# DESIGN CHOICES OF MULTI-PORTED MEMORY FOR FPGA

Hoang Trang[1]

**Abstract**

Many applications need multi-ported memories to increase the process speed and allow write and read memory banks from multiple ports at the same time. Most FPGAs only support dual port RAM (including simple and true dual ports), and we must build by ourselves multi-port RAM from available logic elements and RAM building blocks of FPGAs when the number of port exceeds 2. In this paper, the conventional approaches for implementing multi-ported memories are analyzed by using Stratix FPGA of Altera. This work proposes Live Value Table-Multipumping based approach that could improve performance in area, speed and increase the memory depth compared with the previous works. The area detail is also given to help the designer defining which blocks affect significantly to the total area of design, and to help designer deciding which blocks should be optimized. This work also helps the multi-ported memory designer choosing which approach is suitable for the specific application.

Nhiều ứng dụng đòi hỏi các bộ nhớ đa cổng để tăng tốc độ xử lý và cho phép ghi/đọc các băng bộ nhớ từ nhiều cổng tại cùng thời điểm. Hầu hết FPGA trên thị trường chỉ hỗ trợ RAM 2 cổng, do vậy chúng ta cần thiết tự xây dựng RAM đa cổng từ các thành phần logic và các khối RAM có sẵn trong FPGA với các ứng dụng đòi hỏi số cổng nhiều hơn 2. Trong bài báo này, các phương pháp truyền thống để xây dựng bộ nhớ đa cổng sẽ được phân tích qua việc sử dụng FPGA Stratix của hãng Altera. Bài báo còn đề xuất phương pháp dựa trên kết hợp Live Value Table – Multipumping; so với các nghiên cứu trước, phương pháp này giúp tăng chất lượng trong diện tích, tốc độ và tăng dung lượng bộ nhớ. Các phân tích, kết quả chi tiết về diện tích cũng được trình bày để giúp người thiết kế xác định được khối nào chiếm tỉ lệ diện tích lớn và từ đó quyết định khối nào cần phải tối ưu. Bài báo này cũng giúp người thiết kế bộ nhớ đa cổng có thể chọn lựa được phương pháp nào là thích hợp cho từng ứng dụng cụ thể.

**Index terms**

Multi-ported memory, multipumping memory, FPGA, Live Value Table.

## 1. Introduction

IN recent years, FPGA technology takes a more and more important role on the integrated circuit area because of minimizing the time to market and related cost of designing any applications. FPGAs can be used to implement any general logic function that ASIC can perform. Modern FPGA devices allow designers to build a larger and more complex system-on-chip device to serve any specific applications that need sharing, queuing, synchronous between functional units. Meanwhile, many applications need multi-ported FIFO and multi-ported register file to increase the process speed and allow write and read memory banks from

(1)Ho Chi Minh City University of Technology.

multiple ports at the same time. For instance, designers often use FIFOs as synchronizer for data transfer between the clock domains. These circuits actually need the multi-ported FIFOs to improve the speed of data transfer between all clocks domains because they allow each functional unit simultaneously read and write with the memory bank at the same time. Next, Instruction Level Parallelism (ILP) is a measure of how many instructions can be implemented simultaneously. And many processors often exploit multi-ported register file to implement ILP to help improve the performance of processors [1]. However, these above applications of multi-ported memories on FPGA are little used in practice. The problem is that it is not efficient to implement multi-ported memories on FPGA. Most FPGAs only support dual port RAM (including simple and true dual ports) [2], and we must build by ourselves multi-port RAM from available logic elements and RAM building blocks of FPGAs when the number of port exceed 2. There are several attempts to implement multi-ported memories for FPGAs, mainly for register files of soft processor. Most of the previous works come from the motivation of improving Instruction-Level Parallelism (ILP), where simultaneously multiple-instruction executing requires multiple concurrent accesses to register file.

