A VHDL-based Design Methodology: the Design Experience of an High Performance ASIC Chip

M. Valle, M. Cornero, G. Nateri, D. D. Caviglia

DIBE – University of Genova
Via Opera Pia 11/A
16145 Genova, Italy

L. Briozzo

SGS-Thomson Microelectronics
Via Olivetti 2
Agrate Brianza (MI), Italy

Abstract

In this paper we present a VHDL-based design methodology which we adopted in the design of an ASIC chip for real time image analysis in a quality control industrial environment. The design methodology is based on the following considerations: i) we explored the design space by applying some high level transformations on the VHDL specifications; ii) we defined some VHDL structural modeling (and design) guidelines to take into account, already in the first design steps, some physical issues which, otherwise, could cause incorrect behavior.

The presented methodology proved to be efficient in avoiding design iterations and giving reliable high performance circuits.

1 Introduction

In the ASIC design, issues such as speed, chip complexity and size, are becoming more and more crucial as the desired performances increase: to obtain high performances, the designer must perform a consistent effort in exploring the design space. This goal can be more easily reached if an efficient design methodology is employed, in particular, if the design space is explored already in the first steps during the high level modeling of the architecture. On the other hand, the design methodology to be efficient must concentrate the iteration loops within the first design steps and must avoid iterations between the physical and the first design steps. This goal can be reached in two ways: i) as already mentioned, exploring the design space in the first steps; ii) adopting a "safe" and efficient circuit modeling approach.

In this paper, we illustrate the design methodology which we adopted in a real ASIC design (a co-processor for real time image analysis system in a quality control industrial environment). The methodology is based on the above considerations and on the usage of VHDL. We explored the design space performing high level transformations on the VHDL specifications during the high level modeling of the architecture. Then, we modeled the circuit blocks at RT level in VHDL, on the basis of some simple but safe modeling (and design) rules which take into account physical issues, mainly, but not only, related to the chip high performance specifications. Such issues cannot be taken into account during the high level modeling but they may cause an incorrect behavior and then some design iteration loops. In the end, we synthesized the circuit and performed the physical design.

In Section 2, the functionality implemented by the chip architecture is presented. The design methodology is illustrated in Section 3. In Section 4 the high level modeling approach and the high level transformations which we performed to meet the specifications are described in a deeper detail. The result of Section 3 is the partitioning into blocks and the architecture of the chip. The VHDL modeling at RT behavioral level and some "safe" modeling guidelines, are illustrated in Section 5. The conclusions are drawn in Section 6.

2 Functionality of the ASIC chip

The chip implements a neural network architecture for feature recognition applied to structural defect detection in ferromagnetic materials. It acts as a co-processor of a general purpose processor that manages the I/O operations and a frame buffer for image storing.

The neural network receives in input an image obtained by fluorescent magnetic particle inspection, [5]...
of mechanical pieces and gives as output a binary code indicating whether a physical defect is present or not in the piece.

The network is based on a supervised neural algorithm [4]. It convolutes the input image with an optimal filter (the filter profile is determined by a set of weights off-line learned); through local, non linear post-processing of the convolution results, the network obtains a binary output indicating the presence of one or more structural defects. The network detects defects that are mainly vertical in shape, thus to detect horizontal defects we need to give in input to the network also the transposed image.

The operations performed by the neural network can be summarized as follows:

1. Convolution of the input image. The input image has size $512 \times 512$ pixels and the convolution mask has size $5 \times 9$ weights.

2. Partitioning of the convoluted image into sub-images of size $24 \times 28$ pixels partially overlapped and computation of the maximum value for each row of the convoluted sub-images.

3. Computation of the sum of the maxima and decision (comparison with a threshold) for recognition of defects for each sub-image.

3 Design methodology

The design methodology is shown in Figure 1. We started from a set of specifications (system requirements), issued by the end user (size of the input image, size and features of the convolution mask, real time speed, heuristics for the decision about the presence of the defect etc.) and from the neural network architecture (e.g. topology of the network, functionality of neurons and synapses, learning algorithm, etc.) which was modeled and simulated at algorithmic level.

