IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 13, Issue 3, Ver. I (May. - June. 2018), PP 38-47
www.iosrjournals.org

Content Addressable Memory for Multi Page Memory Interface

K.Suresh Kumar[1], Y.Rajasree Rao[2], K.Manjunathachari[3]
[1]Electronics & Communication Engineering Department, SSJ Engineering College, Affiliated to JNTUH Hyderabad, Telangana-500075, India
[2]Professor & Dean, Electronics & Communication Engineering Department, St.Peter's Engineering College, Hyderabad, Telangana, India
[3]Professor.&HOD, Electronics & Communication Engineering Department GITAM University, Hyderabad, Telangana, India
Corresponding Author: K.Suresh Kumar[1]

Abstract: *content addressable memory (CAM) interface plays a major role in current applications, where the stored data are coded or processed for interfacing, reading and processing of stored data. In the process of memory interface, as the volume of data is increasing, the conventional single page memory interface are not suitable. Hence, to process over a large volume of data multi page memories are developed. Where, multi page memories are suitable for large volume storage, accessing of such memory is a complex task. In this paper, a new interface design to data interfacing on such memory unit is proposed. The design approach, gives a simpler modeling of data interfacing in content based addressable memory interfacing applications.*

Key words: *Content Addressable Memory, multi page memory interface, system design.*

Date of Submission: 12-05-2018 Date of acceptance: 28-05-2018

I. Introduction

Content addressable memory interface has gained its usage from long time. Such application, data are stored on to a memory location, and read from the addressed location on requirement. Such memory interface has the advantage of storage and accessing on a direct addressing mode. From the initiation of such memory interface design, efforts are been made to perform a simpler accessing and faster processing of data accessing. In various past developments, it is observed that the memory interfacing is been improved via matching logic or addressing register utilization. CAM hardware has been available for decades and many research are addressed to development of high capacity and effective CAM designs at circuit, architectural and application level. A lot of projects are leaning towards "real live" application for effective algorithms for package forwarding based on CAMs and its extended version Ternary CAM (TCAM) [1]. At lower level designs many papers introduce methodologies and optimization to speed, power and physical circuit resources. Authors of [2] in detail describe the principle of CAM functions at transistors and circuits level including core cells, match line and search line structures and power consumption formulation. Also power and area reducing techniques are presented on the circuit level. Practical design on architecture level is presented by [4]. The proposed CAM chip design is based on modification to the RAM chip circuit explained in [5]. CAM memories enhanced with "don't care" states are used for more complex project like hardware based Network Intrusion Detection and Prevention Systems (NIDPS) [9].CAMs have a single clock cycle throughput making them faster than other hardware- and software-based search systems. CAMs can be used in a wide variety of applications requiring high search speeds. These applications include parametric curve extraction ,Hough transformation, Huffman coding/decoding, Lempel–Ziv compression[10]–[13],and image coding applications [14]. However, the conventional single page memory interfaces are not suitable for large volume data interfacing. Hence, multi page memories are developed. In this paper, we present a new approach to multi page memory design interface for large volume interfacing. To present the state approach, this paper is presented in 5 sections, where section 2 outline the approach of data handling and its issues in large data accessing. The proposed design approach of multi page memory interface is outlined in section 3. Section 4 outlines the obtained experimental result for the developed system. The conclusion for the developed work is outlined in section 5.

II. Cam Data Handling

The technology underlying the practical use of CAM hardware, software, and design techniques are fast changing and complex. Designers contemplating the development of products using CAM technology are confused to choose with a vast array of tools, technologies, and methodologies. In this all technologies and methodologies, the issue of data management varies as the application changes. Some of the design challenges that are unique to (or at least especially important for) the design of CAM-intensive systems have been considered. CAM-intensive systems today rely on increasingly complex and computationally intensive applications. Many CAM systems must meet extremely rigorous speed goals, since they operate on lengthy segments of real-world signals in real-time. Another key characteristic of a CAM system is its sample rate: the rate at which samples are consumed, processed, or produced. Combined with the complexity of the algorithms, the sample rate determines the required speed of the implementation technology. The sample rates required for CAM systems vary over many orders of magnitude, depending upon the types of signals being manipulated. In many cases, sample rates are quite high relative to the basic clock cycle time of the available hardware technology. An additional challenge in CAM design verification is the need for realistic test data. The verification of CAM application often requires complex, realistic test signals. The development of CAM system begins with the establishment of requirements for the system. These requirements may include power, size, weight, bandwidth, and signal quality. During the design development, the designer is mostly concerned with exploring approaches to solve the problems posed by the specifications at an abstract level. At the algorithm development level, the designer must be able to specify and experiment with both the control behavior and the processing behavior of the application. System architecture use to describe the overall selection and organization of the main hardware and software components in a system is then a critical task. In selecting a system architecture,

