

# ASIC Design of High Speed FSM based DDR4 SDRAM Controller for Fast Interface

Nitish Kumar Vishwakarma<sup>1</sup> Pankaj Rathi<sup>2</sup>

<sup>1</sup>M.Tech Scholar <sup>2</sup>Assistant Professor

<sup>1,2</sup>Department of Electronics and Communication

<sup>1,2</sup>Gyan Ganga Institute of Technology and Sciences, Jabalpur, MP, India

**Abstract**— The goal of this thesis is to explore the options for a predictable SDRAM controller for the FPGA platform. The variable SDRAM access latencies pose some challenges for its effective use in RTS, while the many-core FPGA platform creates a new context for rethinking the previous results and finding the new solutions for external memory access. The simple working prototype of the single-port SDRAM controller is implemented and integrated with the processor. Several options for multi-port arbitration are considered, and proposal is made for arbitration and interconnects in FPGA project. Xilinx ISE is been used for development of the SDRAM controller.

**Keywords:** SDRAM, DDR, FPGA, RTS

## I. INTRODUCTION

The Synchronous Dynamic Random Access Memory (SDRAM) is widely used external memory because of its attractive combination of high capacity, low cost and competitive performance. However variable SDRAM access latencies pose some challenges for its effective use in RTS. The goal of this thesis is to explore the options for predictable SDRAM controller for use in the FPGA platform. The FPGA project is an ongoing research project supported by the European Union’s 7th Framework Programmer, aiming to develop a homogeneous time- predictable multi-processor platform.

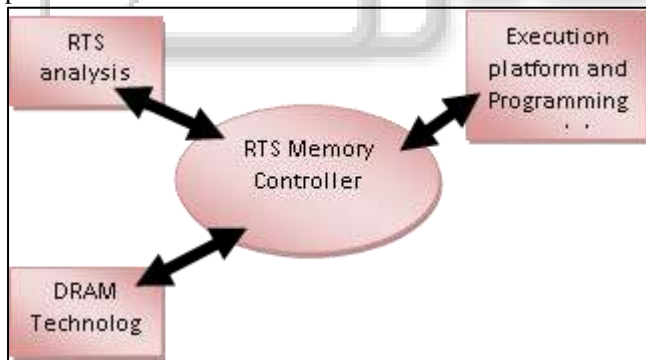


Fig. 1: The context of the memory controller for the Real-Time System

Figure 1 presents the context of RTS memory controller design. First of all the knowledge of DRAM technology is needed because it is the source of the limitations. Secondly, the controller should interact well with the analysis framework, because it derives the performance guaranties for the whole system. Finally, the good performance guaranties of a memory controller are only possible by tuning it to the execution platform and programming model. Only by understanding the interaction of these three domains of knowledge the efficient controller is possible.

## II. METHODOLOGY

The proposed SDRAM controller uses double frequency clock for SDRAM interface, and commences the data to/from requester on both edges of the slower clock. The controller is designed to be used from the external chip, because a single 32-bit input signal is used for address, data in, data out and part of the command encoding. The controller introduces 8 SDRAM cycles (4 system cycles) of additional data latency during read.

A controller is usually organized as the Single-port Controller which can be used by single requester only and the separate interconnect and arbitration layer which allows the controller to be shared across multiple requestors.

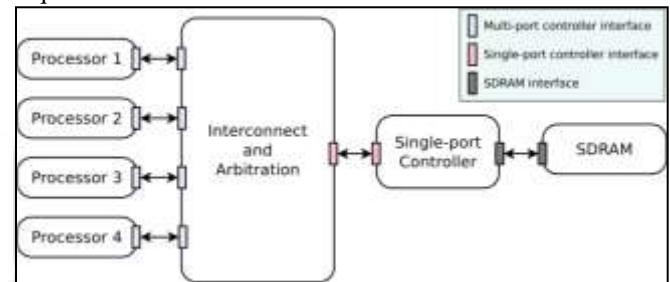


Fig. 2: Controller Organization

Same single-port controller interface for the multi-port controller to make the arbitration transparent, but it might be beneficial to optimize the multi-port interface. The single-port controller translates requestors’ (processors’ or DMA controllers’) memory accesses into legal sequences of SDRAM commands. The linear memory addresses provided to the controller are translated into tuple of rank/bank/row/column addresses. The controller must also keep track of the state of the SDRAM banks and ensure that all the SDRAM timing constraints are obeyed. Usually, the controller is also responsible for issuing SDRAM refresh operations. The designed are organized as 16-bit words in 4 banks of 8192 rows by 512 columns. The chips are speed grade 7, so according the specification they can run up to 143 MHz (with CL=3) or 100 MHz with CL=2. The general timing parameters of the SDRAM were explained in the Timing parameters section of the 3 chapter.

