Enabling MPSoC design space exploration on FPGAs

Enabling MPSoC Design Space Exploration on FPGAs

Ahsan Shabbir, Akash Kumar, Bart Mesman, and Henk Corporaal

Eindhoven University of Technology,
5600MB Eindhoven, The Netherlands
{a.shabbir, a.kumar, b.mesman, h.corporaal}@tue.nl
http://www.es.ele.tue.nl/

Abstract. Future applications for embedded systems demand chip multiprocessor designs to meet real-time deadlines. These multiprocessors are increasingly becoming heterogeneous for reasons of cost and power. Design space exploration (DSE) of application mapping becomes a major design decision in such systems. The time spent in DSE becomes even greater with multiple applications executing concurrently. Methods have been proposed to automate generation of multiprocessor designs and prototype them on FPGAs. However, only few are able to support heterogeneous platforms. This is because heterogeneous processors require different types of inter-processor communication interfaces. So when we choose a different processor for a particular task, the communication infrastructure of the processor also has to change. In this paper, we present a module that integrates in a multiprocessor design generation flow and allows heterogeneous platform generation. This module is area efficient and fast. The DSE shows that up to 31% FPGA area can be saved when heterogeneous design is used as compared to a homogeneous platform. Moreover, the performance of the application also improves significantly.

Keywords: FSL, FPGAs, FIFO, MPSoC.

1 Introduction

The overall execution time of an application mapped onto an architecture depends on a number of factors such as memory hierarchy, communication structure etc, however type of processor remains a key contributor. For example, any signal processing application will run faster on a DSP, whereas any control dominated application will not be able to exploit the resources of such processors effectively. Performance can be enhanced if different parts of the application run on different processors which are optimized for those characteristics. Heterogeneous multiprocessor platforms [1] are good candidate for these type of applications.

1.1 Synchronous Data Flow Graphs

Synchronous Data Flow Graphs [2](SDFG) are used to model Digital Signal Processing (DSP) and Multimedia applications. Tasks (Actors) are vertices in

D.M.A. Hussain et al. (Eds.): IMTIC 2008, CCIS 20, pp. 412–421, 2008.

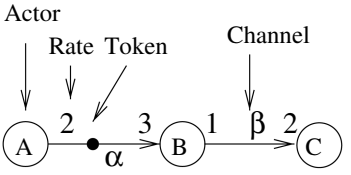

Fig. 1. An example of SDF Graph

the graph and the directed edges represent dependencies between the tasks. Tasks also need input data or control information before they can start and they usually produce output data; such information is referred to as *tokens*. Actor execution is also called *firing*. An actor is called *ready* when it has sufficient input tokens on all of its input edges and sufficient buffer space at its output edges;an actor can only fire, when it is ready. Figure 1 shows a simple SDFG consisting of three actors and two channels. Actor B can fire as soon as three tokens are available on channel *alpha*. Its firing results in consumption of three tokens from channel *alpha* and production of one token on channel *beta*.

1.2 Problem Description

DSP and multimedia applications are mapped on to multiprocessor platforms by using the SDF graphs [7]. Actors are mapped onto processors and communication between the actors is modeled as First in First Out Channels (FIFOs). FPGA vendors provide Platform [6] FPGAs. These Platform FPGAs are very suitable for prototyping multimedia applications. Heterogeneous platform generation on these FPGAs is difficult because each type of processor has its own communication interface. This slows down the design space exploration as both the processor and the communication infrastructure is changed at every design point. The situation becomes even more complex if accelerator attachment is also a possibility. In this paper we propose to have only one type of communication infrastructure for all types of processors in the MPSoC. We also propose to use the same interface for accelerator attachment. To show validity of our proposal we choose Virtex FPGA by Xilinx. These FPGAs contain up to four hard wired PowerPC-405 [4] cores. Xilinx also provides Microblaze [5] soft cores. Microblaze processors have a FIFO based communication link called Fast Simplex Link [13](FSL). Microblaze processors can be connected to each other through these FSLs. However the PowerPC processors do not have FSLs so we can not directly connect these processors with Microblaze processors. To enable rapid heterogeneous platform generation we have designed an interface, which connects with Processor Local Bus (PLB) of PowerPC and provides a standard FSL interface. We have included the interface into Multi-Applications Multi-Processor Synthesis (MAMPS) design methodology [3]. Our design flow takes in application(s) specifications and generates high level hardware description file (MHS file) for Xilinx FPGAs. This paper does not discuss the MAMPS flow, however interested readers are encouraged to read [3]. This paper describes the interface in detail and presents some results about its performance.

