IMEM: A C++ High-Level Synthesis Tool for FPGA based Real-Time Video Processing Systems

Najeem Lawal, Benny Thörnberg, Mattias O'Nils

mattias.onils@miun.se

ITM - Mid Sweden University – Sundsvall, Sweden

http://www.miun.se/itm

Abstract

FPGA based video processing systems are effective but have a complex programming model. The synthesis tool presented here reduces the design complexity and still produces effective implementations.

1. Introduction

This project deals with optimizing FPGA embedded memory required for data buffering in the pre-processing stage of real-time video processing systems (RTVPS). The pre-processing stages are usually neighbourhood oriented. Examples of such 2-D operations are convolution, histogram, spatial and gray-level transforms, erosion, dilation and component labelling [1].

Current approaches to C/C++ based system synthesis or any other synthesis environment do not efficiently make use of the FPGA architecture especially the memory sub-systems for real-time video processing systems. This is due to the manner in which memories are currently being instantiated in FPGAs. In this project, we are developing a system synthesis tool for implementing RTVPS with multiple neighbourhood oriented filters targeting FPGAs. The tool takes advantage of our already developed memory modelling tool IMEM [2], memory allocation [3], boundary conditions management tool [4] and behavioural simulation platform. Within memory modelling, processes such as memory estimation and optimization are carried out. The synthesis process explicitly separates the modelling and implementation of memory requirements and behaviour of the filter functions. In this manner, the memory requirements of the RTVPS will be implemented by the tools developed in this project whereas tools like Agility Compiler can be used to implement the behaviour of the RTVPS filters. The approach supports functional verification through simulation of both the C/C++ and VHDL modules of the filters.

2. Conceptual model – IMEM

The methodology used is to capture video system using a coarse grained synchronous dataflow graph called IMEM, (Fig. 1A). IMEM stands for Interface and MEmory Model and is build on top of the SystemC modelling library [2]. Each node in the IMEM dataflow graph captures both the abstract video interface and the memory model as shown in Fig. 1B whereas each edge in the graph represents the data width of each pixel in the video frame. The model is stated to be conceptual since it explicitly captures the data dependencies. The memory model is a description of the neighbourhood of pixels that the task operates on. Additionally, each node consists of a description of the task's functional behaviour. The task does not include any data dependency or timing related to the dataflow, just an un-timed C++ description of the relation between input and output pixels.

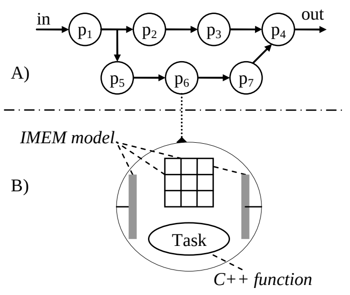

Figure 1: IMEM model of a video processing system.

The architecture in Fig. 2 handles data storage and boundary conditions for the spatial pixel neighbourhood. In Fig. 2, the task is connected to the memory architecture through the port interfaces for all the required pixels data and its output pixel corresponding to the centre pixel in the input neighbourhood. The sliding window controller *SLWC* monitors the centre pixel in a spatial neighbourhood and using the position information provides valid data for all the pixels in the spatial neighbourhood through the *Line buffers*, *Window ctrl* and *Pixel Switch*. The *Line buffers* are required to buffer image data in order to create the spatial neighbourhood. They are implemented in hardware through the memory architecture described in [3]. The architecture groups all the line buffers required by a task to form a global memory object (GMO) for that task. Where the width of the GMO is the number of line buffers required by the task multiplied by the bit width representing a pixel. The length of the GMO is equal to those of the line buffers that formed it [3]. Allocation of the GMOs has been formalised through integer linear programming (ILP) and implemented heuristics in order to achieve optimal results.

Window control (*Window ctrl*) provides control signals used by the *Pixel switch* to build a spatial neighbourhood around the current pixel. *Window ctrl* is implemented in hardware such that only one copy is instantiated and used to control all *Pixel Switch* modules instantiated for all the spatial neighbourhoods in a VIP algorithm involving more than one frames. The *Pixel switch* replaces all pixels in a spatial neighbourhood affected by boundary condition using predefined default values if the centre pixel is at the image boundary. The *output sync* is optional and is required to realign the pixels with other video signals where time synchronized data and control signal outputs are expected. This is because the neighbourhood's output pixel usually has a latency with respect to the input video control signals by an amount depending of the neighbourhood size and the number of pipeline stages.

