An Overview of Multiple CAM Designs in Virtex Family Devices

XAPP 201, September 23, 1999 (Version 1.1)

Application Note: Jean-Louis Brelet

Summary

Flexible CAMs (Content Addressable Memory) are implemented in Virtex™ family devices by taking advantage of the reprogrammability of the basic LUT as a Shift Register (SRL16) or as a SelectRAM+™ memory and the fast carry logic chain. Although CAMs are also feasible in Spartan™ and XC4000X devices, this application note concentrates on Virtex™ and Virtex™-E devices. The flexibility of Virtex devices is a key advantage in designing a CAM, with the application determining the best implementation. As an overview, this application note references XAPP202 “CAM Designs in ATM Applications”, XAPP203 “Designing Flexible, Fast CAMs with Virtex Family FPGAs”, and XAPP204 “Using Virtex Block SelectRAM+ for High Performance Read/Write CAMs”.

Xilinx Family

Virtex and Virtex-E FPGAs

Introduction

A Content Addressable Memory is a storage array designed to quickly find the location of a particular stored value. By comparing the input against the data in memory, a CAM determines if an input value matches one or more values stored in the array. If the comparison is done simultaneously, the CAM is said to be performing at maximum efficiency. If a match exists, it is found in a single clock cycle.

Similar to a RAM, a CAM stores words in an array. The CAM write mode is comparable to a RAM, but the CAM read mode is different. In a RAM, the word in a specific location is read by the address. In a CAM, the data on the input is looking for a match. When a match is found, the output is the address in the array.

The number of address lines limits a RAMs data size. In a typical example, a 10-bit bus addresses 1024 locations of 8-bit data. A CAM does not have this limitation because it does not use an address bus to read a location. To find a match of an 8-bit value in 1024 locations, an 8-bit bus on the input is required. When the data is found in the CAM, a match signal goes active. The output is the matching data address. Because a CAM does not need address lines to find data (read mode), the memory size can be easily expanded. The width is determined by the storage and comparator size. Figure 1 compares a RAM and a CAM in read mode.

The basic core of a CAM has a storage location and a comparator between the storage location value and the input data. This application note describes the different ways to design a basic core optimized for either speed, density, or both.

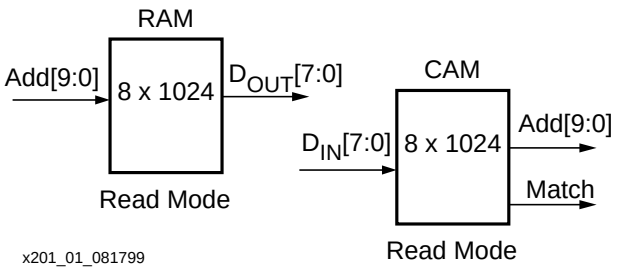

Figure 1: Simple RAM and CAM compared

Typical CAM applications

CAMs are used in telecommunications, networking, Ethernet, ATM switches, and diverse protocol applications. To determine the correct CAM implementation for a particular application, the following should be investigated.

- Word Size (width)
- Number of Words (depth)
- Match or Compare time (read)
- Significance of Write Speed
- Clock Frequency
- Masks and Outputs

Virtex family devices offer the advantage of flexible approaches to designing an optimal CAM. There is not a specific CAM type appropriate for all typical CAM applications. Flexible approaches can address an optimal trade-off between speed and internal device area.

CAM Overview

An AND gate with an optional inverter on each input is a comparator. Each 4-input LUT is a 4-bit comparator. The LUT and the dedicated carry chains can be used for designing wide AND gates. An 8-input wide AND with an optional inverter on each input is a comparator between 8-bit data and pre-encoded data. The output of the fast carry chain is the result of the comparison (output of the wide AND gate). An array of 16 AND gates represents a CAM of 16 words (depth) by eight bits (width), see Figure 2. Taking advantage of the high-performance routing capabilities, the data bus distributes data to each AND gate and the comparison is run concurrently in only one clock cycle. By using reprogrammable LUTs to implement each 4-input AND gate, the AND gate becomes a basic 4-bit storage element or CAM. This concept is also applicable to the true Dual Read/Write Port™ feature of Virtex 4K-bit Block SelectRAM+ memory to pre-decode 16 words of eight bits in each block. As with a LUT solution, Block SelectRAM+ memory is cascadable to implement larger CAMs.