Most soft processors without ILP simply use replication to provide the 1W/2R register file required to support a typical three-operand Instruction Set Architecture (ISA). Datapath of soft processors was explored in different approaches [1] and compares them to the Altera's NiosII soft processor [3]. Besides, some soft processors with some ILP may require a simple replicated 1W/2R register file as in [4]. However, the inputs and outputs of the instructions are read/written one-at-a-time, no extra memory or register file ports required and thus having low speed. When multiple instruction operands need to be simultaneously read and written, ILP is formed as Very Long Instruction Word (VLIW). One example of VLIW soft processor is from [5] in which multi-ported memories enabling efficient communication between multiple functional units is presented. However, multi-ported register file in VLIW soft processor is implemented entirely using the FPGA's reconfigurable logic and thus limits the operating frequency of the processor. A fast multi-ported register file for a VLIW soft processor was proposed and implemented in [6] by using both replication and banking of block RAMs. However, this approach is good one for supporting multithreading [7], [8] but for Write-After-Read (WAR) violations. To avoid this violation, a form of multipumping by performing all writes before reads was presented from [9] in which a random-access multi-ported register file was described. However, this design requires using multiple-phase clocking in the entire system. To overcome this issue, the Live Value Table (LVT) based approach was given in [10] to implement multi-ported memories with an arbitrary number of read and write ports. This approach was shown that it could give smaller, faster design than pure reconfigurable logic and pure multipumping implementations. However, LVT-based approach would be limited in the depth of memory and in speed because multipumping approach is orthogonal to LVT-approach and LVT approach combines replicated and banking approaches. This work proposes the LVT-multipumping based approach that could increase the memory depth, increase speed, and reduce the area.

The rest of this paper is organized as follows: conventional multi-ported memory approaches and proposed approach are introduced briefly in session II. Session III describes the implementation results of conventional and proposed approaches with Stratix FPGA of Altera that will show the benefit of this work. Finally, a conclusion is given in session IV.

## 2. Multi-ported memory design approaches

In this section, conventional approaches of designing multi-ported memory are described briefly. Each one will be introduced in the structure of circuits, operation, advantages, disadvantages and corresponding applications. At the end of this section, proposed approach is presented to reduce most of the conventional approaches' disadvantages, to give effective performance that could satisfy the demand of modern designs.

### *2.1. Adaptive Logic Modules-ALM based approach*

The first method is to implement Multi-Ported Memory by ALMs (Adaptive Logic Modules) which are the basic building block of logic in the Stratix III architecture [2] and provide advanced features with efficient logic utilization. This design approach uses D locations memory (equivalent to $log_2 D$ address width) with D **(m-to-1) multiplexers** (m is the number of write ports) in front of memory to select the suitable memory locations to write data and n (n is the number of read ports) **(D-to-1) multiplexers** behind memory to select suitable data to read from memory. The problem is that a very large area of ALMs will be used to implement 2 groups of multiplexers and memory; they will reduce the operation frequency and make the CADs tool take a long time to implement place & route in FPGAs. So, this method is not effective to implement multi ported memory when the number of ports as well as the data depth is large [11], [12]. Figure 1 illustrates the operation of multi ported memory using ALMs with m Write Ports and n Read Ports. In this figure, only the data signals are shown (the address, clock, enable signals are ignored).

Implementing multi-ported memory using ALMs usually takes more resource than using specific RAM blocks, because ALMs are usually configured to implement logic, arithmetic, and register functions, thus not good in both area and speed performance to implement memory [13]. RAM blocks would be concentrated to implement multi-ported memory.

### *2.2. Replicated approach*

The replicated method as in Figure 2 is used to expand the number of read ports of memory by adding some extra copies of RAM blocks with the number of copies based on the number of read ports. This method is only used when the memory have only one write port.

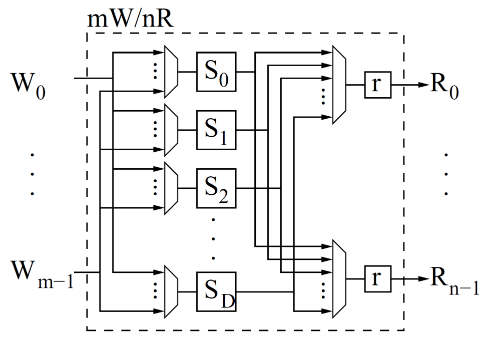

*Figure* 1. *mW/nR Multi-Ported Memory using ALM*

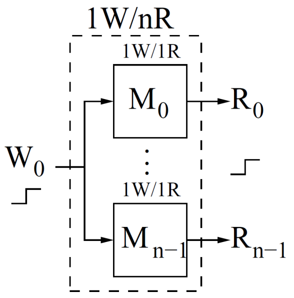

*Figure* 2. *1W/nR Multi-Ported Memory using Replicated approach*

## *2.3. Banking approach*

Compared with the replicated method, the banking method as in Figure 3 is different and can support an arbitrary number of read ports and write ports. Memory will be divided into m sections (corresponding to m write ports). So that, each write port and read port can only access to only one corresponding section. Therefore, this method does not support fully multi-ported memory because a write port cannot access to any location and similarly with the read port. And finally, this method must be combined with any other methods to create the truly multi-ported memory.