$$\text{Cycles Random Read} = \max(t_{RC}, \max(t_{RD} + t_{CAC} + BL \neq (t_{PQL} + 1), t_{RAS}) + t_{RP}) \quad (1)$$

The Write transaction would require: Recharge, Activate, Burst transfer, Write recovery cycles. The write recovery cycles are  $t_{DPL} \neq 1$ , because  $t_{DPL}$  contains the cycle of the Recharge operation. Again, the whole sum must be at least  $t_{RC}$ .

The SDR SDRAM memory targeted by the controller supports all the available options. The dedicated Auto-Recharge command needs the same amount of time as

regular pair of Activate and Recharge (tRC), so it does not impact the slot size for simple transactions. While for interleaved transactions the refresh should be performed manually by Activates interleaved in a same way. That is the refresh of the first bank can be overlapped with the previous transfer from the last bank etc. In case of interleaved burst of length two (for example Figure 3.4), the Recharge commands won't fit in to the transaction sequence. The write operation sequence with disabled data mask signal can be used to perform refresh in such case.

**SDRAM Initialization:** The controller has to perform the SDRAM initialization sequence:

- 1) Wait for power-up and CLK stable.
- 2) No operation for 200us.
- 3) Recharge all banks.

The sequence of 8 Auto-Refreshes cycles, with the usual timing requirements.

Configure the device by programming the Mode Registers.

In case of SDR SDRAM, the initialization sequence is simple and requires some additional states in the controller's state machine. Though some controllers delegate this responsibility to software operation.



Fig. 3: Eight word operations interleaved over four banks

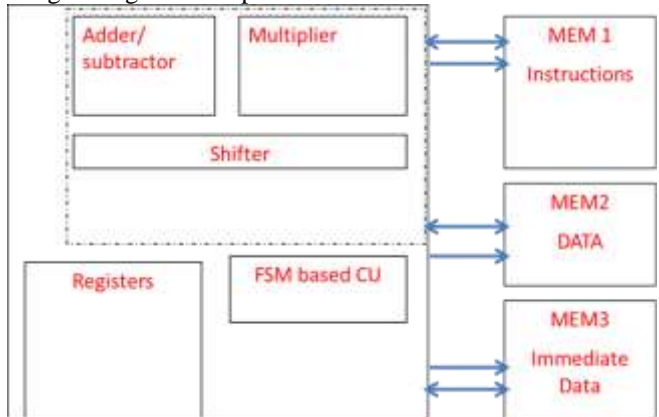


Fig. 4: Proposed architecture of SDRAM interfacing

The first deficiency can easily be solved, because controller keeps track of when the data will be available, and this information can be used for rdy\_cnt. Solving the second deficiency requires modification of the controller and/or extensions to Simpson interface. The Simpson supports pipelined transactions, and could be used for SDRAM if some signals are added to denote when the transaction corresponding to the same burst are finished. The controller could then use this information to start the bank recharge. In

addition the controller would need to be modified to support pipelined transactions.

Proposed FSM: CU is FSM based design shown in figure 4 below for maintaining the synchronization between all modules explains above. Proposed work also proposed a new RISC Instruction set for DSP application along with its OPCODE is been show in table 1. The FSM is been developed for the maintaining synchronization and generating control signals in synchronize manner in proposed work total 10 stage Mealy type FSM is been used table 1 below shows the state transaction.