CAM designs in Virtex devices

Three CAM designs are briefly compared in this application note. These designs are described in detaill in separate application notes with accompanying HDL reference designs. The designs are based on Virtex-specific device features including fast dedicated carry chains, distributed RAM, built-in shift registers (SRL16E) and Virtex Block SelectRAM+ memory.

Basically, there are three different ways to implement a CAM in Virtex devices.

- A design with single cycle read but with slow write access (16 clock cycles) - XAPP203
- A design optimized for large width and depth but with slow read access (16 clock cycles) - XAPP202
- A design with a single clock read and write but limited to an 8- bit data width per Block SelectRAM+ memory - XAPP 204

Basic CAM design

A read-only CAM is equivalent to a decoder (an AND gate with inverted or non-inverted inputs) for each word to compare. Figure 2 is a basic read-only CAM. The carry logic is used to implement the wide OR gates or AND gates in the Virtex CLBs.

Because the Virtex family devices are SRAM based and the Virtex LUT is programmable as distributed SelectRAM+ or Shift Register SRL16E, the decoder can be re-configured. The re-configuration is the CAM write operation. One Virtex LUT stores four bits and is a 4-bit comparator at the same time. Figure 3 is a comparator in a Virtex slice.

The word width is expandable with a minor timing penalty (each four bits add a one MUXCY delay). The depth is also expandable, since the number of simultaneous comparisons is flexible in Virtex devices. Figure 4 details a CAM with 8-bit words matched. A LUT configured as a Shift Register SRL16E is used in Virtex designs to store and compare each CAM word. A LUT configured as a SelectRAM+ memory is used in either Virtex, XC4000X, or Spartan families.