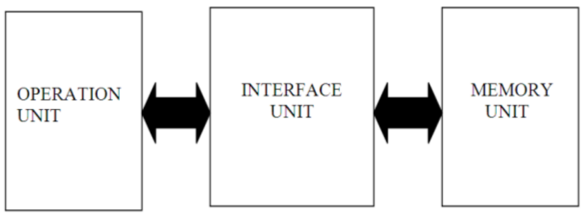
the designer's inputs typically include the algorithms and other functionality to be implemented. The design of system architecture is often the most important step in the design process in terms of its influence on the product's performance, cost and design time. Signal processing software is perhaps the most important part of CAM system software. Many CAM systems are implemented using multitasking, which requires a real-time operating system to coordinate the execution of multiple tasks. CAM system involve the development of new hardware-particularly those systems targeted at cost-sensitive, high-volume applications. System integration has to take place throughout the design process. As the designs are refined, the integration is also refined. Starting system integration early in the design process and refining the integration as design proceeds paves the way for relatively straightforward final system integration. The current CAM systems, unlike the general-purpose memory, use a non-uniform addressing model in which the primary components of the memory system is the DRAM and dual tag less SRAMs are referenced through completely separate segments of the address space. The recent trend of programming CAMs in high-level languages. In many of today's high-performance CAMs this non-uniform model is being replaced by a uniform model a transparent organization like that of most general-purpose systems, in which all memory structures share the same address space as the DRAM system. In such a memory organization, one must replace the CAM's tagless SRAMs with something resembling a general-purpose cache. The dual-channel SRAM design provides a large amount of fast storage while supporting guaranteed single-cycle access to both operands of a typical instruction (e.g. the product terms of a multiply accumulate). The SRAMs are tagless because they are not transparent to the programmer: CAMs offer segmented memory spaces where different physical memory structures (SRAM0, SRAM1, DRAM, ROM, etc.) are explicitly indicated by corresponding sets of data addresses. A CAM

memory model explicitly places the system's memory structures at specific disjunction locations in the name space. Because they behave differently, traditional caches and tagless SRAMs have different implementations and different ramifications. Hence, the memory interface for CAMs, as their applications tend to behave slightly different than traditional general-purpose applications.

III. Multipage Memory Interface

For the development of a CAM design, for Higher level integration a interface design with multiple page memory interface is suggested. The proposed system design is as given below, The basic operation interface of the suggested CAM system is shown in figure 1.

Figure 1: Interface for a CAM unit



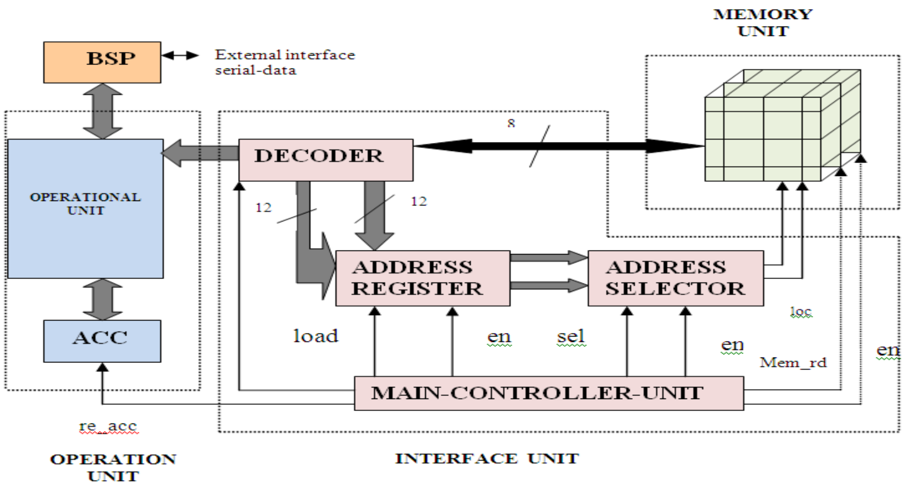
The system architecture of the suggested CAM system deals with the three units namely, 1. Operation Unit 2. Interface Unit and 3. Memory Unit.