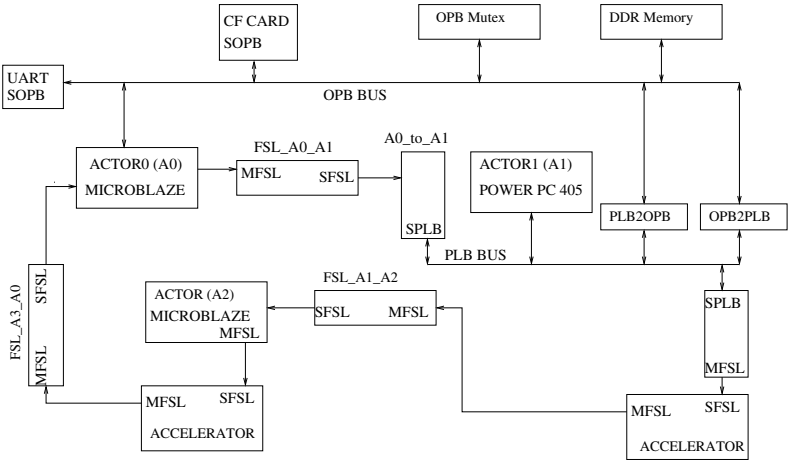

Fig. 2. Proposed Architecture

The proposed architecture is shown in figure 2. FSL is used as the basic communication infrastructure in the design. The architecture also supports Shared memory access. "Mutex" modules are used to access the shared sections of the memory.

The paper is organized as follows. Section 2 discusses some similar work. We explain implementation details in section 3. In section 4, we give some experimental results before concluding the paper in section 5.

2 Related Work

In ESPAM [8], a communication controller has been designed to interface with the instruction/data bus of Microblaze and PowerPC processors. ESPAM has two configurations for multiprocessor platforms. In the cross-bar based configuration, Every processor is connected with communication controller and the communication controller is further attached to the cross bar and communication memory. In the point to point configuration (Which is similar to our work), Every processor writes to its own communication memory through communication controller. Other processor in Kahn network [9], which has to use this data connects with this communication memory through its own communication controller. As the Communication Controller has been connected to processors using the processor address and data bus. The performance of Communication controller suffers due to this bus sharing [10]. On the other hand, we use the standard FSL bus for communication which means that Microblaze processor does not need additional hardware and has faster interface as compared to ESPAM. For the PowerPC processor, the Processor Local bus (PLB) is used to connect peripherals which provide master and slave interface to FSL channels.

So our design is area efficient and performs better due to standard interface on Microblaze side.

3 Implementation Details

FSL is directly integrated into the pipeline of microblaze processor so it is very efficient and fast interface. The bus contains FIFO buffers. Depth of these buffers is programable. FSL also has a control bit which shows that the location being read contains data or control information. FSL bus can be used synchronously or asynchronously. Every bus has only one Master and Slave so it is dedicated point to point bus. FSL Master and Slave interface signals are shown in figure 3. It is a unidirectional 32-bit bus. Hence to have bidirectional communication, a