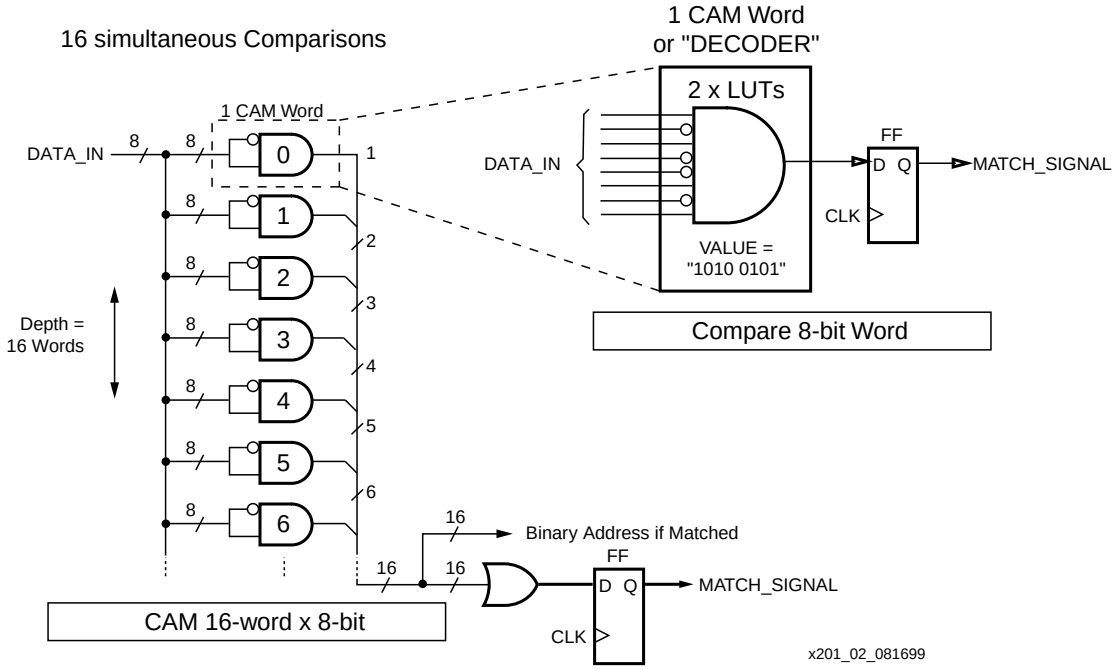

Figure 2: Basic READ only 8-bit word CAM

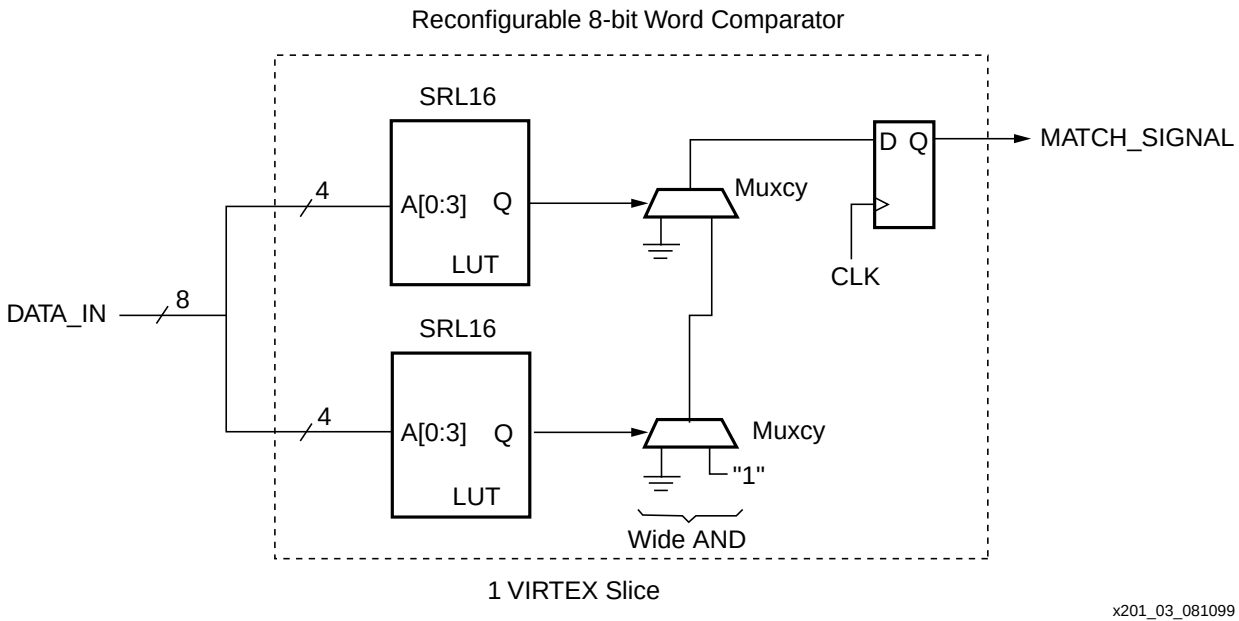

Figure 3: Comparator from a Virtex Slice

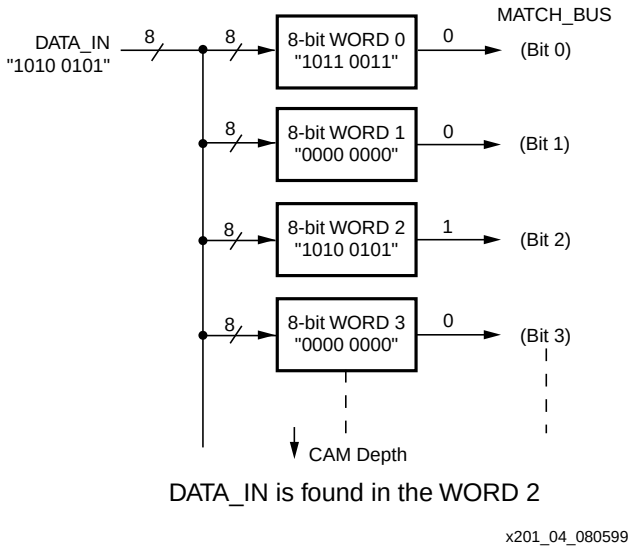

Figure 4: 8-bit CAM words matched

CAM implementations

Definitions

The CAM basic unit is a storage element (x-bits wide) and an input value comparator.

Read efficiency is measured as the number of bits per LUT per cycle. The largest value is the most efficient in terms of performance. The fastest CAM detects a match or absence of a match in one clock cycle.

A CAM “nb_word” x “nb_bit” represents a “nb_bit” bits width and a “nb_word” words depth.

Design Examples

Table 1 lists design examples of various CAM sizes and performance specifications. Some of the largest CAM examples are a useful indicator of the breadth of the Virtex architecture. The following descriptions of each reference design help determine the best solution for specific CAM requirements. Implementation results can be optimized when the CAM is instantiated in the user design, showing the percentage of slices used in a Virtex device.

