XAPP260 (v1.1) February 27, 2002

Using Virtex-II Block RAM for High Performance Read/Write CAMs

Author: Jean-Louis Brelet and Lakshmi Gopalakrishnan

Summary

Content Addressable Memory (CAM) offers increased data search speed. In various applications based on CAM, there are differing requirements for data organizations and read/write performance. The innovative design described in this application note is suited for small embedded CAMs with high-speed match and write requirements. The reference design is built using the true dual-port block SelectRAM+™ feature for the Virtex™-II series, including the Virtex-II Pro™ devices.

Introduction

The block SelectRAM+ memory built into the Virtex-II devices can be used as a 32-word deep by 9-bit wide (32 x 9) CAM using the innovative design techniques described in this application note. **XAPP204** explains how to implement a 16-word by 8-bit wide (16 x 8) CAM in a Virtex-II block RAM. A reference design (see Appendix A) provides parameterizable Verilog and VHDL code to cascade several block RAMs configured as 32 x 9 CAM. CAM speed is equivalent to the access time of a Virtex block RAM for a single clock cycle match (read), and a one or two clock cycles write.

Medium size CAMs can be implemented in Virtex slices with different design techniques.

CAM32x9 Macro

Features of the CAM32x9 include:

- 288 bits
- 32-word x 9-bit organization
- Independent match (read) and write data input buses
- Decoded address output (or 32-bit “one-hot” decoded address)
- Fully synchronous match port (or read port)
- Fully synchronous write port
- Single clock cycle match (single or multi-matches)
- Single clock cycle write (and single clock cycle erase)
- Match enable input
- Write enable input
- Write or erase mode (assert to “one” if write, to “zero” if erase)
- Reset match port (forces the output bus to zero synchronously)
- Cascadable
- Initialization during device configuration (empty by default)
- Fits in one Virtex-II block SelectRAM+ memory

Figure 1 shows a CAM32x9 macro built on the true dual-port block SelectRAM+ memory. Write port A is independent of read port B. Both ports are fully synchronous relative to their respective clock.

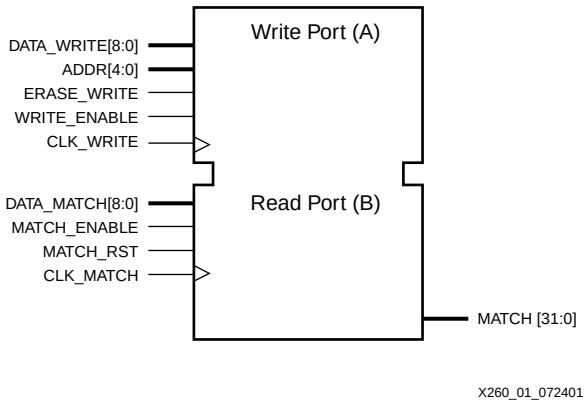

Figure 1: **CAM 32x9 Macro**

Virtex-II FPGAs facilitate different CAM sizes based on the number of block RAMs in each device. Using the cascadable CAM32x9 macro, Table 1 shows possible combinations of CAM organization in Virtex-II devices.

Table 1: **Maximum CAM Width and Depth in Virtex-II Devices**

Virtex-II Device	Number of Block SelectRAM	Maximum CAM Size (bits)	Maximum CAM Width (bits)	Maximum CAM Depth (bits)
XC2V40	4	1152	32x36	128x9
XC2V80	8	2304	32x72	256x9
XC2V250	24	6912	32x216	768x9
XC2V500	32	9218	32x288	1024x9
XC2V1000	40	11520	32x360	1280x9
XC2V1500	48	13824	32x432	1536x9
XC2V2000	56	16128	32x504	1792x9
XC2V3000	96	27648	32x864	3072x9
XC2V4000	120	34560	32x1080	3840x9
XC2V6000	144	41472	32x1296	4608x9
XC2V8000	168	48384	32x1512	5376x9

Because CAM output is a decoded address (similar to a 32-bit “one-hot” decoded address), the depth is expandable without additional logic. Each location has a single address bit output. When data is present at a particular address, the corresponding address line is asserted High. When data is not present, the address line is Low. A decoded address output allows multiple matches, *i.e.*, the same data might be found in several locations in only one clock cycle. The write mode is similar to writing 9-bits of data in one of the 32-word x 9-bit memory blocks.

Figure 2 shows the read mode of a 128-word x 9-bit CAM built on four block SelectRAM+ primitives. The 128-bit decoded address bus is generated by merging four 32-bit decoded addresses.