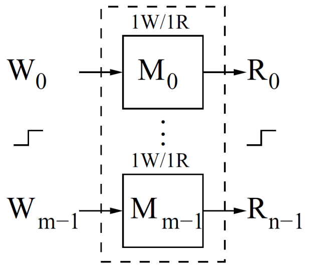

*Figure* 3. *mW/nR Multi-Ported Memory using Banking approach*

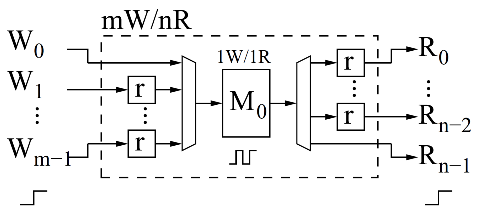

*Figure 4. mW/nR Multi-Ported Memory using Multipumping approach*

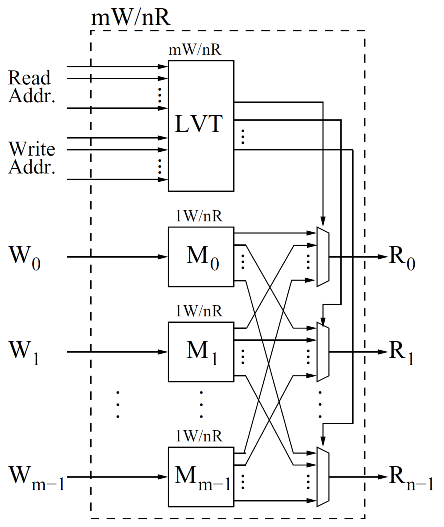

*Figure 5. mW/nR Multi-Ported Memory using LVT approach [10]*

### *2.4. Multipumping approach*

This approach uses an external clock that is multiple speed of system clock (that called multipump factor) to control the memory location access process as in Figure 4. The temporary registers and multiplexers will be added to the circuit before and after the memory to multiplex the read/ write operations. For instance, a 1W/1R can support to 2 write ports and 2 two read ports with the multipump factor is 2. Therefore, multipumping approach helps to reduce the area of the design and is suitable for some applications that don't need high speed. On the contrary, the main disadvantage of this design is the reduction of the operation speed to multipump factor time, for example, multipumping with multipumping factor 2 will reduce the operation speed by two times.

### *2.5. Live Value Table-LVT approach*

The LVT approach [10] was proposed from a form of indirection through a structure called the Live Value Table (LVT), which is itself a small multi-ported memory implemented in reconfigurable logic similar to Figure 1. This approach allows a banked design to behave like a true multi-ported design by directing reads to appropriate banks based on which bank holds the most recent or "live" write value as in Figure 5 [10].

### *2.6. Proposed approach: LVT-Multipumping based approach*

A novel approach is proposed to increase the performance of design by combining LVT approach, multipumping approach and additional circuitry to solve data contention, and reduction of the operation speed due to multipump factor time. In this design, each write port will write data to corresponding memory bank being same as the banking approach. The read ports are constructed similar to the replicated approach. The number of extra copied RAM blocks in each bank based on the number of read ports. Therefore, the mW/nR memory will be designed with m x n of the 1W/1R memory. Besides, the circuit takes responsibility to solve the data contention which banks are storing the newest data for each memory locations. From those results, each output port will select the suitable extra bank for reading each memory locations. Intuitively, this proposed approach makes the multi-ported memory have high operating speed, nearly equal to the RAM block speed; area cost is smaller than LVT-based approach when the number of ports increases because the resource equal mxn times of the real requirement of RAM blocks is taken and multipumping approach is used.

## 3. Implementation result

This paper uses Altera's Quartus 11.0 Web Edition and Stratix III FPGA to analyze the results and to compare with the Nios II soft processor [3] and the previous works.

### *3.1. 1W/2R memory*

To evaluate the performance of different methods to design multi-ported memory, a 1W/2R memory is chosen because it is simple, commonly used and supported by the most FPGA devices [3],[4]. In the Figure 6, the parameters of area and speed of different implemented methods are indicated with each point shown as the depth of memory (for example 32, 64, 128 and 256 bit). These results are used to discuss the conventional techniques for building multi-ported memories on FPGAs.