Table 1: Virtex Family Reference Design Comparison

Reference Design	Depth (number of words)	Width (number of bits)	CAM Block (bits)	Read		Write	number of SRL16E or RAM16x1	number of BlockRAM	number of TBUF
				Match	Encode				
XAPP 204	32	8	256	4.5 ns	11.5 ns	2 x 11.5 ns	16	2	
XAPP 204	128	8	1K	5.5 ns	15 ns	2 x 15 ns	64	8	32
XAPP 204	256	8	2K	8.5 ns	19 ns	2 x 19 ns	128	16	64
XAPP 203	16	16	256	7.5 ns	7.5 ns	16 x 8.5 ns	64		
XAPP 203	32	16	512	8 ns	8 ns	16 x 10 ns	128		
XAPP 203	128	40	5K	12 ns	12 ns	16 x 14 ns	1280		32
XAPP 203	256	24	6K	12.5 ns	12.8 ns	16 x 15 ns	1536		64
XAPP 202	256	16	4K	16 x 12 ns	12 ns	12 ns	256		
XAPP 202	4096	16	64K	16 x 20 ns	20 ns	20 ns	4096		

SRL16 design

Fast match CAM in SRL16 primitives (one read per clock cycle) - Application Note XAPP203. A typical application is a protocol selection.

- Implementation = four bits per LUT (SRL16E or distributed SelectRAM memory + MUXCY)
- Speed = 115 MHz for read and write and over 133 MHz access time (Match)
- Read = one clock cycle
- Write = 16 clock cycles
- Efficiency = four bits/LUT/cycle (Read)

Application examples include address decoders/encoders and control logic.

- XCV50 implementation = CAM 32 x 16, (26% of the slices), 86 LUTs, 128 shift registers (or LUTs), and 43 slice registers
- XCV300 implementation = CAM 128 x 40, (49% of the slices), 313 LUTs, 1,280 shift registers (or LUTs), and 141 slice registers
- XCV400 implementation = CAM 256 x 24 (42% of the slices), 585 LUTs, 1,536 shift registers (or LUTs), and 270 slice registers

The basic element is a LUT. It stores four encoded bits, allowing direct comparison to the input data. All LUT outputs for each bit of the word are AND-wired in the Virtex carry logic chain. Extra control logic and counters are used to write to the LUT.

This design is described in Application Note XAPP203 and an HDL reference design is available.

Note: If a dynamic mask is required, the implementation is two bits per LUT. A pre-defined mask at write time maintains four bits per LUT.

CAM for ATM

Optimized CAM (16 reads per clock cycle) - Application Note XAPP202. ATM applications require large CAM and may be able to compromise on several clock cycles to find a match.

- Implementation = 10 bits per LUT (Distributed RAM + LUT + MUXCY)
- Speed = 80 MHz
- Read = 16 clock cycles
- Write = one clock cycle
- Efficiency = 0.63 Bits/LUT/cycle (Read)

Application examples:

- XCV50 implementation = CAM 256 x16 (33% of the slices), 224 LUTs, 256 RAM16x1 (or LUTs), and 46 slice registers.
- XCV400 implementation = CAM 4096 x 16 (83% of the slices), 3,298 LUTs, 4096 RAM 16 x 1 (or LUTs), and 261 slice registers.

A SelectRAM+ memory cell (RAM16x1s) is the basic element used to store data. The write operation is similar to any classic write operation in SelectRAM+ memory. The comparator is built in the LUT and wired-AND in the carry chain to generate the match signal.

A key element of this design is the Virtex distributed SelectRAM+ memory and the fast carry chain.

This design is described in Application Note XAPP202 and an HDL reference design is available.

CAM using BlockRAM memory

Fast Block SelectRAM+ CAM: CAM16x8 - Application Note XAPP204. This solution is optimal for applications requiring one or two clock cycles for both read and write.

- Implementation = Block SelectRAM+ memory (CAM16x8 in each Block SelectRAM+ memory)
- Speed = 90 MHz for read and write and over 200 MHz access time (Match)
- Read = one clock cycle
- Write = one clock cycle after one erase cycle
- Efficiency = 128 bits/SelectRAM+/cycle (Read)

Applications examples:

- XCV50 implementation = CAM32x8
  32 LUTs, 16 RAM16x1 (or LUTs), two SelectRAM+ blocks and eight slice registers.
- XCV50 = CAM 64 x 8 in one column (SelectRAM+ blocks and adjacent CLB), 57 LUTs, 32 RAM16x1 (or LUTs), 16 TBUFs, four BlockRAMs, and nine slice registers.
- XCV300 implementation = CAM 128 x 8 in one column (SelectRAM+ block and adjacent CLB), 111 LUTs, 64 RAM16x1 (or LUTs), 32 TBUFs, eight SelectRAM+ blocks, and ten slice registers.
- XCV1000 implementation = CAM 256 x 8 in one column (SelectRAM+ block and adjacent CLB), 226 LUTs, 128 RAM16x1 (or LUTs), 64 TBUFs, 16 SelectRAM+ blocks, and 11 slice registers.

