

VERIFICATION OF ADVANCED COLLABORATIVE INFRASTRUCTURE BY AFFINE FPGA DESIGN

K. Siekierska, D. Obrębski, A. Kokoszka,
N. Ługowski, A. Pawlak
Institute of Electron Technology (ITE)
Warszawa, Poland
{k.siek, obrebski, kokoszka, lugowski, pawlak}
@ite.waw.pl

M. Carballeda, B. Schlichter
Thales Optronique S.A. (TOSA)
Guyancourt, France
{manuel.carballeda, bruno.schlichter}
@fr.thalesgroup.com

***Abstract:** The paper presents the distributed design process of Affine FPGA performed in a collaborative way between ITE and TOSA. The additional aim of this design was to verify the Advanced Collaborative Infrastructure (ACI) developed by the E-Colleg project. The design specification was defined by TOSA following rigorous industrial requirements. Details of the Affine FPGA design are presented, preceded by a short presentation of the ACI and especially GUIs that have been developed for the E-Colleg ACI. Additionally, some important issues related to the collaborative design process are presented. One of them is a selection of distributed tools which remote invocation is justified by design process efficiency and economical reasons.*

1 Introduction

Collaborative engineering constitutes a new engineering paradigm that enables efficient product development by a distributed team of engineers who are integrated by an appropriate infrastructure. A number of R&D projects, like DSE [4], ISTforCE [5], TOCEE [6] or WELD [7], were aiming at either whole infrastructures or tools that enable collaborative engineering in intranets and/or in the Internet [1, 2].

Following these trends C-Lab has developed the ASTAI® environment [3] which can integrate remote tools. These remote tools can form in particular a distributed IC design environment. The experiments undertaken in the E-Colleg project have however pointed out to the firewall problem, as firewalls establish a real obstacle for data transfer in an ASTAI® workflow. Thus, a need for new solutions which allow for Internet-wide secure collaboration, and in particular allow for controlled data transfer through firewalls.

This challenge has been addressed by the E-Colleg project development of the Advanced Collaborative Infrastructure (ACI) [10]. The ACI prototype that has been developed in a frame of the E-Colleg project allows crossing of firewalls transparently. Advanced Network Transport Services (ANTS) are a set of services which assure secure peer-to-peer transport of data using network infrastructure. ANTS are used by TRMS (Tool Registration and Management System) to perform data transmission over firewalls. The system has been developed on the Java platform with commonly available technologies and tools, like UML,

XML, SQL, and Web. The TRMS environment is composed of one ANTS server, which is connected to the Internet directly (without a firewall) and a lot of client and server computers (PCs or workstations) that can be connected to the Internet by firewalls. The infrastructure allows to retrieve files, synchronise client/server databases and invoke tools remotely.

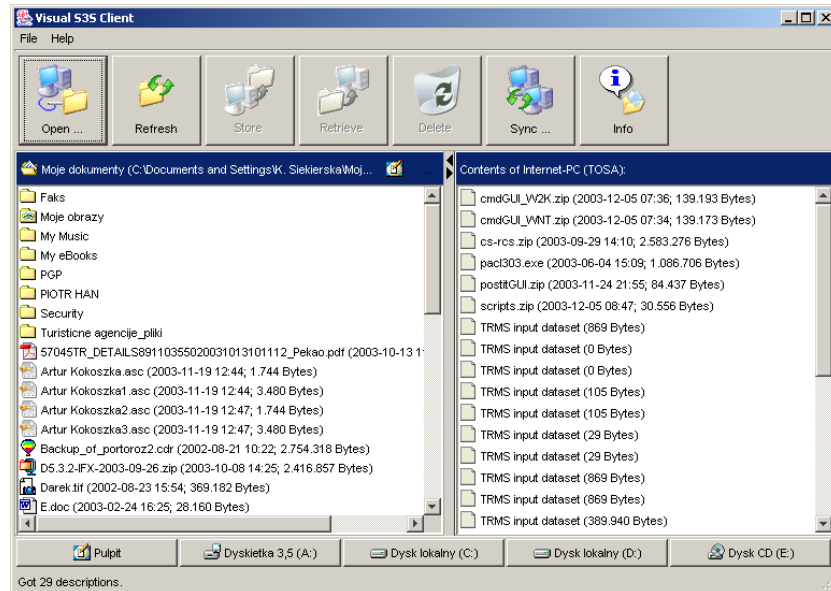


Figure 1. Visual S3S Client GUI.