Figure 6 shows the comparison performance of different methods to implement 1W/2R memories: Pure-ALM (only use available logic elements), Repl-MLAB (use MLAB for replicated approach), Repl-M9K (use M9K for replicated approach), MP-MLAB (use MLAB for multipumping approach) and MP-M9K (use M9K for multipumping approach). From these results, the Repl-M9K would be the best approach because it costs little resource and obtains a high speed close to the speed of RAM block. This approach does not use different logic elements besides RAM block, but the disadvantage is that it is only applied when the number of write port is 1. For clear presentation and deep analysis in 1W/2R, results of area (ALMs) and speed (MHz) in each approach are given in Table 1-Table 5. In these tables, first column describes the memory depth from 32 to 256 bit. Next columns describe the area and speed

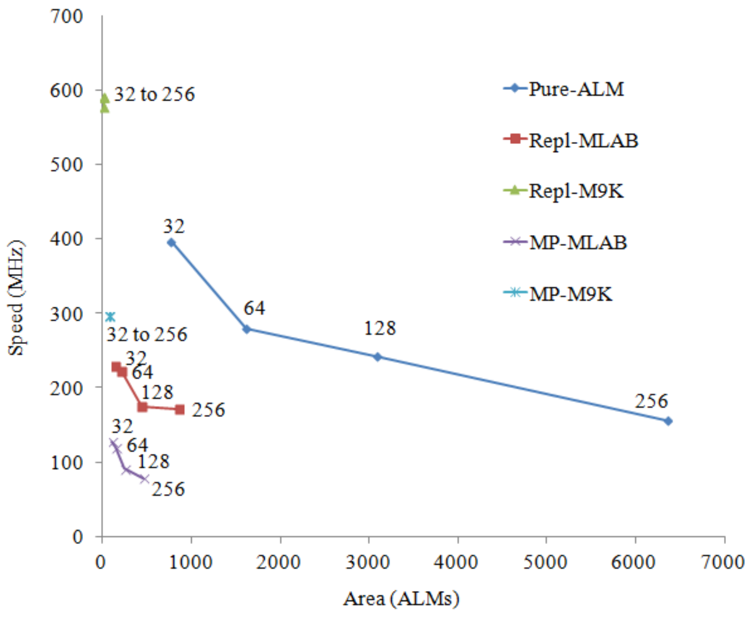

*Figure* 6. *Comparison Performance of Different 1W/2R Memories*

*Table* 1. *Performance of 1W/2R Memory using Pure ALMs*

| Pure ALM | | | | |
|---|---|---|---|---|
| Memory depth | Area (ALMs) | Speed (MHz) | Area Increment | Speed Drop |
| 32 | 744 | 390.5 | 1.0 | 0.0% |
| 64 | 1608 | 299.7 | 2.2 | 23.3% |
| 128 | 3079 | 275.0 | 4.1 | 29.6% |
| 256 | 6333 | 164.3 | 8.5 | 57.9% |

of design. Last two columns compare the area increment (Area Increment) and speed drop (Speed Drop) over the case of memory depth of 32 bits when increasing the memory depth.

*3.1.1. Pure ALMs:* Table I presents the results in implementing 1W/2R memories by using only available logic elements (Pure-ALM). The design of multi-ported memories using pure ALMs is not good scaling both in area and speed. When increasing the memory depth in double, the resource needed to implement also increases nearly double, and the operating speed also drops significantly with the maximum speed is 390.5 MHz when the memory depth is 32. The speed drops 57.9% when the memory depth increases from 32 to 256. One remark extracted from this result is that Pure-ALM approach is not effective to implement multi-ported memory when the number of ports as well as the memory depth is large.

