxM size.

For example, the transmission of an I-image will require the next steps: first to pack the information (NxM). Secondly to indicate in the header that it is an I-block and the routing information of the RIB. Finally, it will be send to the next router according to the RIB information.

However, in cases of B and P blocks, the block images size is 2P x 2P (P = Search Region from 16 to 64). Each one of these 16 blocks is packed by the corresponding header. All 16 packet headers will have the same RIB information because they belong to the same image, which goes to the same destination tile.

TABLE I. ROUTING TABLE

Macro Block Type	Resource	Next Resource
I	S	TC
I	TC	TS, ITC**
I	ITC	MC
I	MC	TS
B	S	MC
B	TC	TS
B	ITC	*
B	MC	TC
P	S	MC
P	TC	TS
P	ITC	*
P	MC	TS

* Images P and B do not need ITC processing. In these cases the resource attached to the router do not operate and the router only addresses the block.

** In case of Image I; afterwards, to get image P and B macro-blocks, we also need previous original image.

In the router declaration we specify the name, type and coordinates **x** and **y**. In this way, each router can calculate for each packet their RIB. The resource coordinates are shared in a common file.

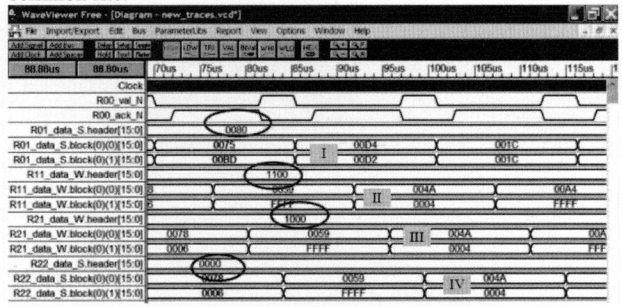

Figure 4: Sequence routing waveform trace of an image I inside our NoC. Just relevant signals are showed for better understanding. From each router, only two pixels data are showed from all NxM pixels.

The waveform trace obtained after simulating the texture coding (TC) of an image block is shown in figure 4 together with the routing paths in spatial compression of an I image.

The numbers on the waveforms correspond to the steps that data follow in the network shown in figure 4. In step I, data leaves the source tile with routing information in the header, indicating that the destination tile has 01 **xy** coordinates. Router 01, after receiving and checking header information and image type information, passes data to the TC2 resource connected to the L port. Afterwards, when TC2 has finished, router 01 calculates the new header according to data image type. As the image type is I, the resulting data must be sent to VLC-VS module. According to the **xy** routing algorithm, to go to the VLC-VS tile (with coordinates 22), the routing is done first in the **x** direction (steps II and III) and finally in the **y** direction (step IV).

Steps III and IV are faster than step I because information is not processed in the intermediate routers. Routers 11 and 21 only update the header and address the information from the incoming port to the outcome port. That is from port W to E, in case router 11, and from W to N, in case router 21.

Furthermore, marked with circles, we can see how the header information changes at each step but not the data, which only changes after the TC2 in step II.

The step IV header signal is null because at the last step, data arrives to destination (Xmod and Ymod values are 0), and it comes out from port N (Xdir value is 0).

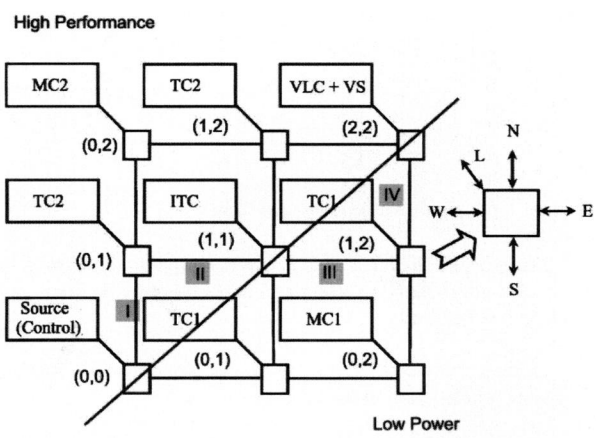

Figure 5. Mesh 3x3 NoC topology

6. SoC design

Our proposal is to dispose in the same system on chip different tiles specialised in different performance power balance. With this different cost/performance tiles, the routing assignment will be implemented according to the system level requirement, so that for low power implementation we will prefer to move first to low power implementations (lower right or **x** direction in figure 5) whereas for high speed we will prefer the corresponding high speed implementations (upper left of Y direction in figure 5). The resources distribution in this network is symmetric, so that the RIB calculation will be the same in both cases with the difference that the coordinates **xy** of a high performance implementation will be translated to **yx** (**x→y** and **y→x**) coordinates for low power implementation.

