

Synthesis and Estimation of Memory Interfaces for FPGA-based Reconfigurable Computing Engines*

Joonseok Park and Pedro C. Diniz
University of Southern California/Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, California 90292
{joonseok, pedro}@isi.edu

1. INTRODUCTION

As the densities of current FPGA continue to grow it is now possible to generate System-On-a-Chip (SoC) designs where multiple computing cores are connected to various memory modules with customized topology with application specific memory access patterns. For example, Xilinx has recently introduced devices to which a paired down version of a PowerPC core can be mapped and connected to a set of internal memories.

Given the complexity and heterogeneity of these target architectures, it is very likely that high-level compilation tools will be required to perform a wide variety of high-level program transformations. In order to determine the impact of these transformations in the resulting designs in terms of area and speed these tools will have to make use of fast and accurate estimation data.

Unfortunately, most commercially available synthesis tools for FPGAs have mostly ignored system level issues when dealing with external memories. While some tools now incorporate internal RAM modules and the mapping of array variables to them, they have avoided all external memory interfacing issues and corresponding estimation of the appropriate interfaces. Typically, programmers must separately synthesize the datapath if they want to exploit the estimation capabilities of the tools and then integrate those designs with handcrafted memory interfaces.

In this paper we address the problem of synthesizing and estimating the area and speed of memory interfacing for Static RAM (SRAM) and Synchronous Dynamic RAM (SDRAM) with various latency parameters and access modes. We describe a set of synthesizable and programmable memory interfaces a compiler can use to

automatically generate the appropriate designs for mapping computations to FPGA-based architectures. Our preliminary results reveal that it is possible to accurately model the area and timing requirements using linear estimation functions.

An important aspect of our approach is the decoupling of the datapath design, or application specific core design, with the development and synthesis of the external memory interfaces. To this effect we have designed a decoupled memory control scheme that allows the development of the actual computation core using commercially available synthesis tools. In this work we have extended our previous work with the addition of a more generic memory controller capable of both SRAM with and without pipelined memory accesses, with SDRAM with and without page-mode memory accesses.

2. EXTERNAL MEMORY CONTROLLER

Figure 1 depicts the structure of the memory controller. The controller interfaces the external memory subsystems and a datapath. In this architecture the memory controller associates with each data port in the datapath a FIFO queue, and binds with this queue the notion of a data channel. Each data channel is also bound to an entry in the address generation unit (AGU) used to generate the consecutive addresses in the external memory that correspond to a stream of data. The contents of the entry of the AGU can be reloaded at run-time to allow multiple binds in time for the same data channels if needed. The architecture also has a controller module that performs the external memory signaling for all of the channels and feeds the data into the FIFO queues.

The architecture in figure 1 represents a compromise between modularity and performance and has been tested successfully in the context of a compiler that maps applications directly to FPGA [1]. An important feature of this design is that it decouples the synthesis of the datapath with the synthesis of the memory controller and isolates the physical memory access protocol with the issues of behavioral synthesis – including scheduling of memory accesses. These features come at the price of an extra virtualization layer by requiring the datapath to execute an handshaking protocol to load and store data from external memories.

* Funded by the National Science Foundation (NSF) under grant number CCR-0209228

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
FCCM'03, April 08-11, 2003, Napa, California, USA.

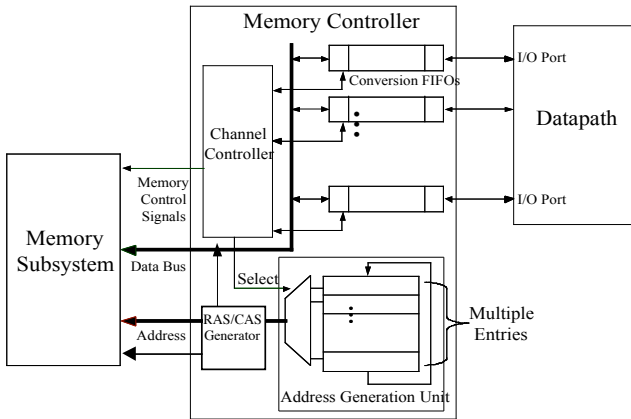


Figure 1. Memory Controller Architecture.