*3.1.2. Replication:* There are two ways to implement multi-ported memories on FPGA by replicated approach that are using MLAB block and M9K block. The implementation results of these methods are presented in Table II and Table III. By using available RAM block, the design has better scale than using logic elements on FPGA. In the replication MLAB method, the max area is 856 ALMs and the min speed is 326.3 MHz; the area increases

*Table 2. Performance of 1W/2R Memory using Replication MLAB*

| Replication MLAB | | | | |
|---|---|---|---|---|
| Memory depth | Area (ALMs) | Speed (MHz) | Area Increment | Speed Drop |
| 32 | 114 | 387.6 | 1.0 | 0.0% |
| 64 | 210 | 363.4 | 1.8 | 6.3% |
| 128 | 422 | 370.4 | 3.7 | 4.4% |
| 256 | 856 | 316.3 | 7.5 | 18.4% |

*Table 3. Performance of 1W/2R Memory using Replication M9K*

| Replication M9K | | | | |
|---|---|---|---|---|
| Memory depth | Area (ALMs) | Speed (MHz) | Area Increment | Speed Drop |
| 32 | 90 | 580.0 | 1.0 | 0.0% |
| 64 | 90 | 580.0 | 1.0 | 0.0% |
| 128 | 90 | 580.0 | 1.0 | 0.0% |
| 256 | 90 | 580.0 | 1.0 | 0.0% |

nearly double when doubling the memory depth of design while the speed decreases only 18.4% when memory depth increases from 32 to 256. Therefore, this design has better scale in speed drop than in area increment.

Being compared with Pure ALMs method in section III.A.1, this design only needs $\frac{1}{7}$ area and obtains the speed faster than twice.

On the replication M9K method, the design obtains the best scale compared with the above described. It needs only 2 M9K RAM blocks (equivalent 90 ALMS area) and the speed is unchanged when increasing the memory depth because this design does not need any logic elements besides RAM blocks, thus the speed is equal to the M9K RAM block speed. Therefore, replication M9K is the best method to implement multi-ported memories with only one write port.

*3.1.3. Multipumping:* The implementation results of multi-ported memory using multi-pumping approach with MLAB and M9K block are given in Table IV and V, respectively. As shown in the section II, MLABs have only supported simple dual port RAM in which one port is read and the remaining port is write; M9Ks have supported simple dual port and true dual port in which two ports can be dedicated as read and or as write. Multipumping approach is implemented well on the true dual port RAM because this method is time multiplex access to read and write port and suitable for the structure of true dual port to obtain a good performance design. In the multipumping MLAB method, the max area is 467 ALMs and the min speed is 142 MHz; the area increases nearly 1.5 times when doubling the memory depth of design while the speed decreases to 35.2 % when memory depth increases from 32 to 256. Therefore, this design type has better scale in area than in speed, thus, this method is suitable with the applications that do not need high speed but require the minimum area. On the multipumping M9K method, the scaling of area and speed is also good when increasing the memory depth,

*Table 4. Performance of 1W/2R Memory using Multipumping MLAB*

| Multipumping MLAB | | | | |
|---|---|---|---|---|
| Memory depth | Area (ALMs) | Speed (MHz) | Area Increment | Speed Drop |
| 32 | 100 | 219.3 | 1.0 | 0.0% |
| 64 | 147 | 192.3 | 1.5 | 12.3% |
| 128 | 252 | 171.5 | 2.5 | 21.8% |
| 256 | 467 | 142.0 | 4.7 | 35.2% |

*Table 5. Performance of 1W/2R Memory using Multipumping M9K*

| Multipumping M9K | | | | |
|---|---|---|---|---|
| Memory depth | Area (ALMs) | Speed (MHz) | Area Increment | Speed Drop |
| 32 | 116 | 291.9 | 1.0 | 0.0% |
| 64 | 118 | 291.9 | 1.0 | 0.0% |
| 128 | 120 | 291.9 | 1.0 | 0.0% |
| 256 | 121 | 291.7 | 1.0 | 0.1% |

but the speed is only equal a haft of implementing by replication M9K method. And the area is also larger than replication method because this type of design uses true dual port instead of simple dual port. Thus, multipumping M9K method would be not efficient when implementing multi-ported memory with the small number of port and suitable with the not high speed RAM applications.

### *3.2. Multi-ported memory 2W/4R*

The implementation results of speed and area for 2W/4R memory configuration in different approaches are presented in Figure 7. The area (ALMs) in LVT-based approach of [10], and this work with LVT-multipumping approach is clearly better than Pure-ALM approach, especially when using M9K RAM blocks in FPGA. In using MLAB block, the approach in this work could help the designer to obtain the memory depth up to 256 bits that is larger than one in [10] of up to 64 bits. This work approach in M9K implementation has the lowest area and highest speed because it has lower interconnection requirements and not the additional logic circuits similar with the ALM or LVT[10]-MLAB implementations. When memory depth increases from 32 to 256, the speed in this work with M9K implementation decreases from 584 MHz to 343 MHz, thus this approach is suitable for the high speed applications, for instance, FIFO and the registers for Instruction Level Parallelism (ILP).