7. Results

All the values shown are relative to the reference case (figure 1) with high performance encoder implementation. In Figure 5, the reference case, High performance FSME corresponds to the computational part (in dark) of the bars 4 to 6, whereas the communication part due to NoC (in clear). Complete bars correspond to the final implementation including NoC routing.
Low power implementations has always higher execution time that the high performance ones due to the lower parallelism of their computation. The use of logME or SpiralSME (with Dmin <16) let to a factor 7 for P Macro-block and almost 10 for B macro-block in gain respect reference case. Remember that in case of SpiralSME execution time is completely dependent on the similarity of images; so, computation can decrease a lot if no-movement is perceived between images whereas worst case of SpiralSME is approaching to FSME when images are completely different.

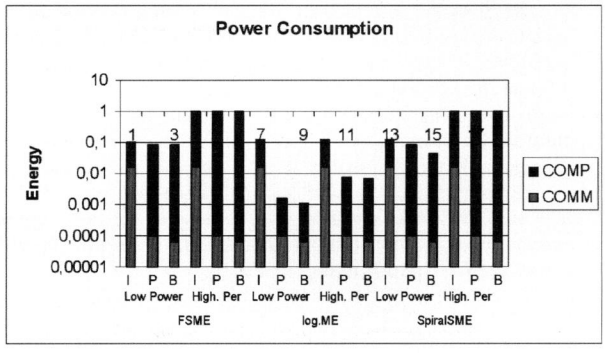

Figure 6. Time execution in clock cycles.

Higher Power consumption values are in majority due to computation in FSME and SpiralSME because they are sharing the same data memory transfer structures.
Power consumption in Spiral (low power) is reduced 10 times without losing to much Quality. In case of logME, power decreases a higher factor but loosing more Quality [13].

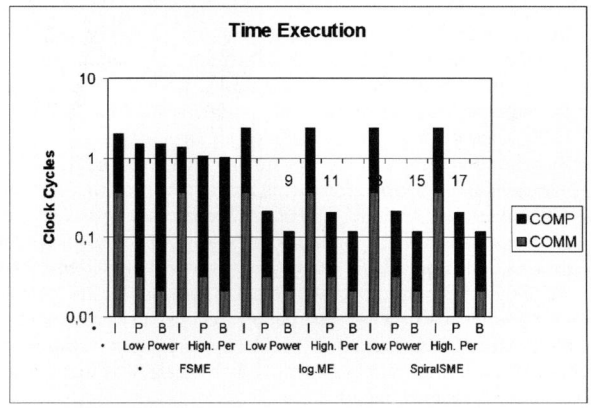

Figure 7. Power Consumption in nJ related with Fig.1.

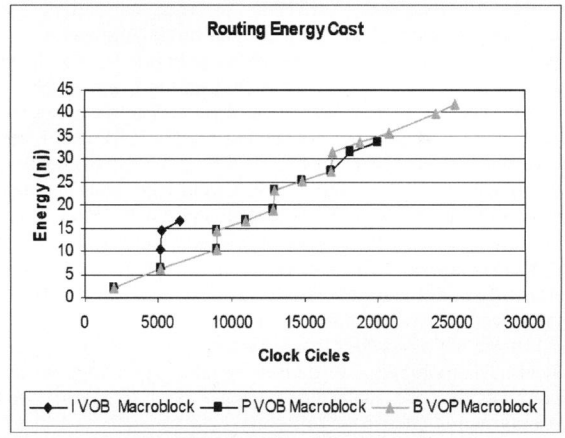

Figure 8. Power consumption and NoC latency for different macro-blocks.

In figure 8, it is shown the power consumption versus clock latency for I, P and B macro-blocks due routing through the NoC structure.
This high performance description has been synthesized with CoCentric System Studio in a Stratix FPGA EP1S30896C5 device, resulting in a 27 MHz for high performance MC tile. Low power solution without loop and any loop unrolling has been synthesized, resulting a 57 MHz solution with just using two DSP blocks. Complete synthesis results are shown in table II and III.