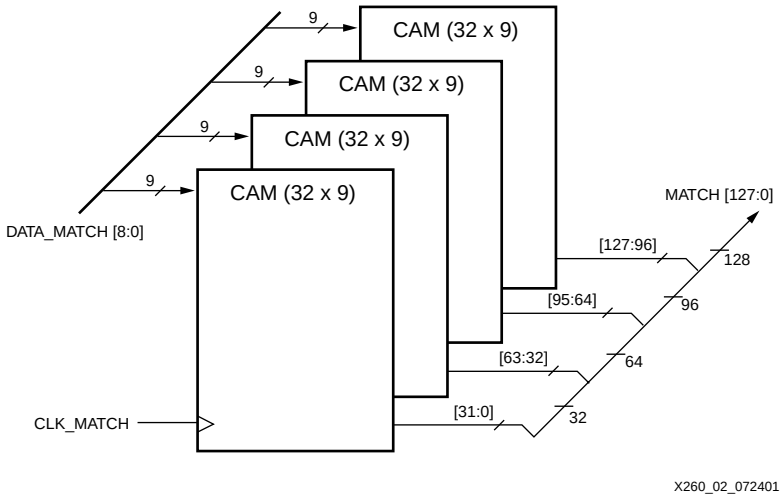

Figure 2: **CAM 128-word x 9-bit in Read Mode**

In a write mode, a 7-bit bus is used as an address for the 128 x 9 CAM. This address bus is composed of a 5-bit bus to each set of CAM32x9 address inputs and a 2-bit bus decoded to select one of the CAM blocks.

To write 9-bits of data to one of the 128 locations (7-bit address bus), one of the four CAM32x9 blocks is selected with the two MSBs of the 7-bit address bus. The five LSBs are connected to the regular address inputs ADDR[4:0] of each block.

The width of the design is also expandable with extra AND gates. Each CAM32x9 contains only 9-bits of the total word width. Two 32x9 CAMs allow for a 18-bit width, with 9 bits stored in the first CAM32x9 block and 9 bits in the second block. A match is found only if both 9-bit locations match the specified 18-bit data. A 2-input AND gate for each CAM output signal provides the final decoded address. Figure 3 shows a 32-word x 32-bit CAM in read mode, built with two block SelectRAM+ primitives

Because the basic CAM32x9 macro has a “one-hot” decoded address, both CAM depth and width are simultaneously expandable to build the desired CAM size while providing the same input/output features.

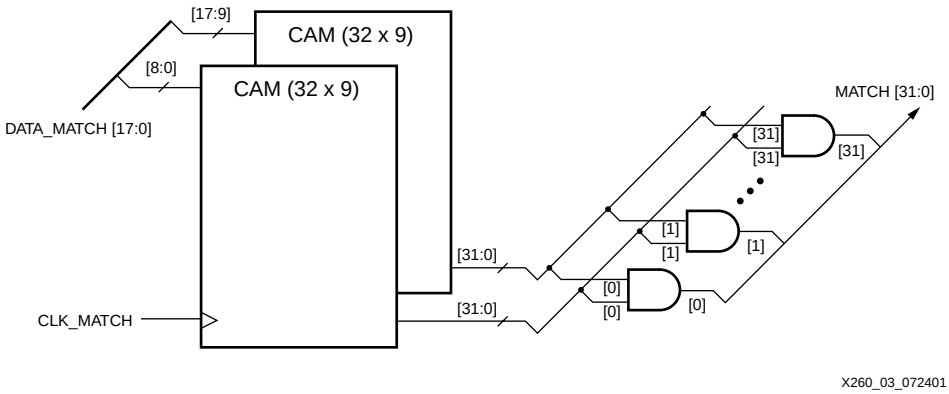

Figure 3: **CAM 32-word x 18-bit in Read Mode**

To achieve an optimal performance when cascading CAM32x9 blocks, the number of block SelectRAM+ memories per Virtex-II column has to be considered. The number of blocks per column depends on the Virtex-II device.

Applications

An embedded CAM block is useful in many applications including networking designs and data processing. A small CAM can provide instantaneous protocol detection, parallel processing choices, or networking and router filtering.

A multi-protocol application typically requires different treatments of an input data stream. A selection is made based on the protocol type, generally encoded in a data header. An embedded CAM detects, in one clock cycle, the right protocol by searching an array of possible protocols. The designer must store all these protocols in the CAM block and can dynamically modify this list. When a protocol is found in the CAM, the corresponding address selects further operations.

Parallel processing or data recognition can benefit from embedded CAM blocks as well. Because the search for a matching data is performed in one clock cycle, even if multi-matches are found, CAM blocks accelerate those applications.