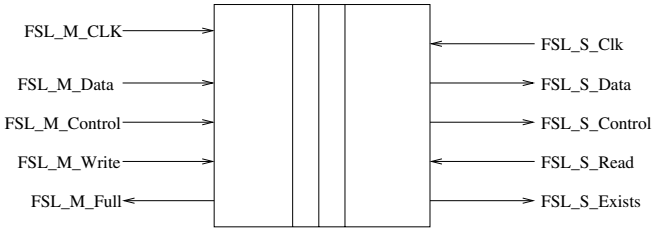

Fig. 3. FSL Interface signals

set of master and slave is required on each side as the master only sends data to FSL and Slave only receives. In our approach we have designed one slave PLB peripheral for each direction as shown in figure 4. By doing so we can have the flexibility of having any combination of FSL buses. For example we can have three buses sending data from processor "A" (PowerPC) to processor "B " (Microblaze) and two buses from processor "B" to processor "A". The peripheral, that reads FSL data is named as "Microblaze_to_PowerPC"', and the peripheral that sends data to FSL is named as "PowerPC_to_Microblaze". FIFOs inside the FSL are used in asynchronous mode. The reason for this choice is because PLB and FSL can have different clocks and if we use the FSL bus synchronously then we restrict our designs to use the same clock for PLB and FSL. So our design allows Microblaze and PowerPC processor to run at different frequencies and still communicate over FSL. Microblaze can run up to 100 MHZ, where as the operating frequency of PowerPC processor in Virtex FPGAs can be as high as 400MHz. This gives a large number of Task distribution options among the processors as the PowerPC running at higher frequency can take more of application load than a microblaze.

We implement our interface on Xilinx Virtex-II Pro 2VP30 FPGA, using the Xilinx Embedded Development Kit (EDK8.2i) [12]. The 2VP30 consists of 13,696 slices and up to 2,448 Kbits of on-chip BlockRAM memory. The FPGA contains two PowerPC-405 processors. We recommend to use ISCOM and DSCOM buses

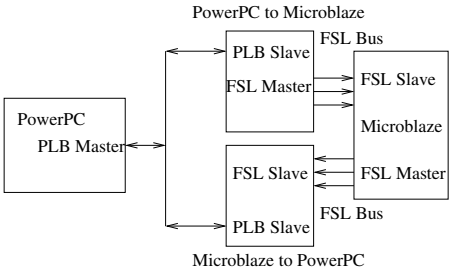

Fig. 4. Peripheral IPs are used to connect PowerPC with Microblaze

to connect instruction and data memories with the PowerPC processors as these are dedicated buses. On the other hand, if we use PLB bus for instruction and data along with our FSL interface peripheral, the bus contentions will drop the performance of the system. Sixteen FSLs can be attached to the PLB bus. On the PLB side two registers are designed which are used to read the status of FSL bus and also to enable the sending of a control bit along with the data if required.

The peripherals can be easily integrated into the designs by copying only a "pcores" directory. Software driver files are included in this directory. The whole design space of the application is explored in very short time by mapping different tasks of the application on to different processors and monitoring the execution time of the configuration. A case study of JPEG mapping is presented in the next section.

4 Application Mapping

We select JPEG as target application for our MPSOC platform validation. The JPEG Encoder application software is obtained from `http://www.opencores.org/people.cgi/info/quickwayne`. First we map the JPEG application on three Microblaze processors. These processors are connected to each other through FSL as shown in figure 6. The application is divided among three actors. These three actors are

1. File Parser (FP).
2. Color Covesion (CC) and Discrete Cosine Transform (DCT).
3. Variable Length Coding (VLC).

Figure 5 shows the "XML" file snippet used for this design. Three actors defined above are visible in the figure. Number of input/output tokens is also specified in the file. The token size can have the granularity of a macroblock or can be as small as a byte. In the "XML" file we also specify the type of processor as an "attribute". Design space exploration is performed by changing only the processor attributes of the actors. The MAMPS tool takes this file as input and generates the XPS [12] project files which are then synthesized and run to get the

```
<?xml version="1.0"?>
<sdfMapping version="1.0">
 <applicationGraph>
  <sdf name="g" type="G">
    <actor name="FileParser" type="a0">
     <port name="IN" type="in" rate="1"/>
     <port name="OUT" type="out" rate="1"/>
    </actor>
    <actor name="CCDCT" type="a1">
     <port name="IN" type="in" rate="1"/>
     <port name="OUT" type="out" rate="1"/>
    </actor>
    <actor name="VLC" type="a2">
      <port name="IN" type="in" rate="1"/>
      <port name="OUT" type="out" rate="1"/>
    </actor>
```