3.1 Operational Unit

The Operation unit performs all the arithmetic and logical operations for instruction executions. The Interface unit acts as an interface for data between operation unit and memory unit here the data is interfaced with address registers, address selector, decoder and a main control unit for controlling all the

interfacing of memory accessing and delay computing. The last two cycles are used to shift the data to be used with the next input. The operational units for the suggested system is briefed in below section.

Figure 2: Proposed system architecture for CAM interface in Multi page memory



modules. The Memory unit consists of multi pages of memory in order to store large data. The block diagram of the data interface system is shown in figure 2.

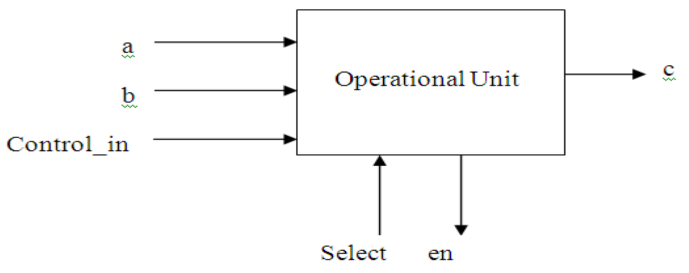
3.2 Data Interface unit

This block of the interface handles all of the data and previously stored values from the memory. Looking at the operation for a multi page memory operation, for multiple pages addressing that need to be accessed from the data block: b0, b1, b2, a1, and a2. Also, the values of Ω_1 and Ω_2 are values that are accessing delayed through the system. These values are stored in memory interface unit to perform a synchronous interface at correct time of interfacing. This allow the interface to correctly derive the new address values for the address signal to interface. Therefore, this memory set up in a “shift-and-rotate” fashion so that when a new value enters the system, the old values will be rotated out and the delayed values can be stored correctly. For the implementation of the stated design, a general flow was used defined using VHDL designs. Seven cycles were used for the entire design in VHDL. The first cycle is used to trigger an addressing logic. A temporary register was used to store the data until the next cycle. The next five cycles load the data to the address interfacing for data accessing and delay buffering. For each of these cycles the fetched data from the memory is fed into temporary register. To attain a synchronous operation, the clock that controls the cycles is set with a period long enough for the

Functional Units:

a) Operational Unit

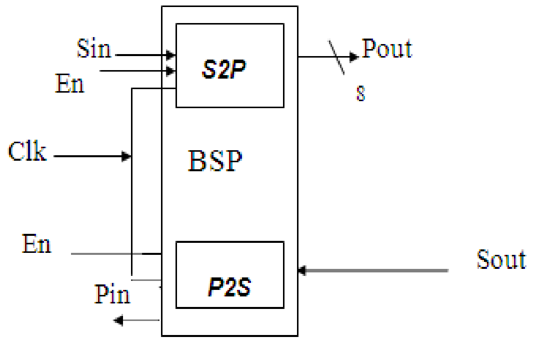
Figure 3: Block unit for operational Unit



This unit performs all operations as per the memory-accessed instructions. This unit fetches data from data bus and performs operations based on the instructions passed. The instructions are decoded and executed based on the operations specified. All the instructions after being executed are passed to the storage Unit. For the interfacing of the memory unit, a serial interfacing is suggested.

b) Bit Stream Processing unit

Figure 4: Bit stream interface to the developed system



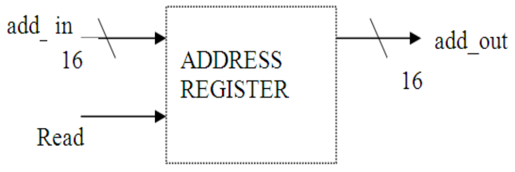
BSP is data transfer logic, constituting of two transfer logics namely S2P and P2S.It receives the input data from the peripherals.

S2P: This module transformation generally receives data from host to peripherals to process in the process architecture. The input S2P transforms the serial data to parallel. The modules containing serial data and buffers the module upto reaching a temporary register. The whole transformation takes parallel input and transforms to serial data.