A distributed collaborative environment requires especially well developed graphical user interfaces (GUIs), as dispersed engineers need to be well aware on the availability of distributed design files. Here the goal is an engineer friendly Workflow Management System (WfMS) that supports management of the design tasks flow. Many WfMSs have been developed in the last decade, nevertheless, only the very few have had engineering applications as an objective [1, 3].

E-Colleg didn't develop a new WfMS, nevertheless a number of interesting experiments were conducted with simpler GUIs.

2 Graphical User Interfaces

There are some GUIs developed for the ACI to make the environment user friendly.

The Visual S3S Client (Fig.1) is a simple GUI, developed by C-Lab [3], to facilitate data transfer between onsite and a selected node. Selection of the needed server (one of available at TOSA) from ones seen in the list (all the nodes with running ANTS) is done by Open command. The GUI aides transfer (Retrieve, Store, Sync commands) of files. The Sync command is very useful in collaborative engineering, because it allows to store the same versions of the files on both side (onsite and remote server) after each modification carried out by one of the partners. The GUI performs well but it offers too few functionality and there is no possibility of differentiate clients permissions – each node is accessible by any other server. This feature is a big disadvantage from the security point of view.

Two GUIs were developed in TOSA. POSTIT GUI (Fig.2) is an interface developed for communication between project members from TOSA and ITE working on TRMS nodes (each node has a dedicated button). It still uses scripts written before and invoked previously

from command line. CmdGUI (Fig.3) is an interface developed for invoking the ITE/TOSA design workflow under the TRMS infrastructure. It is very useful for designers who take part in collaborative design and need access to remote tools and who have to synchronise a common data base. It still uses scripts written before and invoked previously from command line. An additional disadvantage is that this environment is closed, as it is dedicated to the selected workflow integrating only selected tools.

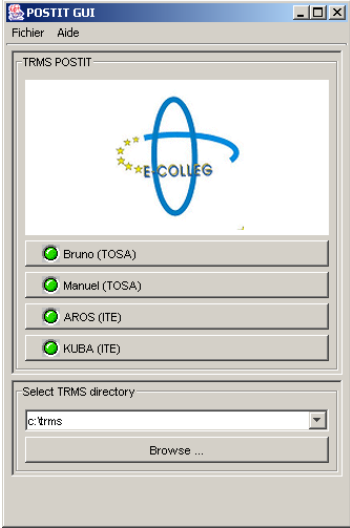


Figure 2. Postit GUI.

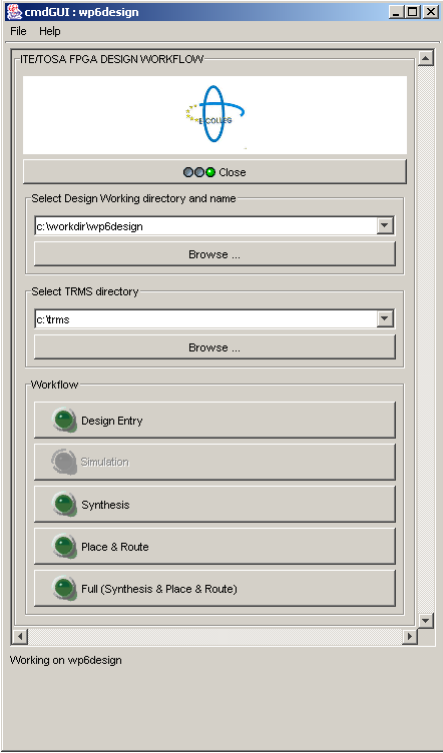


Figure 3. Command GUI.