When the application uses a large data width, selected data bits (a smaller data width) are often sufficient to search the CAM block. An associated block RAM can store and retrieve the complete data. The scalable size of the CAM blocks in Virtex-II devices and the flexible features offer many solutions to the designer. The concurrent access to search for data in a single clock cycle guarantees a high-speed result.

General Description

The unique Virtex-II block RAM approach is used to build the CAM32x9 block. This methodology is based upon the true dual-port feature of the block SelectRAM+ memories. Ports A and B can be configured independently, anywhere from 16K-word x 1-bit to 512-word x 32-bit. Each port has separate clock inputs and control signals. The internal address mapping of the block SelectRAM+ memory is the primary feature in designing a CAM in a true dual-port block RAM. Each port accesses the same set of 16K memory locations using an addressing scheme dependent on the port width. Table 2 shows the low-order address mapping for a 1-bit and a 32-bit port width.

Table 2: **Block SelectRAM+ Port Address Mapping**

Port Width	Port Addresses																	
1	16384...	31...	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32	512...	00																

This design technique configures port A as 16K-word by 1-bit wide and port B as 512-word by 32-bits wide. The block SelectRAM+ primitive in this particular configuration is named RAMB16_S1_S36. Each port contains independent control signals. Port A is the CAM write port, and port B is the CAM read or match port. Both the read and write CAM ports are fully synchronous and have dedicated clock and control signals.

Decoded 9-bit Data in a Block RAM

A 9-bit data has 2^9 (= 512) possible values. A classic RAM stores the 9-bit data into a 9-bit location. However the 9-bit data can also be represented as a 512-bit word, with all zeros and a single “one” at the nth location, where n corresponds to the position given by the decoded 9-bit data. For example, if the data is “000000111” (a decimal seven), the decoded 512-bit word is “0000....000010000000”, where the “one” is at the seventh location counting from zero. Thirty-two 512-bit words store 32 decoded 9-bit words. As shown in Figure 4, an array of 32 x 512 represents 32 addresses from 0 to 31.

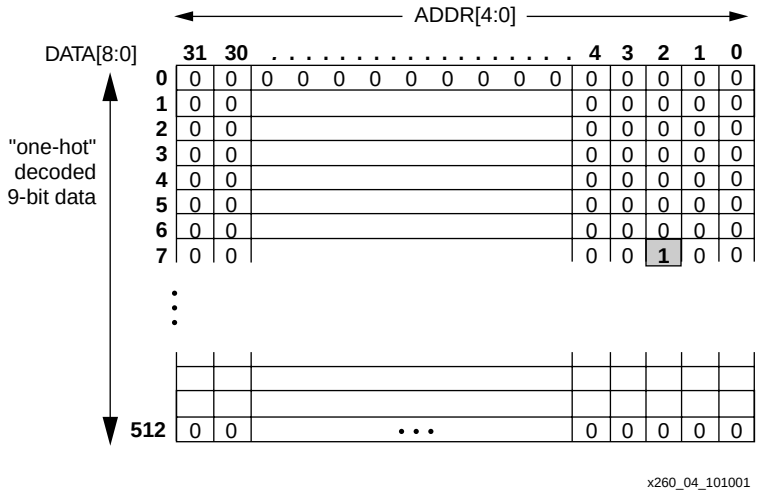

Figure 4: **CAM32x9 Port Address Mapping**

Finding a Match

The 16K-bit RAMB16_S1_S36 with a 32-bit wide data port (port B), and a 4-bit wide port for parity, generates 32 values simultaneously. If the 9-bit data (DATA_MATCH) to be searched is connected to the 9-bit address (ADDRB) of port B as shown in Figure 5, the 32-bit port B generates the matches concurrently. Using the fact that a particular location corresponds to the decoded 9-bit data, the matching operation is equivalent to searching 32 locations for specific 9-bit data at the same time. Figure 5 shows the RAMB16_S1_S36 port B configured as a CAM read port.