P2S: This process enables for 8-clock cycles of parallel data. When transfers the stored data to process in the 9th cycle, the 8-bit data constitutes for one clock cycle and for a serial data upto peripheral the data is transformed.

c) Address Register unit

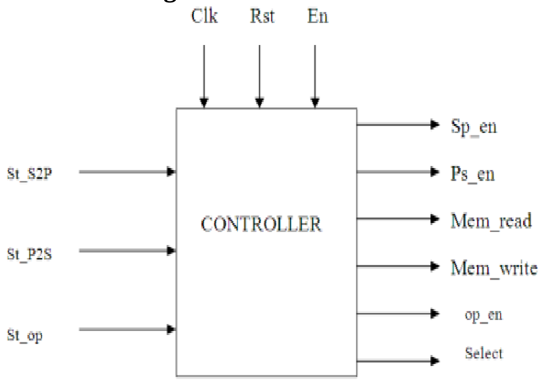
Figure 5: Address Register unit



This is a 32-bit address register used for addressing the memory location. The memory is fetched from the location pointed by the address register. The address register combines with the page address and is passed to the memory. This unit takes a 32-bit address as input 'add_in' and based on the selection addresses the particular memory location.

d) Controller unit

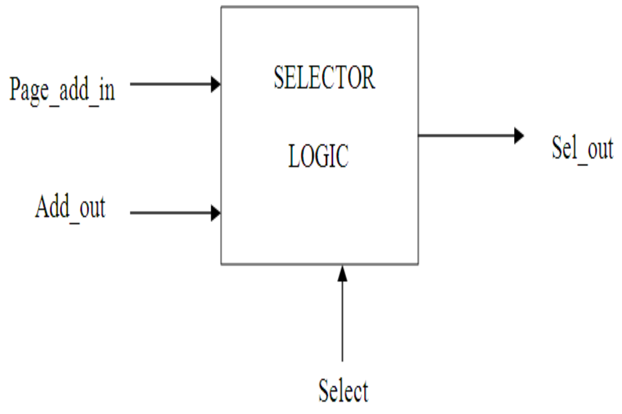
Figure 6: Controller unit



The controller logic is realized or carried out using FSM (Finite State Machine).The Controller reads the status of each unit and generates the control signals to individual blocks based on the selectivity of each module. Based on the status, the next module for operation is carried out.

e) Selector unit

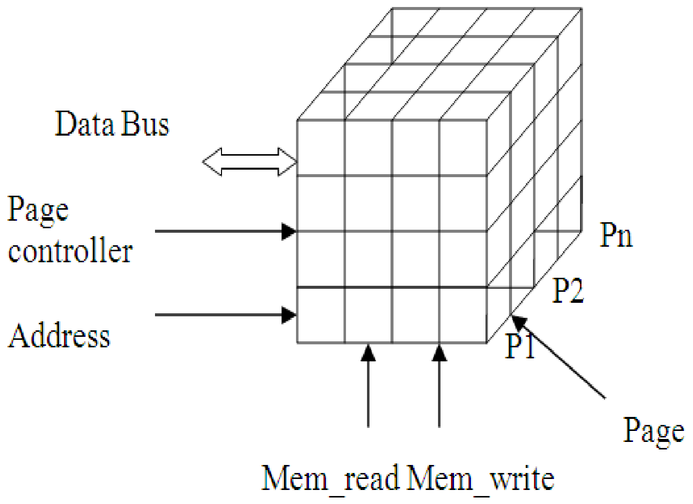
Figure 7: Selector Logic unit



Page controller and Address Register can be selected by means of selector. Based on selected input, the desired selection is being done. Page Controller and Address Register are selected based on the input line ‘select’. If select is high, by default the page address is being generated and Address Register is generated when select line is low. Otherwise if select ‘1’, page address is generated and vice-versa.