Present state	Input signal	Next state
Idle	Rising edge of clock and enable = '1'	Fetch-1
Idle	Rising edge of clock and enable = '0'	Idle
Fetch-1	Rising edge of clock	Fetch-2
Fetch-2	Rising edge of clock	Decode-1
Decode-1	Rising edge of clock	Decode-2
Decode-2	Rising edge of clock	Readm
Readm	Rising edge of clock	Execute
Execute	Rising edge of clock	Wait1
Wait1	Rising edge of clock	Writem
Writem	Rising edge of clock and count <= 15	Readm
Writem	Rising edge of clock and count > 15	Adrinc
Adrinc	Rising edge of clock	Fetch-1

Table 1: The FSM state transaction

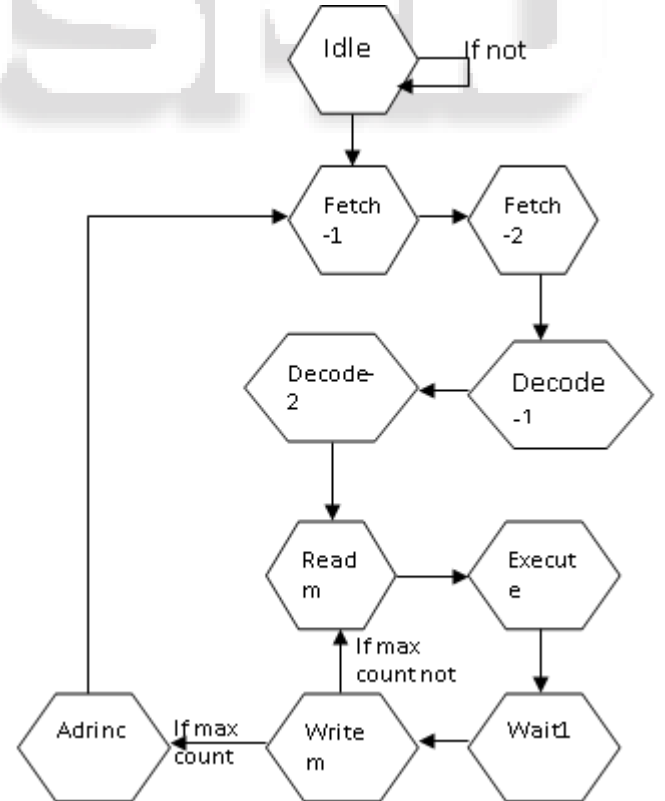


Fig. 5: The Finite State machine for Control Unit

- Idle: when reset the processor
- Fetch1 and fetch-2: when opcode fetched (fetch-1) and when data fetched (fetch-2)

- Decode1 and Decode-2: Identify the opcode (Decode-1) and set signal starting pointer for reading the data (Decode-2)
- Readm: to read the both of data from respective memory
- Execute: to perform respective operation
- Wait1: wait for the let the operation in ALU complete.
- Writem: write the result in respective memory
- Adrinc: the next instruction to be fetch from next address so it increases the address.

### III. RESULTS

Figure shown below shows the RTL (Register transfer Logic) schematic/ technological Schematic top of proposed module here:-

clk is the clock input signal  
rst is the reset input signal  
enb is the enable input signal  
dout1 and dout2 are of 16 bit each output data, dout2 MSB 16 bit when multiplication done and dout1 is the 16 bit LSB output  
addrw if 4 bit address of output memory  
rw1 and rw2 are the output of control for write operation  
The signal1, signal2 and signal3 are already inside the module in the form of memory and 32 bit instruction is also inside

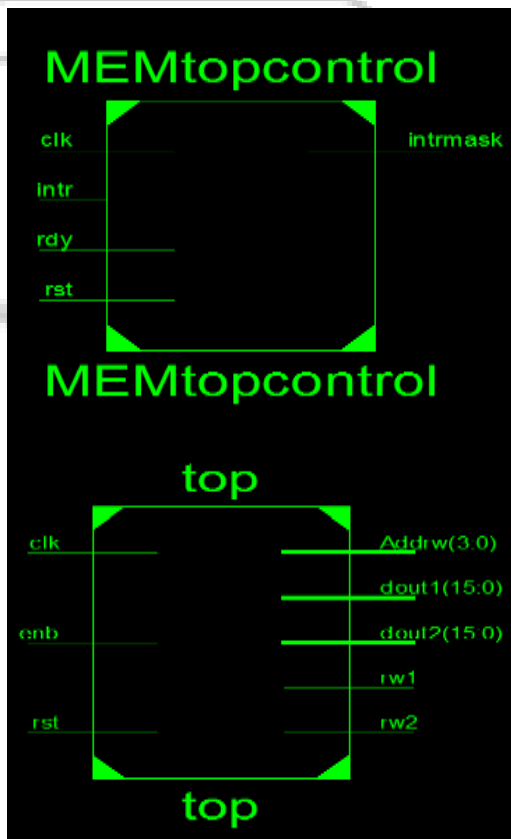


Fig. 6: RTL/Technological schematic TOP of SDRAM controller

In this part we look at the synthesis results of the controllers. In the first part we compare implementation of our controller described with some general purpose SDR SDRAM controllers. We will compare the FPGA synthesis results of our controller with 3 other designs. We first

describe each design in short and provide the results at the end of the section.