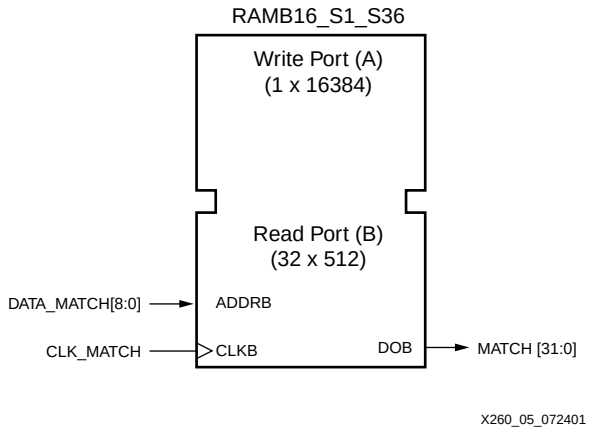

Figure 5: **Read Port of a CAM32x9 in a SelectRAM+ Block**

Reading into the CAM32x9 Example

If the data “0000 00111" was previously written at the address “00010” of the CAM32x9, a match operation is equivalent to a read operation of a block RAM with the 9-bit data “0000 0111" placed on the address input of port B (ADDRB[7:0]). The 32-bit port B output is “0000 0000 0000 0000 0000 0000 0000 0100" corresponding to a match found at location 2, as shown in Figure 6. Port B output is the decoded CAM address bus. If no match is found, the output is “0000 0000 0000 0000 0000 0000 0000 0000." If one or several matches are found, each corresponding location equals “one”. In this last case, the same 9-bit data has been stored at different addresses, and the CAM32x9 output is multi-matches.

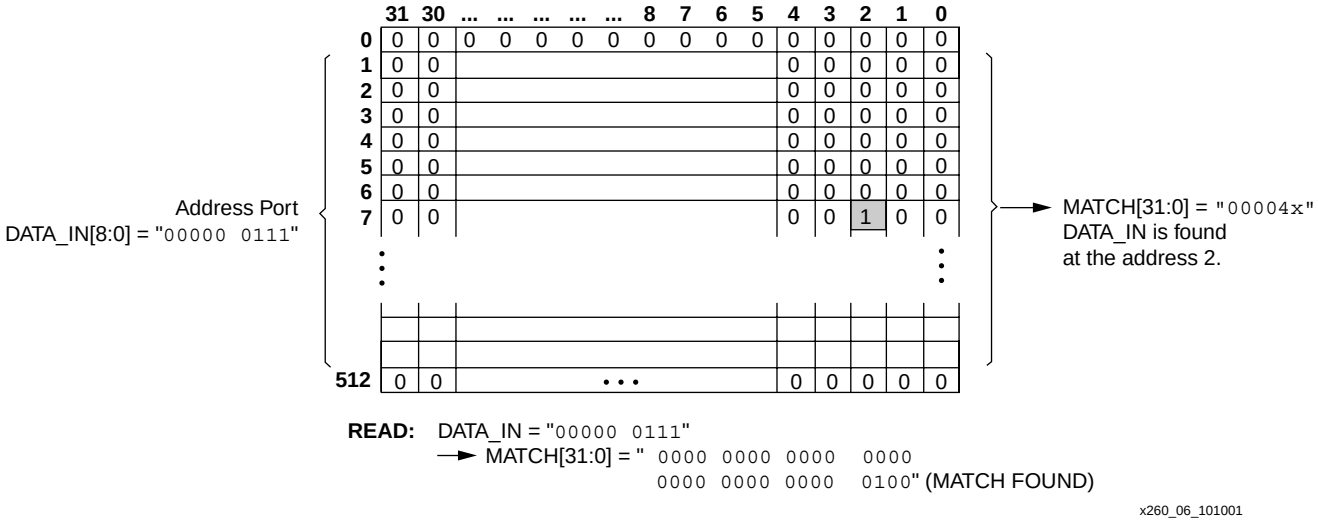

Figure 6: **An Example of a Read Into the CAM32x9 Block**

Writing into the CAM32x9

The CAM write port inputs are a 9-bit data bus, an address bus (five bits to address the 32 locations), control signals, and the clock. The 5-bit address bus selects a memory location. Writing new data into this location is equivalent to decoding the 9-bit data into a 512-bit “one-hot” word and storing the 512-bit word. However, if the CAM32x9 macro is initialized to zero, only one bit of the 512-bit word has to toggle. The location of the “one” is determined by the “one-hot” decoded 9-bit value.

Taking into account that the address port of the block SelectRAM+ primitive decodes the address bus, both operations previously described are combined into a simple write in the block RAM.

Port A configured as 16384 x 1 has a 1-bit data input and a 14-bit address input. The data input is asserted to “one”, and the 9-bit data plus the 5-bit address are merged in a single 14-bit address input.

With the 9-bit data as MSB and 5-bit address as LSB, the resulting 14-bit address input decodes the 9-bit data and selects one of the 32 memory locations simultaneously. The clock edge stores a “one” at the corresponding location. The Figure 7 shows the write port of the 32 x 9 CAM in a block SelectRAM+ primitive.