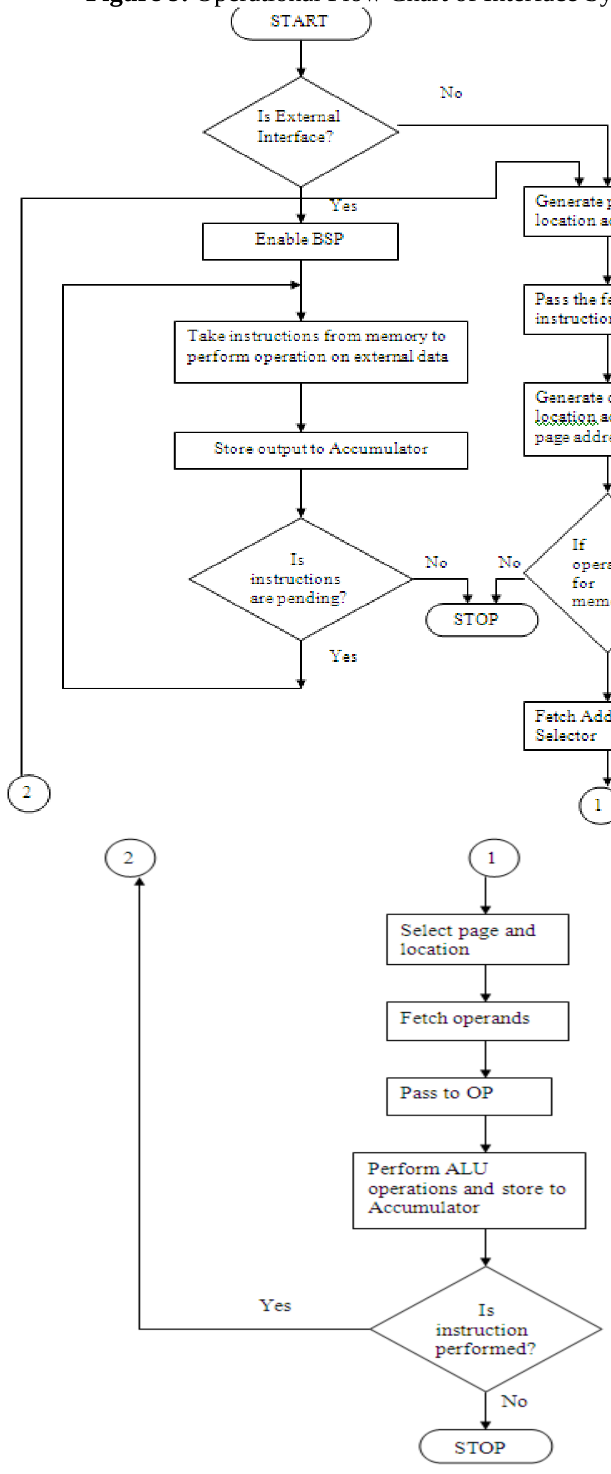
f) Multi Page Memory unit

Figure 8: Multi page memory unit



This unit is used to store large amounts of volumetric data occupying less space. Based on the Controller request the memory is allocated for read operation or write operation. The page memory is passed to the processing unit through the data bus. The memory allocation in this unit is in the form of layers and each layer is known as “page”. All the pages in multipage memory are controlled by the page controller. The overall operational flow chart is illustrated in figure 9.

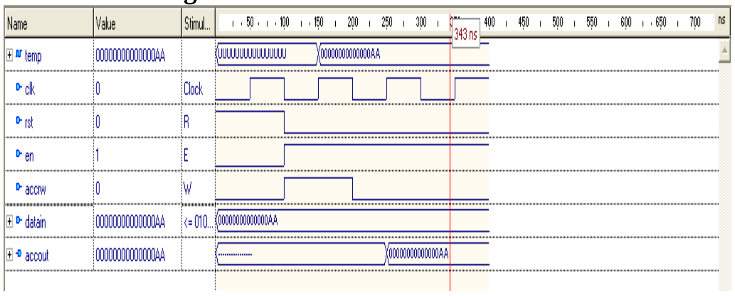
Figure 9. Operational Flow Chart of Interface System


START
Is External Interface?
No
Yes
Enable BSP
Generate page and location address
Pass the fetched instruction to Decoder
Take instructions from memory to perform operation on external data
Generate opcode, location address and page address
Store output to Accumulator
Is instructions are pending?
No
No
If operands for memory?
STOP
Yes
Yes
Fetch Address Selector
2
1
2
1
Select page and location
Fetch operands
Pass to OP
Perform ALU operations and store to Accumulator
Yes
Is instruction performed?
No
STOP

IV. Experimental Result

To validate the developed approach, a simulation approach to multi page memory interface is carried out. For the implementation, the proposed approach is defined in VHDL language, simulated over Active HDL tool and synthesized for Xilinx FPGA device carried out over Xilinx synthesizer tool. The obtained simulation results are as shown below,

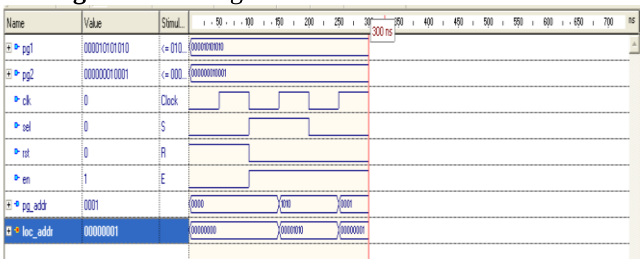
Figure: 10 Data Accumulation results



Name	Value	Stimul...
temp	00000000000000AA	
clk	0	Clock
rst	0	R
en	1	E
accrw	0	W
datain	00000000000000AA	<= 010...
accout	00000000000000AA	

Figure 10 illustrates the accumulated data for processing accessed from different pages of the multi page memory unit. The accumulated data are accessed with page address and location address simultaneously. To perform the address selection operation address selector unit is designed.