3 Affine FPGA design

The Affine FPGA collaborative design was used for testing ACI. This complex design was motivated by a real need of TOSA. Thus, ACI verification was done in true industrial conditions. Affine FPGA constitutes a main part of a digital camera specified in TOSA for use in aircrafts. The design performs image processing in real time. It includes some components (Fig.4). Flash Memory Interface (F2) and Affine Controller VHDL (F3 & F4) models were created and tested in ITE. The first component was used in testing of the TRMS script files before the command GUI was developed and the next ones in testing of the TOSA GUIs. The whole design was integrated into FPGA (XC2V2000-5FF896) and tested in TOSA.

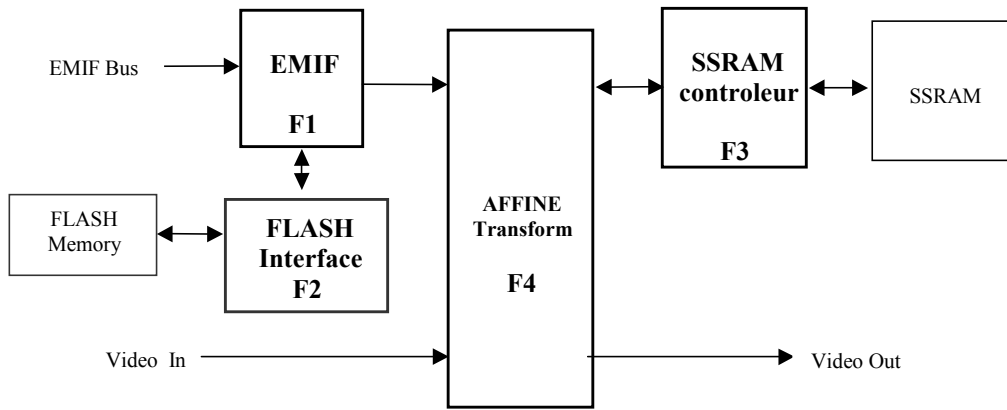


Figure 4. Schematic of Affine FPGA.

3.1 Flash Memory Interface

The flash memory interface is a part of the digital camera design. The last utilizes non-volatile AMD am29lv160B FLASH memory with built-in controller. This memory can operate in many modes, which results in a complicated protocol. The main role of the designed interface is to simplify the memory access protocol by implementing only these modes, which are utilized by the digital camera. The code of command which has to be executed in a memory is set on the `cmd [3 : 0]` bus and next, the positive pulse of the `start` signal is asserted. The circuit executes a sequence of writing to or reading from the memory controller registers. The latencies of writing and reading which are in a range of several hundreds system clock periods are defined by two separate VHDL *generics*. The polling technique is used to determine the end of embedded algorithm execution. The interface cooperates with the memory in one byte or 16 –bits wide data bus dependently of the value of one VHDL *generic*. Constants, which define *generic* values and a limited instruction set, are declared in the package `inter_pack`, developed in VHDL for configuration of the interface component.

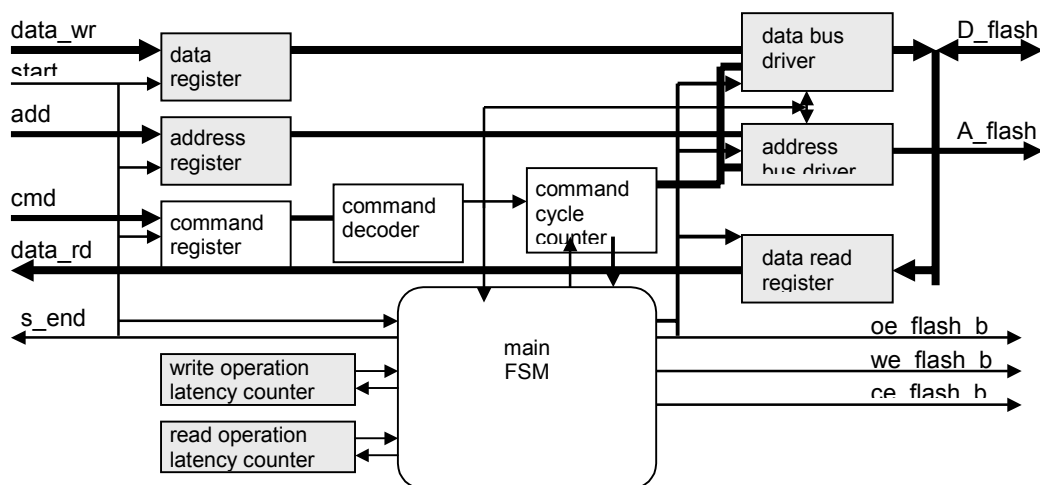


Figure 5. Schematic view of the interface.