We performed the high level modeling of the convolutional functionality and we identified the computational bottleneck in the computation of the convolution; then we applied a set of high level transformations to the kernel of the convolution to meet the specifications. The high level transformations identify quite easily the partitioning of the chip architecture into blocks and we detailed the chip architecture which was described at RT-level. We modeled in a synthesizable way, all logic blocks and, using a logic synthesis tools, we mapped the architecture onto a real circuit in the target technology. The output of the synthesis is an EDIF netlist which has been used as input for the physical design. In the physical design phase, same macro-blocks (e.g. RAMs) were generated using macro-blocks generators.

3 High level modeling and transformations

The convolution of the input image is expressed, in mathematical form, as follows:

$$cnv_{pix_{i,j}} = \sum_{m=0}^{4} \sum_{n=0}^{8} img_{i+m,j+n} \times mask_{m,n}$$

with $0 \leq i \leq 507$, $0 \leq j \leq 503$ (1)

Figure 1: Design methodology.

Figure 2: High-level modeling of the convolution block.

All design steps, except for the physical design, were performed using a VHDL based approach: the synthesis tool is Synopsys.

The physical design was performed using ST-Unicad, a design kit of SGS-Thomson based on Cadence Design Framework II. The validation of the design at physical level was performed through HDL simulations (i.e. Verilog).

4 High level modeling and transformations

The convolution of the input image is expressed, in mathematical form, as follows:
for i in 0 to 507 loop
for j in 0 to 503 loop
acc := 0;
for m in 0 to 4 loop
acc := acc + img(i+m,j)*mask(m,0);
acc := acc + img(i+m,j+1)*mask(m,1);
..........................
acc := acc + img(i+m,j+8)*mask(m,8);
end loop;
conv_img(i,j) := acc;
end loop;
end loop;

Figure 3: High-level modeling after loop unrolling.

where \( img_{i+m,j+n} \) is the \((i+m)^{th},(j+n)^{th}\) pixel of the input image, \( enc\_pix_{i,j} \) is the \(i^{th},j^{th}\) pixel of the convolved image and \( mask_{m,n} \) is the \(m^{th},n^{th}\) pixel of the convolution mask. The high-level, behavioral specification of the convolution block, derived from equation 1, is shown in Figure 2. The input image is supposed to be stored in the bi-dimensional array \( img \) and the convolution mask in the bi-dimensional array \( mask \). The two innermost loops perform the convolution operation on each input pixel \( img(i,j) \), as indicated in equation 1, while the two outermost loops perform the iteration through all the pixels of the input image.

As it is easy to verify a direct implementation of this specification, i.e. a circuit composed of a single adder-multiplier performing the inner loop, would result in an unfeasible solution. In fact, as already said, the convolution block, like the rest of the circuit, must elaborate in real time the input images which are generated at a rate of 25 images per second. Moreover, to detect defects which are mainly horizontal in shape, for each image one must elaborate the image and its transposed; then the real operational speed is 50 images per second. It follows that the convolution of each single image must be performed in 0.02 s. On the other end, the body of the innermost loop of the specification must be performed 508 \times 504 \times 5 \times 9 = 11,521,440 times, i.e. with a maximum delay of 0.02 / 11,521,440 \approx 1.74 \text{ ns} for a single multiply-accumulate operation, which is clearly not feasible.

In addition to the performance bottleneck, following the specification of Figure 2, the order used to access each single pixel of the input image implies an inefficient memorization scheme.

A straightforward transformation of the original specification is the unrolling of the innermost loop, as shown in Figure 3, in order to extract the implicit parallelism contained in the original specification. The innermost loop body of the transformed specification is composed of a set of 9 statements, each of them depending on the result of the previous one (acc). Because of this data-dependence the parallelism of these statements cannot be exploited. However, as indicated in Figure 4, if the order of the \( m \) and \( j \) loops were inverted, it would be possible to exploit the parallelism by means of pipelining, using an high-level transformation called loop folding[3]. In fact for the pipelined implementation to work the input data \( img(x,y) \) should be processed by feeding in the circuit with successive pixels with respect to the index \( y \), while in the specification the innermost changing index is \( x \) (:i:m).