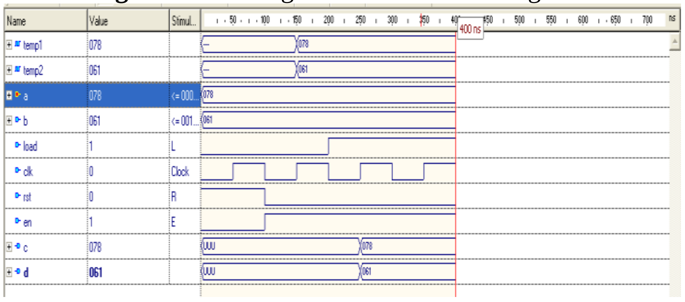
Figure 11: timing result for address Selection unit



Name	Value	Stimul...
pg1	000010101010	<= 010...
pg2	000000010001	<= 000...
clk	0	Clock
sel	0	S
rst	0	R
en	1	E
pg_addr	0001	
loc_addr	00000001	

Figure 11 illustrates the address selector results. The address line is selected based on the selection signal passed. Based on the page addressed and the location address a pointer is derived, which simultaneously locate the data location from the developed multi page memory unit. The 2 page contents in the multipage memory is shown in figure 11.

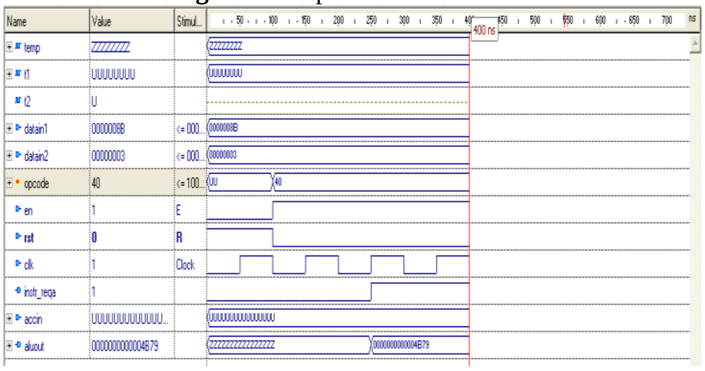
Figure :12 Timing result for Address Register



Name	Value	Stimul...
temp1	078	
temp2	061	
a	078	<= 000...
b	061	<= 001...
load	1	L
clk	0	Clock
rst	0	R
en	1	E
c	078	
d	061	

The operation timing result for address register used for addressing the memory address is illustrated in figure 12. The address register temporarily stores the address into temporary register, temp1,2 and read the addressed data to the output lines for processing.

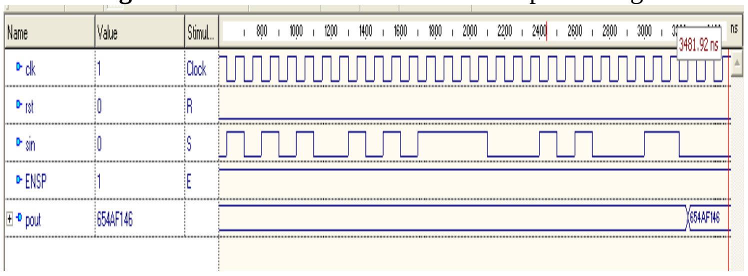
Figure: 13 Operation unit results



Name	Value	Stimul...
temp	ZZZZZZZZ	
t1	UUUUUUUU	
t2	U	
datain1	0000008B	<= 000...
datain2	00000003	<= 000...
opcode	40	<= 100...
en	1	E
rst	0	R
clk	1	Clock
instr_reqa	1	
accin	UUUUUUUUUUUUU...	
aluout	0000000000004B79	

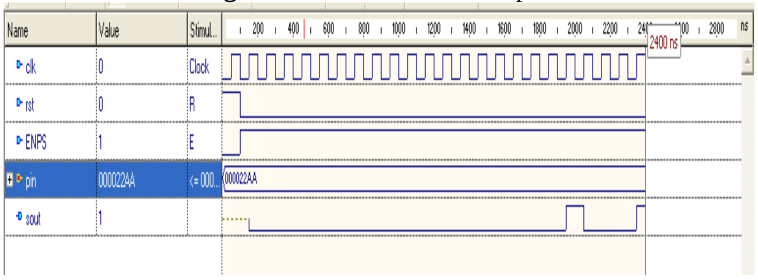
The simulation result for the developed operational unit is shown in figure 13. The results obtained are cross validated with the theoretical evaluation. The output of the addressable memory are processed for a arithmetic accumulation operation, where the two addressed data are buffered based on the instruction passed. The correlation accumulation results are seen on the output line.