Fig. 5. Snippet of JPEG Encoder application specification

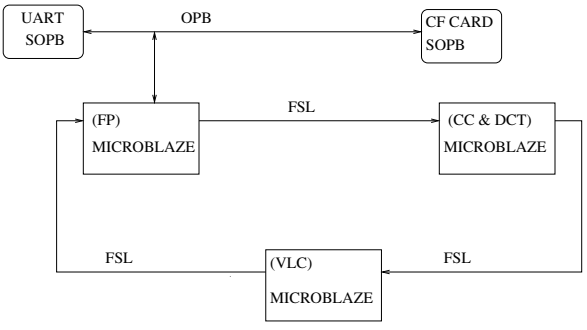

Fig. 6. Homogeneous platform, consisting of 3 microblaze processors,connected to each other through FSL

results. Various performance measuring timers are also configured in the XML file to measure actor execution times. Performance results at different design points are compared and best configuration is selected.

Figure 6 shows the homogeneous platform consisting of three microblaze processors. Compact Flash card (CF) and UART are connected to actor0 through on chip Peripheral Bus (OPB). UART is used for debugging and information display where as the CF card contains the input BMP file to be converted into JPEG format. First actor (actor0) opens the BMP file stored in the CF card, and sends the data to next actor (actor1). Actor1 converts the RGB format into YCrCb(4:2:0) format, computes the discrete cosine transform (DCT) and forwards the data to third processor (actor2). The final processor (actor2) performs VLC of the input, and sends the encoded data to actor0. Actor0 writes back the JPEG encoded stream received from Actor2 into the CF card.

For Heterogeneous platform generation we replace one microblaze processor with PowerPC processor as shown in figure 7. We first choose "Actor1" as a candidate for replacement with PowerPC processor and write its XML file. Two PLB slave peripherals are included in the design to have FSL interfaces. The peripheral "A0_to_A1" receives data from actor0 where as peripheral "A1_to_A2"

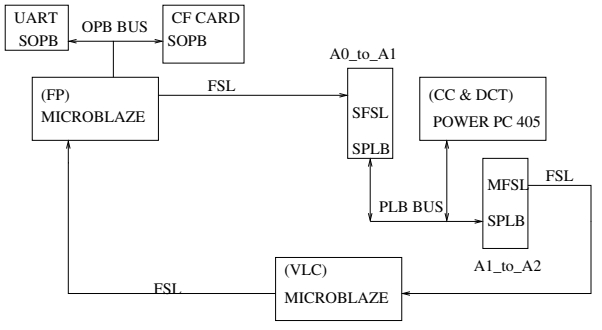

Fig. 7. Heterogeneous platform, Two microblaze and one PowerPC processor. PLB slave peripherals provides interface to connect PowerPC processor with the FSL buses.

sends data to actor2 as shown in figure 7. In both Homogeneous and Heterogeneous configurations, the cache of all processors is kept disabled. The input BMP file size is 983,094 bytes and we use Xilinx University Program (XUP) Virtex-2 Pro development board for our experiments.