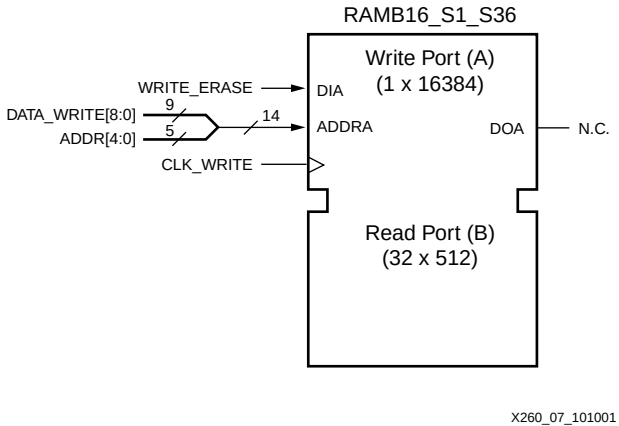

Figure 7: **Write Port of the CAM32x9 in a SelectRAM+ Block**

Erasing Data

To erase previously stored data, the selected location must be initialized to “zero” (512-bit word). The basic option is to write “zero” during 512 clock cycles by incrementing the 9-bit MSB of the address input from 0 to 511. The 5-bit LSB of the address input is fixed and is used to select the CAM location.

The solution presented in the reference design requires a single clock cycle. The erase operation is equivalent to the write operation with the exception that a “zero” is stored instead of a “one”. The port A address input is again a combination of the 9-bit data to be erased and the 5-bit address bus. To perform the correct selection, the 9-bit data must be stored into a separate RAM block. The reference design shows a methodology using nine SelectRAM+ blocks (32 x 1 RAM in nine LUTs) to memorize the 32 words.

The choice between a no erase mode (the application doesn't need to delete the CAM data) and an erase mode is open to the designer. In the second case, two solutions offer different trade-offs between speed vs. Virtex-II slice area).

Performance of the 32 x 9 CAM32x9

A single clock cycle generates the 32-bit decoded address. A match operation on the CAM32x9 block is only a read operation into the SelectRAM+ block, and has the same performance. The write operation uses the standard SelectRAM+ write mode. The Virtex-II series data sheets detail the SelectRAM+ timing characteristics. Figure 8 shows the basic implementation of the CAM32x9 macro, built on a RAMB16_S1_S36 primitive.

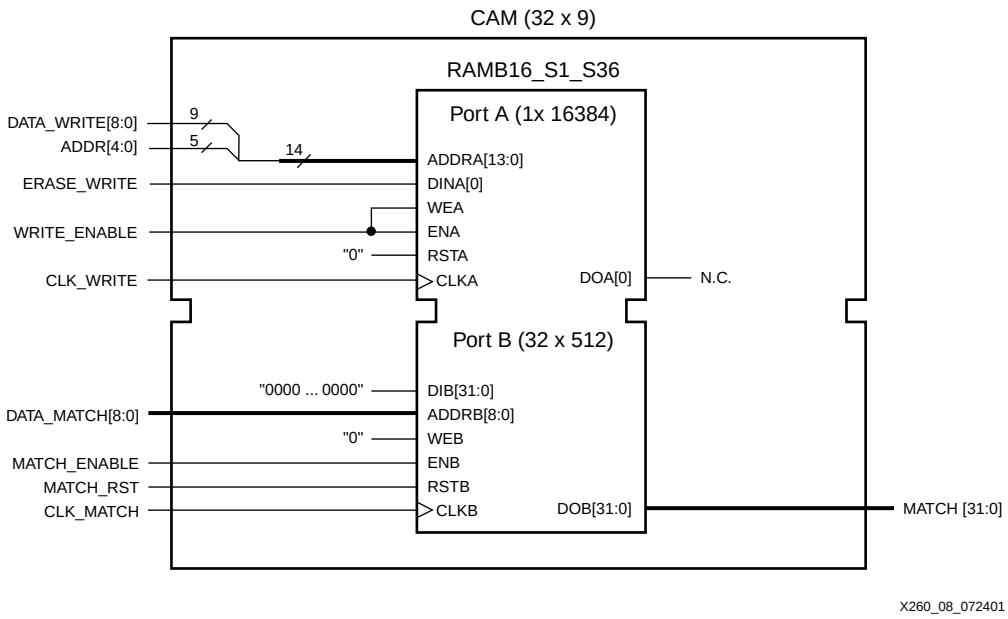

Figure 8: **CAM32x9 Macro Built on a RAMB4_S1_S16 Primitive**

Design Overview

The reference design described in this application note has an adjustable word depth. The output of the basic CAM32x9 macro is a decoded address generated in a single clock cycle. The number of output lines equals the number of words in the CAM. It is a multiple of 32, which represents the number of words in each CAM32x9 macro.