Figure :14 Serial interface of bit stream processing



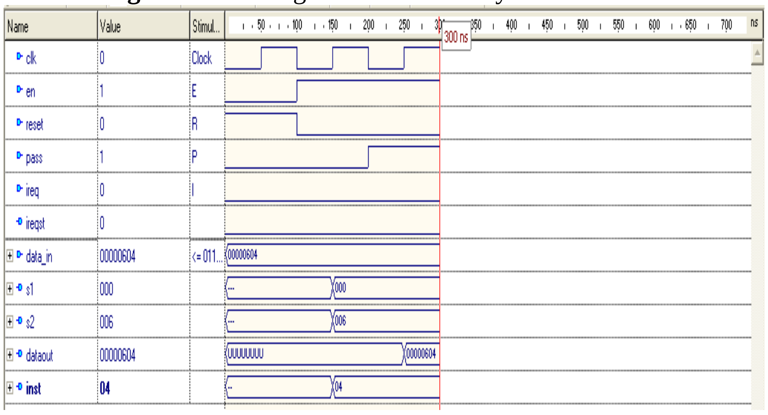
The serial interfacing of the processing unit is passed with serial stream of data. The interface unit converts the serial stream of data into parallel address bits, and pass to the addressing logic for memory interface.

Figure :15 BSP interface for parallel data



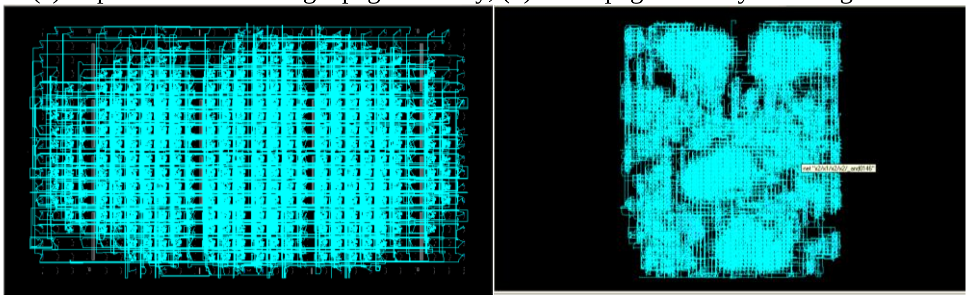
For the external device interface, such as USB, or computer networks, the processed parallel data need to be passed into serial port. The parallel to serial interface for the developed design unit, is shown in figure 15. The parallel data accumulated are serialized to the 'sout' port for the interface.

Figure :16 Timing result for Memory interface unit



The interface for the input and output unit of the developed multipage memory unit is shown in figure 16. The result illustrates the read and write operation on to the memory unit. The interface unit control the memory accessing by controlling the read and pass signal for the developed memory unit. The instruction for memory unit is passed over 'inst' signal, and 'datain' value is stored to the memory. On the write operation on addressing the buffered location is read back to the output port. To evaluate the implementation performance, the developed HDL code is passed to Xilinx synthesizer targeting to Xilinx-Virtex FPGA device (Xc2s50e-ft256-7). The obtained implementation results are illustrated in below section.

Figure 17 (a) Implementation of single page memory, (b) Multi page memory onto targeted FPGA device



A comparative analysis for the developed approach is carried out for the conventional single page memory and multi page memory unit. The performance metric of processing overhead is computed as,

Overhead = (volume of data accessed / Time taken to access the data) x 100

Table 1: Comparative analysis for the developed systems

Memory unit	Logical Blocks (CLB)	Device speed (MHz)	Power consumption (mW)	Accessing performance	Memory size	Processing overhead (%)
Single page memory	865	173.75	181	Single address	1Kb	65
Multi page memory	1478	121.56	213	Multi address	3Kb	34

The comparative analysis for the developed approach illustrates an improvement in the processing overhead optimization. As the result illustrates, targeting to the FPGA device, a multi page memory unit takes about 1478 logical values for its implementation, as the developed controller unit process for larger memory addressing and a page controller is used to address the multi page controlling, the single page has only 865 requirement. Hence, the operation speed is also observed to be 52MHz higher in single page as compared to multi page processing. However the offered memory size in multi page memory interface is about 2 times higher than the single page memory. Due to page addressing the overhead for a 3 page memory unit is 50% lesser as compared to a single page memory unit. This minimization in overhead result in 3 times faster memory accessing as compared to a single page memory interface. The overhead for the developed approach is lower making the system applicable to lowersource system interface.