MEMtopcontrol Project Status (08/09/2017 - 08:18:24)			
Project File:	protop.vhdl	Parser Errors:	No Errors
Module Name:	MEMtopcontrol	Implementation State:	Synthesized
Target Device:	xc7s030-3FG324	Errors:	No Errors
Product Version:	ISE 12.2	Warnings:	372 Warnings (0 new)
Design Goal:	Balanced	Routing Results:	
Design Strategy:	Ultra Default (locked)	Timing Constraints:	
Environment:	System Settings	Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	93	19200	0%
Number of Slice LUTs	39	19200	0%
Number of Fully used LUT-FP pairs	35	97	36%
Number of bonded IOBs	4	200	1%
Number of Block RAM/FIFO	1	32	3%
Number of BRAM/BUFGCTRLs	3	32	9%

Fig. 7: the synthesis summary

Design	Fmax (MHz)	Slices	LUT
Xilinx	251 Mhz	97	72
Edgar Lakis et al [3]	202Mhz	101	
Satish reddy et al [2]		104	
Tian Jin et al [1]	349 Mhz		127
Ours	374 Mhz	93	74

Table 2: comparison of Synthesis results for evaluated SDR controllers.

The original design used full range integer types, resulting in the inferring of 32-bit counters. The comparison shows that the performance of our simple behavioral controller description is reasonable, and also that some speed gains are possible by simple optimization of critical path (see OurOptimized vs. our). We would like to make some comments to the obtained results:

### IV. CONCLUSION

The open source SDR SDRAM controller has been created. Its initial integration into two RTS platforms (FPGA and JOP) was performed and tested. The different options of memory access scheduling for the FPGA platform have been investigated. The analysis included estimates of their RTS efficiency and the hardware implementation feasibility. For hard-RTS, the round robin (RR) does not have advantages over time division. Multiplexing (TDM), whereas WCET bounds can be made tighter with TDM. The static priority (SP) arbiters like CCSP and PBS are not scalable for WCET analysis because the least priority requester will suffer from latency proportional to the total bandwidth allocation of other requesters. The memory access timing analysis performed at WCET level suffers from fundamental limitations in reducing memory bandwidth over-allocation. The local worst case required bandwidth has to be allocated for the whole task's execution period.

### REFERENCES

[1] Tian Jin, Wenxin Li, Xiangyu Hu, A Tow-Level Buffered SDRAM Controller, 2016 3rd International Conference on Information Science and Control Engineering, 978-1-5090-2534-3 /16 -2018 IEEE, DOI 10.1109/ICISCE.2016.37

- [2] satish reddy n, ganesh chokkakula, bhumarapu devendra, ASIC Implementation of High Speed Pipelined DDR SDRAM Controller, ICICES2014 - S.A.Engineering College, Chennai, Tamil Nadu, India, ISBN No.978-1-4799-3834-6/14/2018 IEEE
- [3] Edgar Lakis, Martin Schoeber, An SDRAM Controller for Real-Time Systems, In Proc. IEEE International Workshop on Application of Reliable Computing and Communication, pages 29–34, Dec. 2018.
- [4] Deepali S h a r m a, S hruti bhargava, M a h e n d r a V u c h a, Design and VLSI Implementation of DDR SDRAM Controller for High Speed Applications, Deepali Sharma et al, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (4) , 2011, 1625-1632
- [5] D. Vanden Bout,, application note on XSA SDRAM Controller, September 5, 2002 (Version 1.1)
- [6] Benny Åkesson, An introduction to SDRAM and memory controllers, Philips
- [7] Shabana Aqueel and Kavita Khare, Design and FPGA Implementation of DDR3 SDRAM Controller for High Performance, International Journal of Computer Science & Information Technology (IJCSIT) Vol 3, No 4, August 2011, DOI : 10.5121/ijcsit.2011.3408 101
- [8] Altera. SDRAM Controller Core, Quartus II Handbook Version 9.1 Volume 5: Embedded Peripherals, v9.1 edition, November 2009.
- [9] Altera Corporation. SDR SDRAM Controller White Paper, ver. 1.1 edition, August 2002