Now we further investigate the accelerator attachment problem. We profiled the application to accelerate the tasks mapped to "actor1" in the previous configuration. Table 1 shows the profiling results of task "CC" and "DCT". Both microblaze and PowerPC are operating at 100MHZ. Task DCT uses 60% of the execution time and takes more cycles per call as compared to the Color Conversion task. So we decide to map the DCT on hardware. The area utilization results

Table 1. Profiling Results for Processors performing CC and DCT

Function	DCT (60% of execution time)	CC (40% of execution time)
PowerPC	19964 cycles/call	3478 cycles/call
Microblaze	20927 cycles/call	4425 cycles/call

of implementation are shown in table 2. It is clear that by replacing the microblaze with PowerPC processor, the occupied slices have reduced by 27% and 4 input LUT utilization has reduced by 31%. However, eight additional BRAMS are needed because the PowerPC instruction and data memory interfaces are 64bit wide and require more memory space than Microblaze processor. Columns four and five show the additional area required because of the DCT accelerator. Note that the same accelerator is used for both designs and no change is required in the interface.

Figure 8 shows the performance comparison for homogeneous and heterogeneous platforms. We call the first set of columns in figure 8 as the "software only configuration". Here both platforms are performing "CC" and "DCT" in

Table 2. Area utilization of both designs

Platform Type	Homogeneous	Heterogeneous	Homogeneous with DCT Accelerator	Heterogeneous with DCT Accelerator
Occupied Slices	5067 (36%)	3701 (27%)	6746 (49%)	5712 (41%)
4 Input LUTs	8589 (31%)	5881 (21%)	10787 (39%)	8406 (30%)
BRAM blocks	40 (29%)	48 (35%)	40 (29%)	48 (35%)

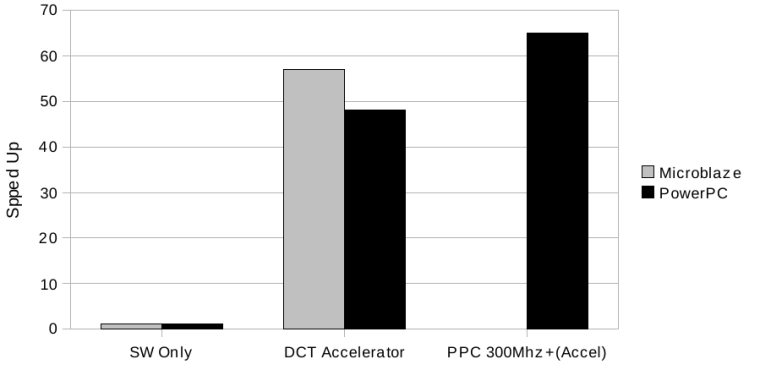

Fig. 8. DCT kernel Speed Up comparison for Microblaze and PowerPC processors

software. Table 1 shows the required number of execution cycles for both microblaze and PowerPC processors. Both processors utilize almost same number of execution cycles for both tasks. This is also the base-line performance and the results obtained by "accelerator based configurations" are compared with this configuration. The second set of columns shows the case when Hardware accelerators are used by both processors. Here the performance of homogeneous platform is is slightly better and a DCT kernel speed up of 57 is achieved. The Speed up by PowerPC-accelerator is only 48. This is a lower speed up as compared to microblaze based system. we share the PLB between two peripherals so the traffic on the PLB increases resulting in lower performance. However, as our interface supports asynchronous clock operation we run the PowerPC processor at 300 MHz and the PLB at 150 MHz. Consequently, the DCT kernel speed up of 65 is achieved. Figure 9 shows the best configuration after going through all the design points in the design space.

Similar results are obtained when PowerPC processor is assigned the VLC actor. However the performance deteriorates by a factor 1.5, when we map function FP to PowerPC as compared to the case when we map DCT and CC to PowerPC. This is because of the fact that file parsing is a slow function and involves low speed interface to OPB. The PowerPC connects to OPB through PLB2OPB bridge. This bridge is responsible for loss in performance as the microblaze connects with the OPB directly. These results suggest that most