This observation suggests a new transformation, i.e. the inversion of the ordering of the two innermost loops, so as to obtain the required input order-
Figure 6: Final specification and architecture of the convolution block.

In Figure 6 each multiplier has been implemented using a lookup table (LT), where each field contains the result of the multiplication of the field address multiplied by the proper mask weight. The maximum allowed delay of each pipelined stage is now:

\[
\frac{0.02}{(508 + 4) \times (504 + 8)} \approx 76\text{ns},
\]

where the term +8 takes into account the delay required to fill in the pipeline. Since each pipeline stage must perform an addition and a multiplication, this timing constraint is still too restrictive.

An additional speedup of the circuit can be obtained by unrolling the loop, and by executing each instance of its body in pipeline and with a separate block, adopting again the loop folding technique, as illustrated in Figure 6. The pipeline execution of the five blocks implies the memorization of four instances of the buffer \texttt{buff}, which can be implemented as a FIFO thanks to the execution ordering. As indicated in Figure 6 each multiplier has been implemented using a lookup table (LT), where each field contains the result of the multiplication of the field address multiplied by the proper mask weight. The maximum allowed delay of each pipelined stage is now:

\[
\frac{0.02}{(508 + 4) \times (504 + 8)} \approx 76\text{ns},
\]

The delay for a multiply-accumulate operation as calculated above is greater than that needed; anyway, we can utilize the extra time to increase the number of images per second or to process each image with, let say, two different weight mask sets.

A relevant advantage of the above implementation is that the input image is processed one pixel at the time, starting from the first pixel of the first line, proceeding with all pixels of the first line, then with the first pixel of the second line and so on until the last pixel of the last line. In this way complex memorization schemes are completely avoided.

5 RT level modeling of circuit blocks

Once detailed the chip architecture, we modeled it at RT level in a behavioral way (i.e. no delays were taken into account). The blocks were divided in two categories: synthesizable and non-synthesizable blocks. The former ones were mapped on the technological library using a synthesis tool while the latter ones were generated using the macro-blocks generators of ST-Unicad: we modeled their behavior only for simulation purposes.

In this Section we focus our attention on the modeling of synthesizable blocks. In particular, we will describe the modeling of the pipeline schema and some safe modeling (and design) guidelines.

5.1 Safe modeling (and design) guidelines

Iterations during the design process are time consuming tasks, in particular at physical level. A good design methodology should be enough reliable to avoid them. Our design methodology is based on some design guidelines which take into account, already in the first design phases, some physical issues which cannot be efficiently modeled in VHDL (e.g. clock skew, safe management of timing signals of storage elements, etc.) and can cause incorrect behavior at physical level. The design guidelines are based on some simple rules (e.g. [2]) and, if adopted at the RT level modeling of the chip architecture, give a reasonable confidence on the reliability and efficiency of the circuits which will be synthesized.
Safe clock management and distribution. An high frequency master clock signal (M.CLK) is used and all other clocks are generated from it. Moreover, we assume that stability exists before each clock transition, [1].

The clock management policy must be very careful to allow reliable and safe distribution of the clock signals on the chip (reliable management of the clock skew).

We decided: i) to use a single-phase docking scheme to decrease the number of clock phases; ii) to distribute M.CLK in a small area and to generate from the it the low frequency clocks to distribute on the whole chip.

Following this approach, the constraints on the clock skew are less severe and can be handled more easily during the physical design phase.

Safe modeling of storage elements (D flip-flops, D-ffs). D-ffs must not have their own clock signals gated with other control signals. Two main motivations underlie to this approach: i) a glitch on the gate output can cause a clock edge and then an undesired data assignment; ii) gating the clock line introduces clock skew.