In figure 1 we have also shown a RAS/CAS generator block that can be thought as part of the address generation unit. This component takes an isolated address and determines, should SDRAM page mode be used, whether or not the corresponding memory access is within the same page of the previous memory access. If so, it informs the channel controller, which in turn bypasses some of its internal states to perform a faster in-page memory access mode. Internally this component consists of a simple table lookup much like a cache block.

The current implementation can interleave the memory accesses of multiple S-DRAM channels, and with minor modifications support both SRAM (in pipelined and non-pipelined) modes with the S-DRAM memory accesses.

We now describe the attributes of the memory interface controller for two basic memory technologies, the SRAM and SDRAM. In both cases we describe the parameters required for both non-burst and burst-mode operations, i.e., pipelining in the RAM cases and page mode access in the SDRAM case, respectively:

- **Type:** SRAM or SDRAM memory access signaling with the corresponding physical signals.
- **Channels:** Number of input and output channels which dictates the number of FIFO queues and the number of AGU entries.
- **Bit widths:** Whereas the memory interface word size is typically fixed, say at 32 bits, the bit width of the channels, i.e., the FIFO queues is programmable and application dependent.
- **Read/Write Latency:** Stipulates how many clock cycles for read/write cycles. This parameter also needs to be refined in the context of burst-mode operations.
- **Burst Mode and Burst Mode Latency:** Either pipelined or page-mode depending whether or not it is defined as an SRAM or SDRAM interface. The burst mode latency specifies the latency value for consecutive accesses, either pipelined or page mode.

We are developing simple library functions written in C that take these parameters and generate a complete synthesizable structural VHDL design, which has been successfully synthesized using XilinxTM tools.

3. ESTIMATION MODELING RESULTS

We have developed a multi-protocol memory interface unit as outlined in section 2 above in synthesizable VHDL specifications. Table 1 below illustrates the complexity and size implementation of these interfaces for the target VirtexTM XCV 1000 BG560 device using the Xilinx ISETM 4.1i toolset with logic synthesis tool. All of the designs in these experiments took less than 3 minutes to synthesize with a medium effort for place and routing on a 800 MHz Pentium III PC with 756 Mbytes of memory.

We report on the actual implementation results for both SRAM and SDRAM interfaces. For each of the interfaces we report on the implementation resources (FPGA slices – the VirtexTM device has a maximum capacity of 12,288 slices) for different channel bit width, i.e., FIFO queue bit widths. We also distinguish between pipelined and non-pipelined access implementations for the SRAM and page and non-page mode for the SDRAM implementation.

These results are for a single memory controller with 2 input and 1 output channels. In the next section we explore the sensitivity of the designs to larger number of channels.

Memory Interface	Transfer mode	Bit width	VHDL lines of code	FPGA Slices
SRAM	Non-pipe	8	1011	268 (2.2%)
		16	1005	235 (1.9%)
	Pipelined	8	1036	254 (2.0%)
		16	1033	226 (1.8%)
SDRAM	Non-page	8	1138	346 (2.0%)
		16	1129	263 (2.8%)
	Page	8	1169	299 (2.4%)
		16	1163	272 (2.2%)

Table 1. VHDL Code and Complexity Implementation.

Next we have used the generated VHDL from these libraries and simulated their performance. Table 2 below presents the performance and transfer rate for the various burst modes for each of the interfaces along with the base clock rate specification. For each of these performance results we derived the number of cycles per transfer for each of the transfer modes through simulation. The maximum clock rate was obtained by running the place-and-route passes of the synthesis tool for the target device. Also, in the experiments we have assumed a predefined latency value of 3 clock cycles to access the external pins of the device. Typically these parameters are variable but known for each implementation of the vendor library. In our case we have used the WildStarTM [5] library for these experiments.

Memory Interface	Transfer Mode	Bit Width	Clock Rate (MHz)	Cycles per byte	Transfer Rate (Mbps)
SRAM	Non-pipe	8	36.7	1.25	29.4
		16	47.9	1.25	38.3
	Pipelined	8	39.6	0.5	79.2
		16	47.7	0.5	95.4
SDRAM	Non-page	8	30.3	2	15.1
		16	33.5	2	16.7
	Page	8	32.0	1	32.0
		16	36.0	1	36.0

Table 2. VHDL Code and Complexity Implementation.