TABLE II
SYNTHESYS HW SUBSYSTEMS

		High Perf.	Low Power
Motion Coding	Device	EP1S30896C5	EP1S30896C5
Block_ME	Total logic El.	24634 (76%)	1537/32470 (5%)
	Total Mem. bits	540672 (16%)	16.896/3317184
Block_MC	Total logic El.	412 (<1%)	412/25660 (<1%)
	Total Mem. bits	16.896(<1%)	16.896/3317184
Texture Coding	Total logic El.	32,438 (99%)	3,167/32470 (10%)
(DCT + Q+ Zig-za	Total pins	23 (3%)	23/692(3 %)
	Total Mem. bits	0	5604(<1%)

DSP block 9-bi El.	**64** /96	(67%)	**2**/96 (2%)
Max Freq	26.99 MHz		57,47 MHz

TABLE III
SYNTHESYS TS (TRANSPORT STREAM MODULE) ON A RISC PROCESSOR (VLC + HEADER BUILDER)

Device	EP1S30896C5
Total logic elements	3863 / 32470 (12 %)
Total pins	215 / 692 (31 %)
Total memory bits	421888 / 3317184 (13 %)
DSP block 9-bit elements	2 / 96 (2 %)
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 2 (0 %)

Conclusions

This paper presents a Video Encoder divided in tiles that are connected to a NoC. Usually System on Chip developers work designing for worst case implementation. In the proposed work, a computation replica specialized in low power or high performance is designed and connected to a mesh-NoC. Adding the communication time in the NoC, and in a very dynamic way (macro block level), several working points are added. Hence, computation is realized depending on multiple targets: low-power higher time consuming, higher performance but high power consuming, macro-block type, kind of computation, quality of the image etc.

REFERENCES

[1] E. Mirsky, A.DeHon, et al. "MATRIX: A Reconfigurable Computing Architecture with Comfigurable Instruction Distribution and Deployable Resources", Proc. IEEE Symposium FCCM, Apr.1996.

[2] H. Singh, M.Lee,G.Lu et al. "MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Applications", IEEE Transactions on Computers,Vol.49,No5, May 2000.

[3] T.Miyamira and K.Olukoton, "REMARC: Reconfigurable Multimedia Array Coprocessor", Proc. ACM/SIGDA International Symposium FPGAs, Feb 1998.

[4] M. Meissner, S.Grimm, W.Strasser, J.Packer and D. Latimer, "Parallel Volume Rendering on a Single-Chip SIMD architecture", Proc. IEEE Symposium Parallel and large Data Visualization and Graphics, pp.107-157,Oct 2001.

[5] C. A. Zeferino, A. A. Susin, "SoCIN: A Parametric and Scalable Network-on-Chip", IEEE Proceedings of the 16th Symposium on Integrated Circuits and System Design (SBCC'03).

[6] C. A. Zeferino, M. E. Kreutz, A. A. Susin, "RaSoC: A Router Soft-Core for Network on Chip ", IEEE Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Designers Forum (DATE 0'4).

[7] ISO/IEC 13818: 'Generic coding of moving pictures and associated audio (MPEG-2) ITU-T Rec. H.262, ISO/IEC 13818-2, . Standard, Oct. 1994. 8. ISO/IEC 14496-2: 'Information Technology-coding of Audio-Visual Objects' (MPEG-4) ITU-T, Standard, Dec 2001.

[8] Forte-Cynthesizer www.forteDS.com

[9] Peter Kuhn, "Algorithms, complexity analisi and VLSI architectures for MPEG4 Motion Estimation", Kluwer Academics, 1999.

[10] Cacti 3.2. Power Models http://research.compaq.com/wrl/people/jouppi/CACTI.html

[11] Wayne Burleson, Prashant Jain, Subramanian Venkatraman, "Dynamically Parametrized Architectures for Power Aware Video Coding: Motion Estimation and DCT", ...

[12] David Bertozzi et al, "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip ", IEEE Transactions on Parallel and Distributed Systems, vol 16, No.2, Feb.2005.

[13] V. Bhaskaran and K. Konstantinides, "Image and Video Compression Standards-Algorithms and Architectures", Kluwer Academics Publishers, Second Edition, 1997.

[14] L. Nachtergaele et al. "Optimization of Memory Organization and hierarchy for decreased size and power in video processing systems", in Proc. Int. Workshop on Memory Tech. Aug. 1995.

[15] Blind review

[16] Blind review

[17] F.Rivera et al, "Efficient Mapping of Hierarchical Trees on Coarse-Grain Reconfigurable Architectures", CODES-ISSS Set. 2004.

[18] Francky Catthoor, "Data Access and Storage Management for Embedded Programmable Processors", Kluwer Academic Publishers, 2002.

[19] Andy Lambrechts, et Al. "Design Style Case Study for Embedded Multimedia Compute Nodes", The 25th IEEE Real-Time System SymposiumRTSS Dec. 2004.

[20] SystemC Organization www.systemc.org