In implementation result of area, the total area of design should be divided into different areas depending on the function of each block to help the designer to define which blocks affect significantly to the total area of design, and to help designer to decide which block should be optimized. This detail in area for different 2W/4R memories is given in Figure 8. In this figure, the Pure_ALM only has memory bank, this work with MLAB and with M9K have 3 function blocks. In the 2W/4R memories configuration, the memory banks occupy the

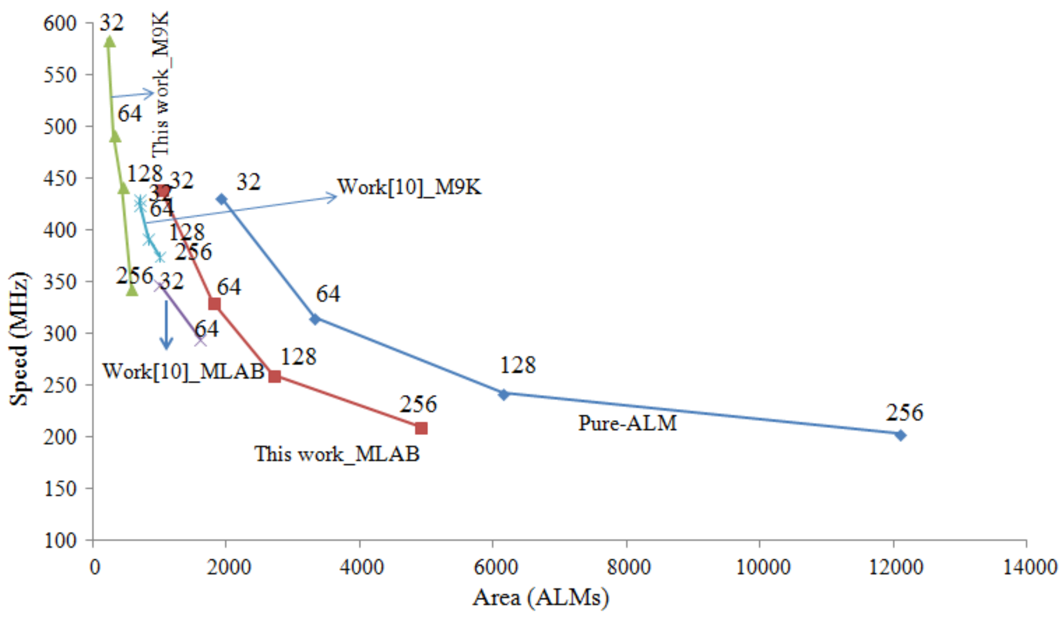

*Figure 7. Performance comparison of Different 2W/4R Memories*

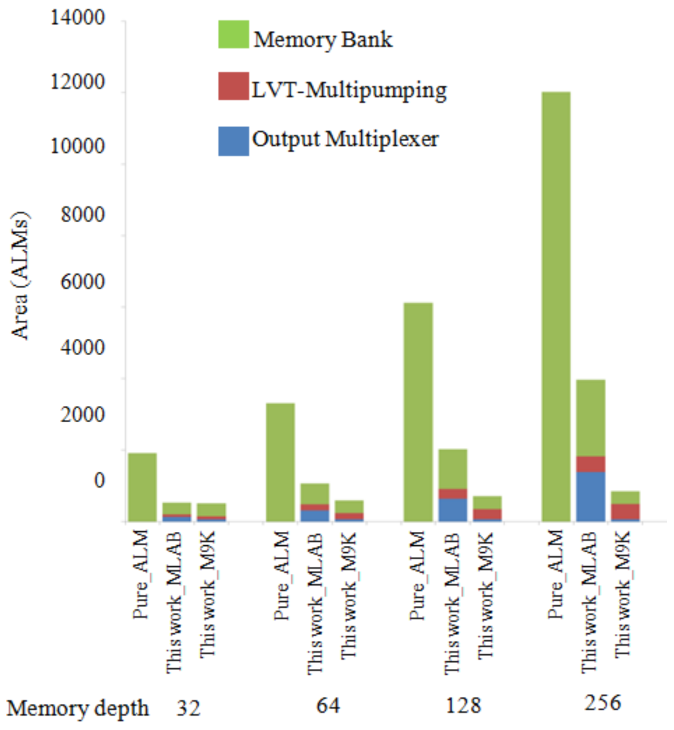

*Figure 8. Area detail in different approaches for different 2W/4R memories*

most area of designs. Therefore, it is very difficult to optimize the logic function to obtain the reduction in area.