These preliminary results indicate that sizes of the interfaces are fairly small for the tested FPGA Virtex devices (less than 3%). In terms of the clocks rates, these designs attain a minimum clock rate of 30 MHz and a maximum transfer rate of 16 Mbps (8 bit channels) and 38Mbps (16 bit channels) for non-pipeline/page modes and 36 Mbps (8 bit channels) and 95 Mbps (16 bits channels).

As expected to clock rate results for the SDRAM page mode accesses are substantially slower than the non-page mode operations. We attribute this fact to the added complexity in the *hit-page* circuitry required to bypass the controller in the same page memory access operations.

We have successfully validated these designs (via simulation for the case of the SDRAM) and on a WildStar™ multi-FPGA board for the SRAM interface. Based on these results we build a simple linear regression estimation models for multi-channel memory interfaces for both types of RAM memories. The results in table 3 below illustrate the various design area plots for various number of channels settings.

Memory Interface	Transfer Mode	Bit Width	Number of Channels (in/out)	FPGA Slices (%)	Clock Rate (MHz)
SDRAM	Page	8	1/1	213 (1.7%)	30.9
		8	2/1	299 (2.4%)	32.0
		8	4/1	495 (4.0%)	28.2
		8	8/1	903 (7.3%)	25.3
		8	16/1	1463 (11.9%)	27.7

Table 3. Sensitivity to Number of Channels for S-DRAM with Page-mode Accesses for 8 bit Channels.

These results suggest, for this particular target architecture and place-and-route tool that the number of FPGA slices (area) and clock rate of a memory interface is given by the linear expression below obtained by linear regression and where N denotes the number of input channels.

$$\text{Area}(N) = 84.08 N + 153.33$$

$$\text{Clock}(N) = 30.49 - 0.27 N$$

While the area metric tracks a linear interpolation fairly well, the clock rate data reveals a more disperse pattern. We attribute this to the fact that these are small designs in the overall FPGA devices (see table 1 above). As such the area typically grows according to a linear function. The clock rate behavior, however, tends to be more sensitive to the vagaries of the place-and-route steps use in logic synthesis.

These results reveal several simple aspects about the proposed memory interfaces. First, their size grows less than linearly with respect to the number of memory access channels. Expectedly, there is a slight degradation of clock rate for larger designs. This degradation. However, is not severe (about 10%) which we expect to worsen for large designs where place-and-route is unable to easily find available FPGA slice resources. *Nevertheless, these results reveal that estimating the memory interface designs using simple linear function is a very good estimate of their overall area and timing characteristics.*

4. CONCLUSION

In this paper we have described a set of parameterizable memory interface designs for both SRAM and SDRAM memory technologies. The proposed designs present a set of abstractions a compilation and synthesis tools can use to automatically generate complete designs that interface with external memory modules. We have reported on the low area and fairly good timing for a wide variety of designs with pipelining and page-mode memory operations. The preliminary area and timing results for our designs reveal that it is possible to accurately model both area and timing of the proposed memory interfaces with linear functions. This estimation model will ultimately allow for tools to incorporate the memory interface component in their estimates for complete designs.

REFERENCES

- [1] P. Diniz, M.Hall, J. Park, B. So and H. Ziegler, "The DEFACTO System: Bridging the Gap between Compilation and Behavioral Synthesis", In Proc. of the 14th Workshop on Languages and Compilers for Parallel Computing (LCPC'01), to be published as Lecture Notes on Computer Science (LNCS 2624) Springer-Verlag, Berlin, 2003.
- [2] Monet™ User's Manual, Release R42, Mentor Graphics Inc., 1999.
- [3] J. Park and P. Diniz, "Synthesis of Pipelined Memory Access Controllers for Streamed Data Applications on FPGA-based Computing Engines", In Proc. of the IEEE Symp. On System Synthesis (ISSS'01), IEEE Computer Society Press, Oct. 2001.
- [4] H. Schmit and D. Thomas, "Synthesis of Applications-Specific Memory Designs," IEEE Transactions on VLSI Systems, 5(1), Mar. 1997, pp. 101-111.
- [5] WildStar™ Reference Manual Rev. 4.0, Annapolis MicroSystems Inc., 1999.
- [6] Xilinx, Inc. Virtex™ 2.5V FPGA Product Spec. DS003(v2.4), 2000.