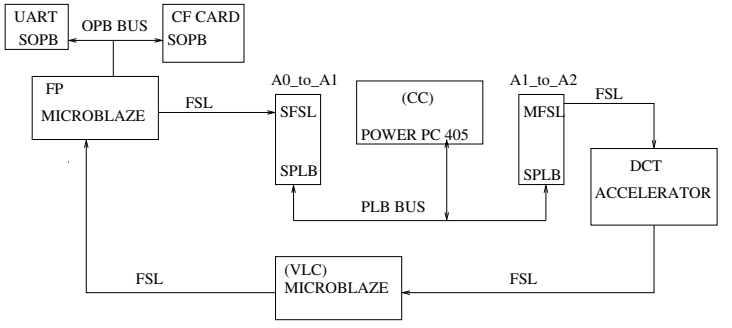

Fig. 9. Heterogeneous platform, Two microblaze and one PowerPC processor. PowerPC only performs color conversion and task DCT is mapeded onto hardware accelerator.

performance benefits from the PowerPC can be obtained by mapping the most computation intensive functions to PowerPC and relatively slow I/O functions to Microblaze processors. The designed interface helps in rapid design space exploration of the application mapping and it took only 3 hours to find the most appropriate application mapping.

5 Conclusion

In this paper, we present an interface, which enables quick DSE of multimedia applications on Virtex FPGAs. The interface is very easy to use and provides up to 31% savings in hardware resources for the same design mapped to Microblaze based platforms. We use JPEG encoder as a case study and explore the whole design space. We also propose to use FSL as standard communication architecture in the design. We observe that due to sharing of the interface at the PowerPC side, some performance is lost but it can be regained if we run the PowerPC processors at higher frequency. DCT Kernel speed up of 65 is achieved by operating the PowerPC processor at 300 MHZ. The interface has been integrated in MAMPS design flow and Platform configuration can be changed very quickly by only modifying an XML file. Our architecture also supports accelerators with standard FSL interface. These accelerators can be easily included in the design flow and their impact on performance can be observed.

Acknowledgments. First author is thankful to National Engineering and Scientific Commission Islamabad, Pakistan for their financial support.

References

1. Enslow Jr., P.H.: Multiprocessor Organization – A Survey. ACM Computing Surveys 9(1), 103–129 (1977)
2. Lee, E.A., Messerschmitt, D.G.: Statis scheduling of synchronous dataflow programs for digital signal processing. IEEE Transactions on Computers (1987)

3. Kumar, A., Fernando, S., Ha, Y., Mesman, B., Corporaal, H.: Multi-processor System-level Synthesis for Multiple Applications on Platform FPGA. In: Proceedings of Field Programmable Logic (FPL) Conference, Amsterdam, The Netherlands, pp. 92–97 (2007)
4. PowerPC Processor Reference Guide. Xilinx Inc. (2007)
5. Microblaze Processor Reference Guide. Xilinx Inc. (2007)
6. `www.xilinx.com.www.altera.com`
7. Sriram, S., Bhattacharyya, S.S. (eds.): Embedded Mutiprocessors: Scheduling and Synchronization, Marcel Dekker (2000)
8. Nikolov, H., Stefancov, T., Deprettere, E.: Multi-processor System Design with ESPAM. In: Proc. 4th IEEE/ACM/IFIP Int. Conference on HW/SW Codesign and System Synthesis (CODES-ISSS 2006), Madrid, Spain, pp. 323–328 (2006)
9. Kahn, G.: The Semantics of a Simple Language for Parallel Language. In: Proc. of IFIP Congress (1974)
10. Thompson, M., Nikolov, H., Stefanov, T.: A Frame Work For Rapid System-Level Exploration, Synthesis, and Programming of Multimedia MP-Socs. In: Proc. 5th IEEE/ACM/IFIP Int. Conf. on HW/SW Codesign and System Synthesis (CODES-ISSS 2007), Salzburg, Austria (2007)
11. Wolf, W.: The Future of multiprocessor system on chip. In: Proceedings of 41st Annual Conference on Design Automation (DAC 2004), San Diego, California, pp. 681–685 (2004)
12. Embedded Systems Tools Guide. Xilinx Embedded Development Kit, EDK version 8.2i ed., Xilinx, Inc. (2004)
13. Fast Simplex Link Channel (FSL), Product Specification Xilinx (2004)