## 4. Conclusion

Multi-ported memory takes a more and more important role on the applications that need sharing, queuing, synchronization between functional units. It helps to increase the process speed and to allow write and read memory banks from multiple ports at the same time. However, most FPGA devices only support 2 ports. Some conventional approaches such as ALM, banking, replication, multipumping retain disadvantages in timing, area, expanding the memory ports ... The LVT-multipumping based approach in this paper is introduced to increase the performance of multi-ported memory in area and speed. This work could help the multi-ported memory designer to choose which approach is suitable for the specific application. One problem when increasing significantly number of memory ports is to take much RAM blocks to construct the required multiple ports memory. Potential work should be done to decrease the number of RAM blocks and still satisfy all memory requirements.

## References

[1] Yiannacouras, P., Steffan, J.G., Rose, J., Application-specific customization of soft processor microarchitecture, Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field Programmable Gate Arrays, ACM, 2006, pp. 201–210

[2] Stratix III Device Handbook, Volume 1 Chapter 4: TriMatrix Embedded Memory Blocks in Stratix III Devices, http://www.altera.com/literature/hb/stx3/stx3_siii51004.pdf, Version 1.8, May, 2009, pp.4.1-4.9

[3] Nios II Processor Reference Handbook, http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf, Version 9.0, Accessed Sept. 2009, March, 2009

[4] Carli, R., Flexible MIPS Soft Processor Architecture, Technical report, Massachusetts Institute of Technology-Computer Science and Artificial Intelligence Laboratory, June, 2008

[5] Jones, A. K., Hoare, R., Kusic, D., Fazekas, J., Foster, J., An FPGA-based VLIWprocessor with custom hardware execution, International Symposium on Field-Programmable Gate Arrays, 2005

[6] Saghir, M.A.R., El-Majzoub, M., Akl, P., Datapath and ISA Customization for Soft VLIWProcessors, IEEE International Conference on Reconfigurable Computing and FPGAs, September, 2006, pp. 1–10

[7] Moussali, R., Ghanem, N., Saghir, M.,Microarchitectural Enhancements for Configurable Multi-Threaded Soft Processors, International Conference on Field Programmable Logic and Applications, ,August, 2007, pp. 782–785

[8] Labrecque, M., Steffan, J., Improving Pipelined Soft Processors with Multithreading, Conference on Field Programmable Logic and Applications, August, 2007, pp. 210–215

[9] Manjikian, N., Design Issues for Prototype Implementation of a Pipelined Superscalar Processor in Programmable Logic, IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, August, 2003, pp. 155–158

[10] Charles, E.L., Gregory, J.S., Efficient multi-ported memories for FPGAs, Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays, 2010, pp. 41-50

[11] Intel Corp, Intel 64 and IA-32 Architectures Optimization Reference Manual, March, 2009

[12] Intel Corp, First the Tick, Now the Tock: Intel Microarchitecture (Nehalem), White Paper No. 319724, 2009

[13] Lange, H., Koch, A., Memory Access Schemes for Configurable Processors, International Conference on Field-Programmable Logic, 2000

*Ngày nhận bài 22-5-2012; ngày gửi phản biện 22-5-2012.*
*Ngày chấp nhận đăng 5-6-2013.*

■

**Hoang Trang** was born in NhaTrang city, Vietnam. He received the Bachelor of Engineering, and Master of Science degree in Electronics-Telecommunication Engineering from Ho Chi Minh City University of Technology in 2002 and 2004, respectively. He received the Ph.D. degree in Microelectronics-MEMS from CEA-LETI and University Joseph Fourier, France, in 2009. From 2009-2010, he did the post-doctorate research in Orange Lab-France Telecom. Since 2010, he is lecturer at Faculty of Electricals-Electronics Engineering, Ho Chi Minh City University of Technology. His field of research interest is in the domain of FPGA implementation, Speech Recognizer, MEMS, fabrication, integration of passive components and function for telecommunications, Embedded System, System-on Chip (SoC).