A structure of the interface is schematically shown in Figure 5. It consists of the main FSM, which is responsible for providing the proper sequence of writing or reading operations for a given command and also for satisfying timing requirements of each operation. It cooperates with three counters: for counting write and read operation latencies and third one for counting the write operations. This last counter is loaded with value depending on the command to be executed (2-6), and successively decremented after performing each write operation. Data to be written to the flash and the proper address are set by the data bus driver and address bus driver respectively, based on the current command and write cycle number. Data read from flash is captured in a separate register. The component includes ~ 1000 equivalent gates.

3.2 Affine controller

Affine_controller performs image processing in real time. It cooperates with two SSRAMs, writing data to the first one and reading data from the other one alternately. It controls transfer of video data from a VIDEO IN block to one of SSRAMs concurrently with sending video data to a VIDEO OUT block based on data previously written to the second SSRAM. Video data are sent in data packages, related to even and odd lines and grouped into two frames.

The process of writing data is very simple, but the reading process is much more complex. Each output data value is calculated using interpolation of four stored data after performing affine transformation. The purpose of affine transform is to provide a real time video processing allowing image rotation, zoom and translation. The transform calculates, in fixed point arithmetic, an input pixel location for every output pixel location based on coefficients input from external component. Resulted location integer and fraction parts are applied in calculation of the pixel value based on bilinear interpolation of four neighbouring pixel values.

Affine controller is a hierarchical block. Six components (Fig.6) are included inside it. All components are configured in respect to many parameters, e.g. pixel number (related line & column numbers), accuracy of data and coefficients (related vector widths), SSRAM size (data & address widths). The values of all related constants are defined in the developed configuration package `affine_pack`, which contains also declarations of all used FSM states types and definition of used procedure `Set_addr` - translation of pixel coordinates to SSRAM address.