The Virtex library has an equivalent to a CAM16x8 primitive. (CAM 16 words by eight bits Synchronous.)

A Virtex Block SelectRAM+ primitive, RAMB4_S1_S16, also represents a 1-cycle read, 1-cycle write CAM16x8 (width = 8 bits and depth = 16 words).

The Virtex Block SelectRAM+ primitive can be a simultaneous 16-output decoder of 8-bit words because it has the unique capability of having each port independently configured to a specific data width.

This design is described in Application Note XAPP204 and an HDL reference design is available.

Conclusion

Unique Virtex device features have key advantages in providing broad system level solutions. These features also offer flexibility in the approach to CAM designs. CAM solutions are similar to the different RAM solutions. A small RAM can be implemented by using the distributed SelectRAM+ features. A medium-size RAM is addressed with the Virtex Block SelectRAM+ feature. CAM requirements can be addressed by the following methods.

- SRL16E-based implementations offering large word widths and high performance. A reference design is in application note XAPP203.
- Distributed SelectRAM-based implementations offering large word width and depth adapted specifically for ATM applications. A reference design is in application note XAPP202.
- Block SelectRAM+ memory-based implementations offering read and write, high performance, and 8-bit width. A reference design is in application note XAPP204.

These application notes will assist the designer in determining the best Virtex device solution to their CAM system needs.

Revision History

Date	Revision	Activity
8/19/99	1.0	Initial Release
9/23/99	1.1	Initial Virtex-E update

The Programmable Logic Company℠

Headquarters

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
U.S.A.

Tel: 1 (800) 255-7778
or 1 (408) 559-7778
Fax: 1 (408) 559-7114

Net: hotline@xilinx.com
Web: http://www.xilinx.com

North America

Irvine, California
Tel: (949) 727-0780

Englewood, Colorado
Tel: (303) 220-7541

Sunnyvale, California
Tel: (408) 245-9850

Schaumburg, Illinois
Tel: (847) 605-1972

Nashua, New Hampshire
Tel: (603) 891-1098

Raleigh, North Carolina
Tel: (919) 846-3922

West Chester, Pennsylvania
Tel: (610) 430-3300

Dallas, Texas
Tel: (972) 960-1043

Europe

Xilinx Sarl
Jouy en Josas, France
Tel: (33) 1-34-63-01-01
Net: frhelp@xilinx.com

Xilinx GmbH
München, Germany
Tel: (49) 89-93088-0
Net: dlhelp@xilinx.com

Xilinx, Ltd.
Weybridge, United Kingdom
Tel: (44) 870-7350-603
Net: ukhelp@xilinx.com

Japan

Xilinx, K.K.
Tokyo, Japan
Tel: (81) 3-5321-7711
Net: jhotline@xilinx.com

http://www.xilinx.com/support/techsup/japan.htm

Asia Pacific

Xilinx Asia Pacific
Hong Kong
Tel: (852) 2424-5200
Net: hongkong@xilinx.com

© 1999 Xilinx, Inc. All rights reserved. The Xilinx name and the Xilinx logo are registered trademarks, all XC-designated products are trademarks, and the Programmable Logic Company is a service mark of Xilinx, Inc. Other Xilinx registered and non-registered trademarks are as listed at http://www.xilinx.com/legal.htm. All other trademarks and registered trademarks are the property of their respective owners.

Xilinx products are manufactured under one or more of the patents listed at http://www.xilinx.com/legal.htm. Xilinx, Inc. does not assume any liability arising out of the application or use of any product described herein; nor does it convey any license under its patents, copyrights, or maskwork rights or any rights of others. Xilinx., Inc. reserves the right to make changes, at any time, in order to improve reliability, function, or design and to supply the best product possible. Xilinx, Inc. cannot assume responsibility for the use of any circuitry described other than circuitry entirely embodied in its products. No other circuit patent licenses are implied. Xilinx, Inc. will not assume responsibility for any circuits shown nor represent that they are free from patent infringement or of any other third-party right. Xilinx,. Inc. assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction if such be made. Xilinx, Inc. will not be liable for the accuracy or correctness of any engineering or software support or assistance provided to a user.