This reference design proposes some modules for generating the encoded output address. Because of the hierarchical VHDL code structure, it is easy to use this module for various CAM sizes according to the designers need along with a choice between LUT and TBUF implementations. A wide OR gate of all the decoded addresses creates a match flag asserted to one when a match is found.

Write Operation

This reference design implements a one clock cycle erase plus a one clock cycle write solution. The data stored in the CAM is also stored in RAM32x1 primitives and read back to erase the CAM. Figure 9 shows the write/erase configuration. The 32x1 primitives utilize the synchronous distributed RAM resources.

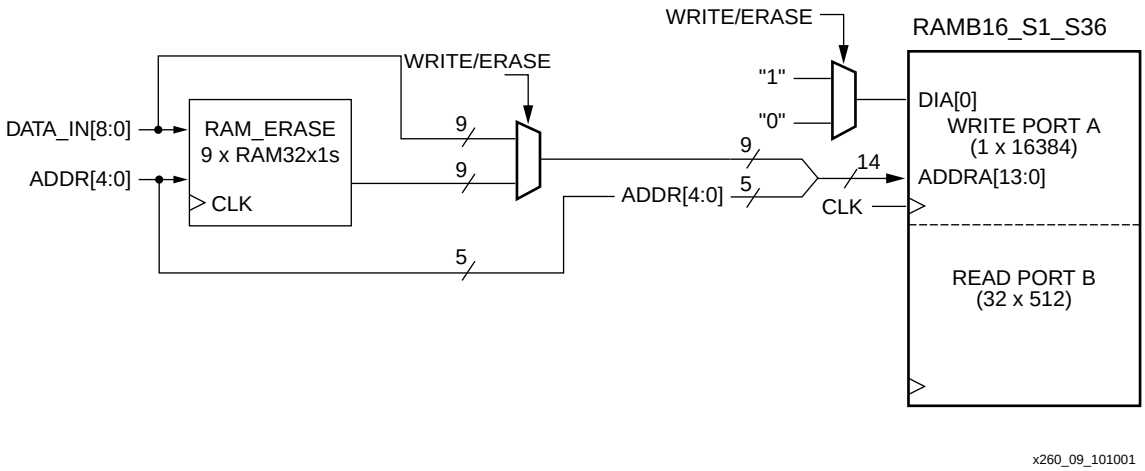

Figure 9: **CAM32x9 One Clock Cycle Erase and Clock Cycle Write**

Read Operation

The read or match operation is performed on port B. The decoded addresses are encoded in either a full logic implementation in a LUT, or a 3-state buffer TBUF implementation.

Built on the outputs of the CAM32x9 macro, the reference design describes how to generate a match signal and how to encode the CAM address output. Figure 10 shows a CAM32x9 with an additional wide OR gate generating a match flag. The MATCH_SIGNAL is asserted if one of more matches are found.

The 32-bit decoded address bus (MATCH) is encoded into a 5-bit address bus (MATCH_ADDR), which represents a valid address only if a single match is found. The reference design proposes to generate an additional signal, asserted to “one” when a single match occurs. For multiple matches, the reference design utilizes the 32-bit unencoded output from each CAM32x9. The user can then encode all the match addresses, depending on the position of 1s in the MATCH output. For example, if the CAM 32x9 MATCH output is “0000 10010001 0000 0001 1000 0000 0011,” then the 9-bit input data is available at locations 0, 1, 11, 12, 20, 24, and 27 of the CAM32x9.