In Figure 7 a), an example of a unsafe circuit description of a D-ff with an asynchronous enable signal EN is shown. In Figure 7 b) a "safe" circuit description for the same example, is shown.

Similar considerations can be made for an asynchronous EN_RES signal that, when active, sets the value stored in the D-ff to 0. In Figure 8 a) an unsafe circuit schema is shown: the reset signal of the D-ff is not directly synchronized with the system RESET signal and an undesired activation of the local reset signal may occur.

In Figure 8 b) a safe circuit configuration and VHDL description of the same circuit is given. The signal EN_RES is synchronized with the system clock and the local reset is directly connected to the system RESET signal.

5.2 Pipeline modeling

As explained in Section 4, the time scheduling of the chip architecture is based on many pipeline stages. VHDL has no built-in constructs for pipelining, [6]; the pipeline management must be explicitly described in the VHDL modeling and concerns the storage elements (D-ffs).

We modeled each D-ff as an independent process synchronized on a global timing signal. The modeling is illustrated on the basis of an example. In Figure 9 two single bit pipelined adders together with four registers (D-ffs) are shown. The example shows a subset of the tree of adders of the convolution blocks. The adders and the registers are arranged in a bit slice structure. To increase the number of bits, we can arrange some of these structures in parallel.

We designed a clock management schema (see Figure 10 and Section 5.1) based on two different clock signals: i) an high speed clock (e.g. 50 MHz, master clock, M.CLK) distributed on a small area (control block) and ii) a clock with a frequency half of the M.CLK frequency (submaster clock, S.CLK) distributed all over the chip.

The pipeline stages in the example of Figure 9 are synchronized with the submaster clock S.CLK. The D-ffs (A0_add, B0_add, A1_add, B1_add) are modeled using processes synchronized on the edges of the S.CLK; each D-ff is modeled using two processes ("master" and "slave"); the "master" process is synchronized on the rising edge of S.CLK, and the "slave" process is synchronized on the falling edge of S.CLK. The
sums are computed when the input data (e.g. A0_add, B0_add, etc.) are available. During each S.CLK period, two sums are computed in parallel.

In our pipeline model we didn’t take into account the block and line delays: our aim was to model the functionality of the chip while the timing analyses were delayed to the physical design phase.

In Figure 10 the timing diagram and the pipeline schema for the circuit of Figure 9 is illustrated. In Figure 9 the VHDL description is also detailed.

6 Conclusions

In this paper, we presented the VHDL design experience of an ASIC chip for image processing in quality control industrial environment. We defined a design methodology based on the following considerations: i) the exploration of the design space is performed in the first design steps applying high level transformations to the VHDL specifications; ii) we defined a "safe" modeling (and design) VHDL methodology to take into account at RT behavioral level, circuit and technological issues and constraints. The design methodology so defined turned out to be efficient and reliable.

The design flow is mainly organized in two steps: i) a VHDL based front end (specification, simulation and synthesis) phase; ii) back end (physical design) phase. The RT level modeling of the chip has been extensively simulated in order to evaluate the correctness of the architecture. In the back end phase, the verifications were performed using an HDL (Verilog) simulator because of its efficiency and reliable cell library component modeling. The back end phase was performed using the SGS-Thomson Unicad (ST-Unicad) Design Framework that is based on the Cadence Design Framework II.

The chip complexity is of about 13,000 standard cells and 7 RAM macroblocks. At present time, the design is completed and it has been validated by the final HDL simulation. All physical checks (DRC, ERC, etc.) are being performed and the realization is expected in three months.

The size of the chip is about 1.0µm x 1.0µm in the SGS-Thomson 0.7 µm HCMOS4T technology. The speed resulting from the HDL simulation, that take into account the cell and wiring delays, is 40 MHz.

Acknowledgements

The authors would like to thank Dr. Luigi Raffo, Dr. Paolo Faraboschi and Dr. A. De Gloria (DIBE, University of Genoa) and Dr. G. Voglino (SGS-Thomson Microelectronics).

References