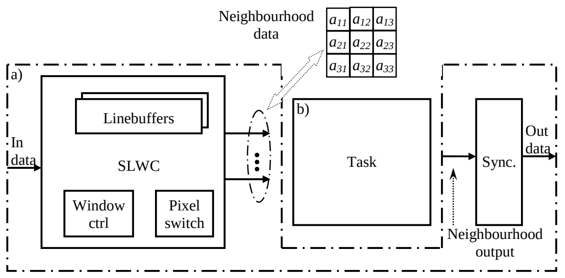

Figure 2: Boundary conditions implementation architecture.

3. System synthesis

The IMEM synthesis workflow depicted in Fig. 3 demonstrates how our research on modelling and high level synthesis fits into an implementation trajectory. This workflow is defined at six different levels along the left-hand axis. The video-processing algorithm is developed and simulated using IMEM at level 1. This executable model can then be verified through functional simulation. Data dependency information, frame sizes, composition of the 3-dimensional neighbourhoods and colour space models are exported into an interface and memory model at level 2. Hence at it is at this level that the memory requirements of a RTVPS are separated from the behavioural C++ description of the RTVPS filters (as shown in Fig. 2B). The interface between the memory and filters of each operator is also defined at this level. The model exported in level 2 is the input to the memory synthesis process at level 3. This is where memory estimation, memory hierarchy optimization [5], memory allocation [3] and address generation is performed. All these processes have been implemented using ILP and heuristics to achieve optimal results. We have also explored the impact of pixel bit-width through buffer re-timing and placement between tasks on the overall memory usage.

At level 3, the SystemC functional description together with the interface template generated from the memory model are synthesized using a SystemC based commercial high-level synthesis tool, in this paper Agility from Celoxica. The VHDL code from both the functional part and the optimized interface and memory model is integrated at level 4 and synthesized at level 5. Hence the components separated at level 2 are integrated at level 5. Hardware simulation and compilation are also carried out.

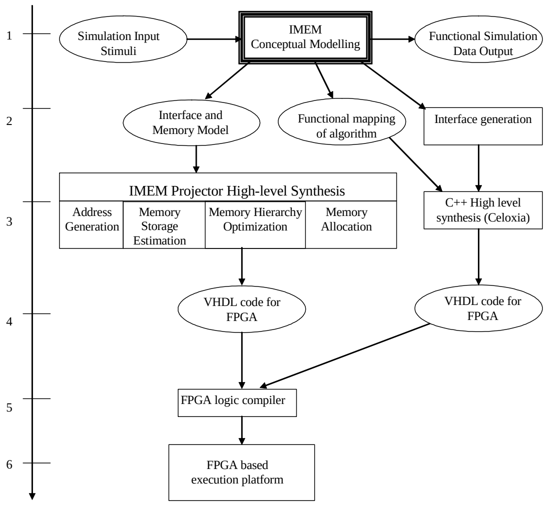
1
2
3
4
5
6
Simulation Input Stimuli
IMEM Conceptual Modelling
Functional Simulation Data Output
Interface and Memory Model
Functional mapping of algorithm
Interface generation
IMEM Projector High-level Synthesis
Address Generation
Memory Storage Estimation
Memory Hierarchy Optimization
Memory Allocation
C++ High level synthesis (Celoxia)
VHDL code for FPGA
VHDL code for FPGA
FPGA logic compiler
FPGA based execution platform

Figure 3: System synthesis workflow

4. Conclusion

In this project, we have developed a design tool based on the IMEM model. The tool manages memory requirements of video processing systems, thus reliving designers of the complexities associated with memory sub-systems. The tool provides designers with a user-friendly FPGA programming model while generating efficient hardware implementations of the memory sub-systems.

5. References

[1] Gonzales, R. C. and Woods, R. E., Digital Image Processing. Addison Wesley (1993).
[2] B. Thörnberg, H. Norell and Mattias O'Nils, "IMEM: an object-oriented memory- and interface modelling approach for real-time video processing systems", Proc. of the Forum on Specification & Design Languages, Sept. 2002.
[3] N. Lawal, Thörnberg, B., O'Nils, M. and Norell, H., "Global Block RAM Allocation Algorithm For FPGA Implementation Of Real-Time Video Processing Systems", Journal on Circuits Systems & Comp., Vol. 15, No. 5, 2006.
[4] H. Norell, N. Lawal and M. O'Nils, "Automatic Generation of Spatial and Temporal Memory Architectures for Embedded Video Processing Systems", European Association for Signal and Image Processing (EURASIP) Journal on Embedded Systems, 2006.
[5] B. Thörnberg, et al, "Bit-Width Constrained Memory Hierarchy Optimization for Real-Time Video Systems", IEEE Trans. on CAD of Integrated Circuits And Systems.

Acknowledgement

The authors would like to thank the Mid Sweden University and the Swedish Knowledge Foundation for their financial supports.