V. Conclusion

The advantage of using multi page memory has given higher opportunity in processing large volume of data. However, the controlling operations were not suitable designed for a single page memory interface. In this paper, a interfacing unit to control the data accessing from a multi page memory unit to a processing unit is proposed. The approach defines a new interfacing controller unit, which generates a common addressing for multipage memory unit accessing multi pages simultaneously. The implementation overhead is observed to be higher due to higher memory pages. However, the processing overhead to this developed system is observed to be half as compared to single page memory interface due to faster memory accessing. This interfacing advantage gives an opportunity to use high volume storage and accessing under low resource devices. The controller unit in such design is observed to operate on multiple page addresses, hence distributed addressing of different operands get possible.

References

[1]. David E. Taylor. Edward W. Spitznagel "On using content addressable memory for package classification", Applied Research Laboratory,Washington University in Saint Louis, 2005
[2]. Kostas Pagiamatzis, Ali Sheikholeslami "Content-addressable memory (CAM) circuits and architectures: a tutorial and survey" , IEEE Journal of Solid-State Circuits, Vol.4l, No.3, March 2006
[3]. Scott Beamer, Mehmet Akgul "Design of low power content addressable memory (CAM) ", Department of Electrical Engineering & Computer Science, University of California, Berkley
[4]. Qutaiba Ibrahim "Design & implementation of high speed network devices using SRLl6 reconfigurable content addressable memory (RCAM)", International Arab Journal of e-Technology, VoI.2,No.2, June 2011
[5]. Enoch O. Hwang "Digital logic and microprocessor design with VHDL" , La Sierra University, Riverside
[6]. Jui-Yuan Hsieh, Shanq-Jang Ruan "Synthesis and design of parameter extractor for low-power pre-computation based content addressable memory using gate-block selection algorithm", Department of Electronics Engineering , Taipei , Taiwan
[7]. M. Arun, A. Krishnan "Comparative power analysis of pre-computation based content addressable memory", Journal of Computer Science 7(4):471-474,2011
[8]. Jinn-Shyan Wang "Low-power high-speed content addressable memories" , National Chung Cheng University, Taiwan

[9]. Haoyu Song, John W. Lockwood "Efficient packet classification for network intrusion detection using FPGA", International Symposium on Field-Programmable Gate Arrays (FPGA'05) ,Monterey ,CA, Feb 20-22, 2005

[10]. Due-Hung Le,MasahiroSowa,Cong-KhaPham,"A Fully –Parallel Infonnation Detection Hardware System Employing Content Addressable Memory," The Fourth International Conference in communication and electronics(lCCE) 20 12,pp.44 7 -452.

[11]. Due-Hung Le,Cong-KhaPham,Katsumi Inoue," A Novel CAM –Based Infonnation Detection Hardware System On FPGA" , The 8th Conference on PhD Research in Microelectronics & Electronics (Prime) 2012, pp.1-4

[12]. K. Pagiamtzis and A. Sheikholeslami, Pipelined match-lines and hierarchical search-lines for low-power content-addressable memories, in IEEE Custom Integrated Circuits Conference (CICC), September 2003, pp. 383-386.

[13]. An Shengbiol, Gao Shuangxi2, Wang Shuhail ,Chen Shuwangl,"CAM Design Based on Virtex Family Devices" The Eighth International Conference on Electronic Measurement and Instruments lCEMI'2007

[14]. K. Pagiamtzis and A. Sheikholeslami, "Content -addressable memory (CAM) circuits and architectures: A tutorial an survey," IEEE Journal of Solid-State Circuits, vol. 41, no. 3, pp. 712-727, March 2006.

Author Biographical notes:

K. Suresh Kumar is currently a Ph.D pursuing candidate in JNTUH Hyderabad, Telangana, India and he is also working as an Assistant Professor at the department of Electronics and Communication Engineering in SSJ Engineering College, Hyderabad, Telangana. He has seven years of teaching experience. He has done B.Tech in CVR College of Engineering, Hyderabad in 2007. He received his Master of Engineering in VLSI Design in Anna University, Tamilnadu, India. His current research interests include Low-power circuit design, fault tolerance.

IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) is UGC approved Journal with Sl. No. 5016, Journal no. 49082.

K.Suresh Kumar1 “Content Addressable Memory For Multi Page Memory Interface" IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) 13.3 (2018): 38-47.