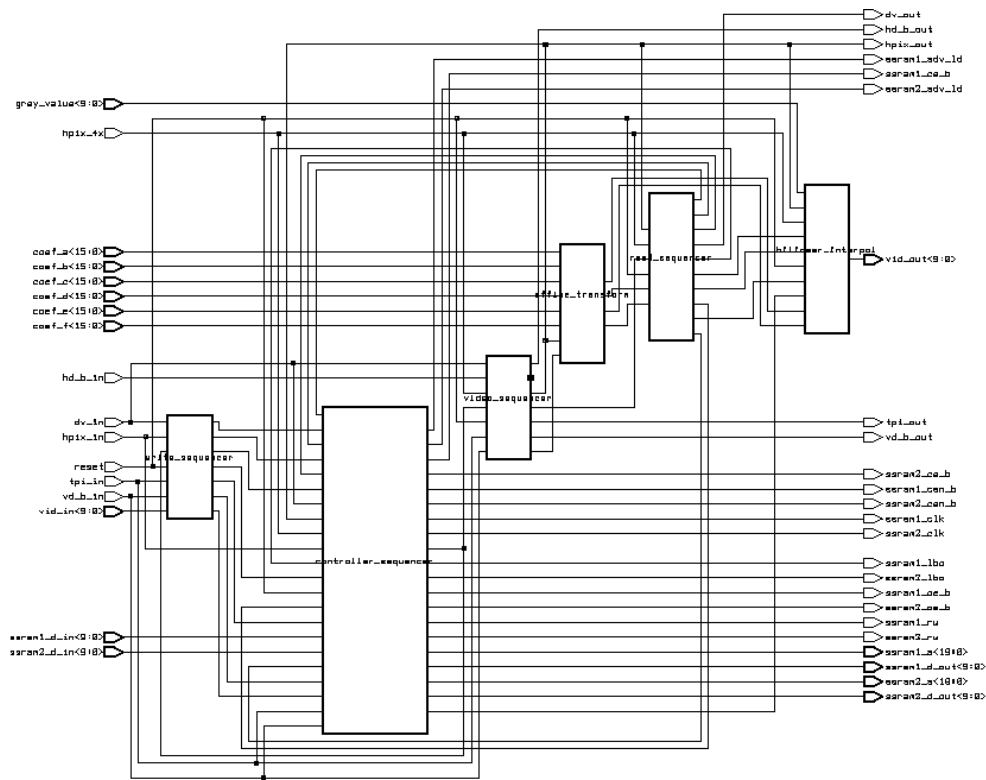


Figure 6. Architecture of Affine_controller.

The controller contains the following components:

1. controller_sequencer – the main sequencer, which 6-states Finite State Machine (FSM) controls start of others sequencers with cntr_en signal, selects alternately one SSRAM to be written and the other to be read. It drives the both SSRAMs with data obtained from write and read sequencers respectively.
2. write_sequencer – a component, which 9-states FSM controls writing video data to SSRAM by generation of needed control signals.
3. read_sequencer – a component which 12-states FSM controls reading video data stored in SSRAM by generation of needed control signals. It generates address signal using Set_addr procedure basing on pixel location internal part, calculated in affine_transform component. It generates also signals used in bilinear_interpol component.
4. video_sequencer – a component which 24-states FSM controls generation of control signals for VIDEO OUT block and other internal components.
5. affine_transform – a component, which performs affine transformation of every pixel location. It gives as results integer and fractional parts for every pixel location, basing on coefficients being input for every new image. The integer parts are used for SSRAM address calculation in read_sequencer component and the fraction parts are used in bilinear_interpol component.
6. bilinear_interpol – a component, which performs the interpolation basing on data values read from SSRAM or grey_value, if an address being result of affine transform is illegal, what is signalled by out_of_range signal. The interpolated data is output to VIDEO OUT block as the signals from video_sequencer component.

The developed models were verified locally using VHDL simulation and obtained results were correct. Then whole affine controller was synthesised remotely in Xilinx Virtex FPGA library. The component includes ~ 6400 equivalent gates.

4 Application of remote tools

The simple collaborative design process was implemented into a workflow (cmdGUI) with some integrated tools. This allowed performing operations step by step either in ITE or at TOSA. In general, collaborative design integrates some number of partners and one of them plays a coordinator role, responsible for a final design. Every partner should have at his disposal a VHDL simulator, then simulation should be performed locally. There are a lot of VHDL simulators on the market and cost of some of them is not too high (even there are GNU-license based free simulators on the Linux platform), besides a graphical format of simulation results can't be sent by network in practice. Analysis of simulation results based only on the text format information, like snapshots of signal values, which can be easily interchanged by a network, is very uncomfortable in a real design. This is especially true in the early phase of a design process, when the design architecture is often modified. Designer has to view waveforms in a debugging phase of a new VHDL component design to verify its correctness. A waveform graphical format depends on the used simulation tool and designer has to have this tool at his disposal in order to view the result, because there are no universal tools that support different formats.

Alternatively the simulation result can be written out in a VCD (Value Change Dump) format (at one site), which can be read in by another simulation tool or waveform viewer at a different site. But the main problem of such approach is that the VCD file for a long simulation is typically quite large, which will result in its slow transfer via the network. In case of using only the text format snapshots of signal values for a given simulation time the debugging process will be much more time consuming, because the designer should create firstly test bench generating text simulation results and then every small modification would result in data transfer in both sides (VHDL model to a tool server and simulation result resend to the client server). Text simulation results are usually generated for a final verified version of a component to be compared with simulation results of a related post-synthesis structure.

A synthesis tool cost is much higher than a simulator one. It is thus reasonable to share the tool (in accordance with the license agreement) in a network of partners that not necessarily have the respective tool. They can perform synthesis of a developed component on a remote machine to check if the code is synthesisable. Data transfer in both sides can be restricted to sending a VHDL model and a tool script and resending synthesis reports with a structural model related to obtained net-list. Simulation of the post-synthesis netlist must be performed remotely, because simulation models of all standard cells from used standard cell library are connected with Design Kit located at the synthesis tool owner place. But this component validation phase by simulation should give the text format to easy compare with final simulation result of the VHDL model.