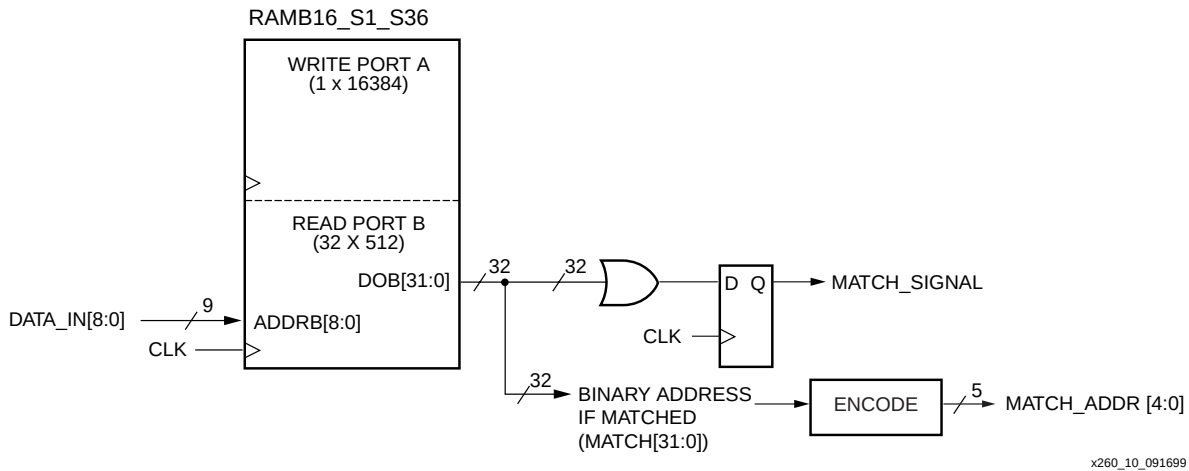

Figure 10: **CAM32x9 with a Match Flag and Encoded Outputs**

Designing a CAM32x9 in Virtex-II Block SelectRAM+ Primitives

With this application note and the reference design file XAPP260.zip, a designer can implement a fast read and write CAM module. The VHDL or Verilog code offers many possibilities to adapt this example to specific applications.

Features

- High-performance single clock cycle match (read)
- High-performance single clock cycle write or erase
- Generic “nb_cam32x9s” defining the number of CAM32x9 macros. The CAM depth is a multiple of 32. The smallest required Virtex-II device depends on the number of block SelectRAM+ memories (see the Table 1).
- Generic “addr_width” defining the number of address lines directly tied to the “nb_cam32x9s”. A couple of examples are a 64-word CAM requiring seven address lines and a 128-word CAM requiring six address lines.

Basic Building Block: CAM

The basic building block CAM32x9 is built on the block SelectRAM+ primitive as explained in the general description. In this design, only one 9-bit data bus is used on both the write port and the read port (DATA_IN[8:0]), and both ports share the same clock. The VHDL or Verilog code instantiating the block SelectRAM+ primitive is named Init_RAMB16_S1_S36 and provides an initialization example. By default, the CAM is empty, initialized to zeros.

Associated with this first module, a second basic building block is named Init_9_RAM32x1s. This module instantiates the nine RAM32x1 primitives to store the 32 words in a standard RAM and is used in the reference design to erase a location of the CAM32x9 in one clock cycle. As with the Init_RAMB16_S1_S36 module, the VHDL or Verilog code provides initialization (all zeros by default). The designer should initialize both modules with the same data for correct functionality.

The CAM_RAMB16 module instantiates the two previous modules to create a complete building block. The CAM_RAMB16 can find a match in one clock cycle, and can erase and write data in two clock cycles. This VHDL or Verilog module could be used in any HDL design as a building block.

Figure 11 shows the CAM_RAMB16 module built on one SelectRAM+ block and nine RAM32x1s. The multiplexer between the DATA_IN to be written and the DATA_WRITE to be erased is implicit in this implementation. (See Figure 9). Because the first clock cycle is the erase mode, the old data is read from the Init_9_RAM32x1s output DATA_WRITE. It becomes the 9-bit MSB address of ADDRA input. The new data (DATA_IN) is written into the Init_9_RAM32x1s and is reflected on the output DATA_WRITE. The second clock cycle is the write cycle with the new data automatically used as the eight MSB address, see Figure 12. The ADDR[4:0] input is unchanged during the two clock cycles and is used as the five LSB address.