Collaboration between design partners takes place, the most frequently, at component development phase - particular components are developed by designers residing in different places. The phase of integration all developed components into one complex project is performed by coordinator. More advanced test benches that allow generation of simulation results in the text format are used for verification of the final design including a lot of previously validated components. This phase is done in a coordinator place locally, similarly as synthesis of whole design and its FPGA implementation

5 Conclusions

The paper has presented a simple example of the collaborative design process which used the cmd-GUI and which was performed successfully, only with some problems related to simulation. But in fact, but the simulation phase should be performed locally. The experiments have revealed one general disadvantage of AC I, namely a very large amount (several hundreds per minute) of messages related to node polling that are sent from the ANTS server by the Internet, often attracting network administrator's attention, as darkening global data transfer and possible attacks.

Application of ACI with GUIs, discussed above, needed creation of great number of *java* scripts and *xml* descriptions dedicated to every task performed by the available tool on any server. Preparation of the environment which supports collaborative design process was very time consuming and error prone and every modification related to applied tools (e.g., in case of upgrade), options or users required creation of new files.

TRMS-GTLS seems in this context to be the most advanced environment [7]. The well integrated GUI with the wide range of background services is the real advantage of the TRMS-GTLS. There is no need to write scripts, all activities can be done from GUI. It is open to integrate any tool from any site. It has integrated a lot of security mechanisms, which offer very good authentication of users and authorisation of performed tasks. This advanced environment is close to designer needs and ITE and TOSA. Further activities are foreseen in respect to testing and further deployment of ACI in the near future.

Acknowledgments

The work described herein was funded by the EU IST project E-Colleg (IST-1999-11746) contract.

References

- [1] Kokoszka A., Siekierska K., Nguyen Q.T.: Are workflow management system useful for collaborative engineering?, Proc. of the 4th European Conference on Product and Process Modelling in the Building and Related Industries, Portoroz, Slovenia, 9-11 Sept.2002, pp. 245-251
- [2] MQSeries Family Platforms <http://www-4.ibm.com/software/ts/mqseries/platforms/>
- [3] C-Lab: ASTAI® Manual v.2.2, Paderborn, 2000, <http://www.c-lab.de/>
- [4] DSE - Distributed System Engineering (IST-1999-10302, <http://cec.to.alespazio.it/DSE>)
- [5] ISTforCE - Intelligent Services and Tools for Concurrent Engineering (IST-1999-11508, <http://www.istforce.com/>)
- [6] TOCEE - Concurrent Engineering in IS (ESPRIT Project 20587, 1996-1998, <http://cib.bau.tu-dresden.de/tocee/>)
- [7] Chan F., Spiller M.: The Design of an Internet-Based Collaborative System Architecture. Univ. of Berkeley, Computer Science Dept. Report Nb. 269.
- [8] Kokoszka A., Ługowski N., Nguyen Q.T., Obrębski D., Pawlak A., Siekierska K., Carballeda M., Schlichter B.: Collaborative Design of the FWMI (FPGA Pulse Width Modulator Industrialised Version), Proc. of E-Colleg Workshop on Challenges in Collaborative Engineering, CCE'03, Poznań, Poland, April 15-16, 2003, pp.115-124
- [9] Tim Sschatkowsky, Wolfgang Mueller: Distributed Engineering Environment for the Design of Electronic Systems, Proc. of E-Colleg Workshop on Challenges in Collaborative Engineering, CCE'03, Poznań, Poland, April 15-16, 2003, pp. 25-31
- [10] E-Colleg project web page <http://www.ecolleg.org/>
- [11] T.Kostienko, A.Pawlak, M.Ślęzak, P.Fraś, J.Magiera, M.Witczyński: Development of TRMS/GTLS – Global Tool Lookup Services, Proc. of E-Colleg Workshop on Challenges in Collaborative Engineering, CCE'03, Poznań, Poland, April 15-16, 2003, pp. 32-41