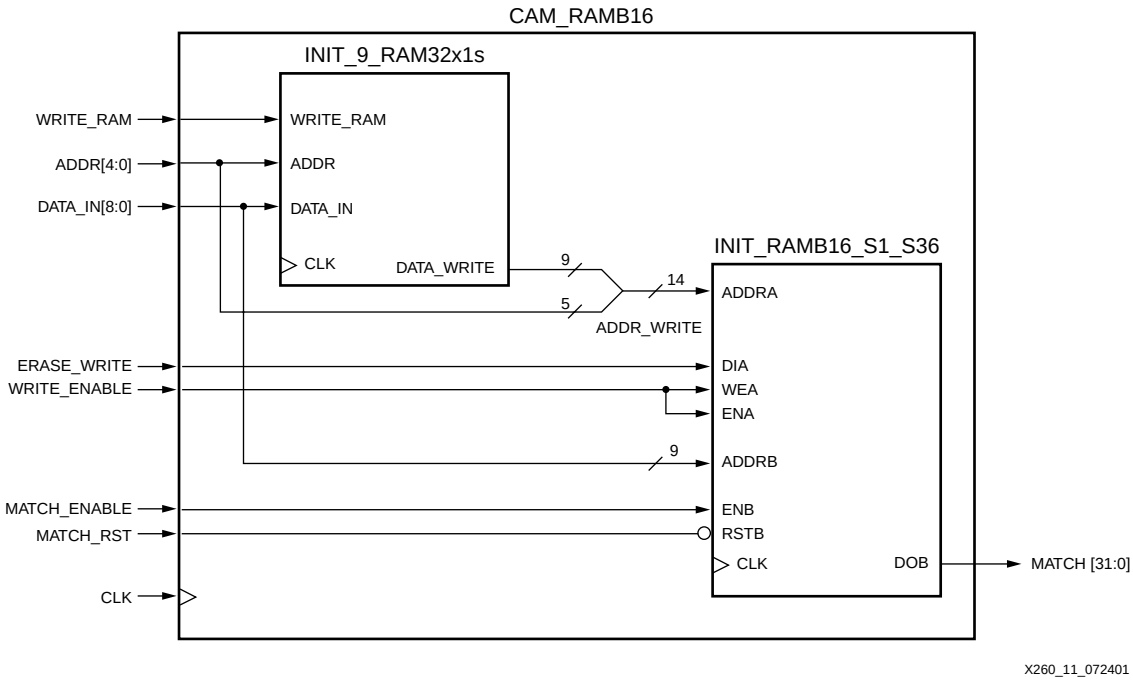

Figure 11: **CAM 32-Word by 9-bit in One Block SelectRAM+ Memory and Nine RAM32x1s**

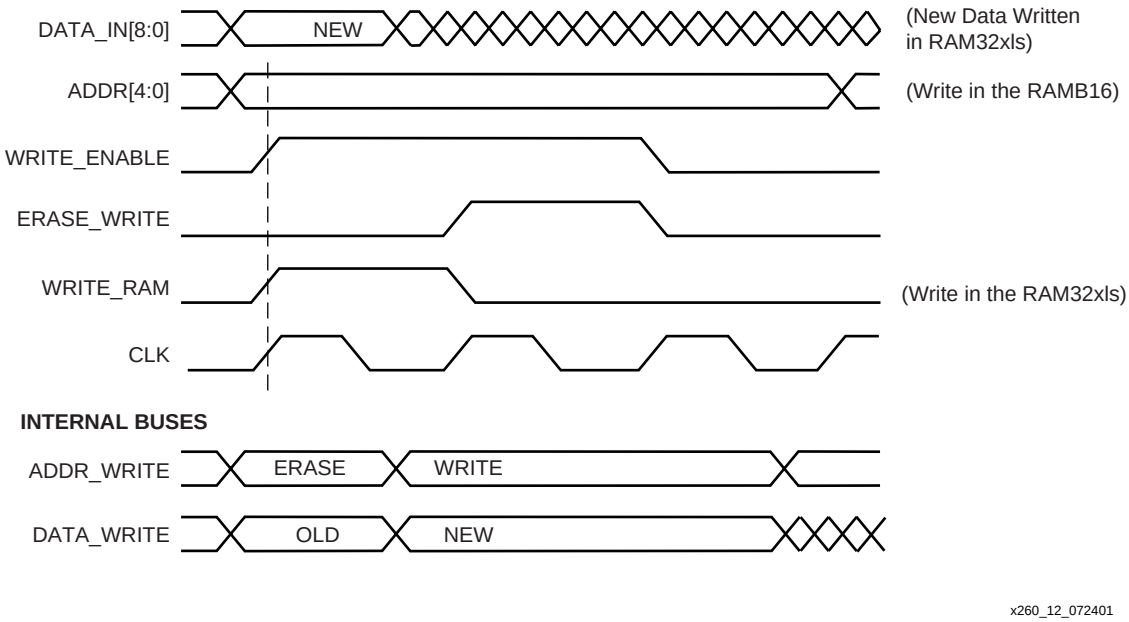

Figure 12: **Erase and Write Waveforms**

Module: CAM_generic_9s

To build the desired CAM size, several CAM_RAMB16 are instantiated in the CAM_generic_9s module, Figure 13. The CAM depth is a generic value “nb_cam32x9s”, defining the number of CAM32x9s to be used. The CAM depth is a multiple of 32. A single CAM_RAMB16 requires a 5-bit encoded address output. The generic value “addr_width” corresponds to CAM depth. A 64-word CAM requires six address lines, a 128-word CAM requires seven address lines, etc. The MATCH buses (CAM_RAMB16 outputs) are encoded to generate both the match address output MATCH_ADDR and a MATCH_OK signal (High when a match is found). This module is similar to the XAPP203 reference design, (CAM_generic_word module). In the XAPP203

reference design, the basic building block is based on Virtex-II slices instead of the SelectRAM+ blocks. The basic module CAM depth is 32 words in both cases, sharing the encoder modules and the top level designs.

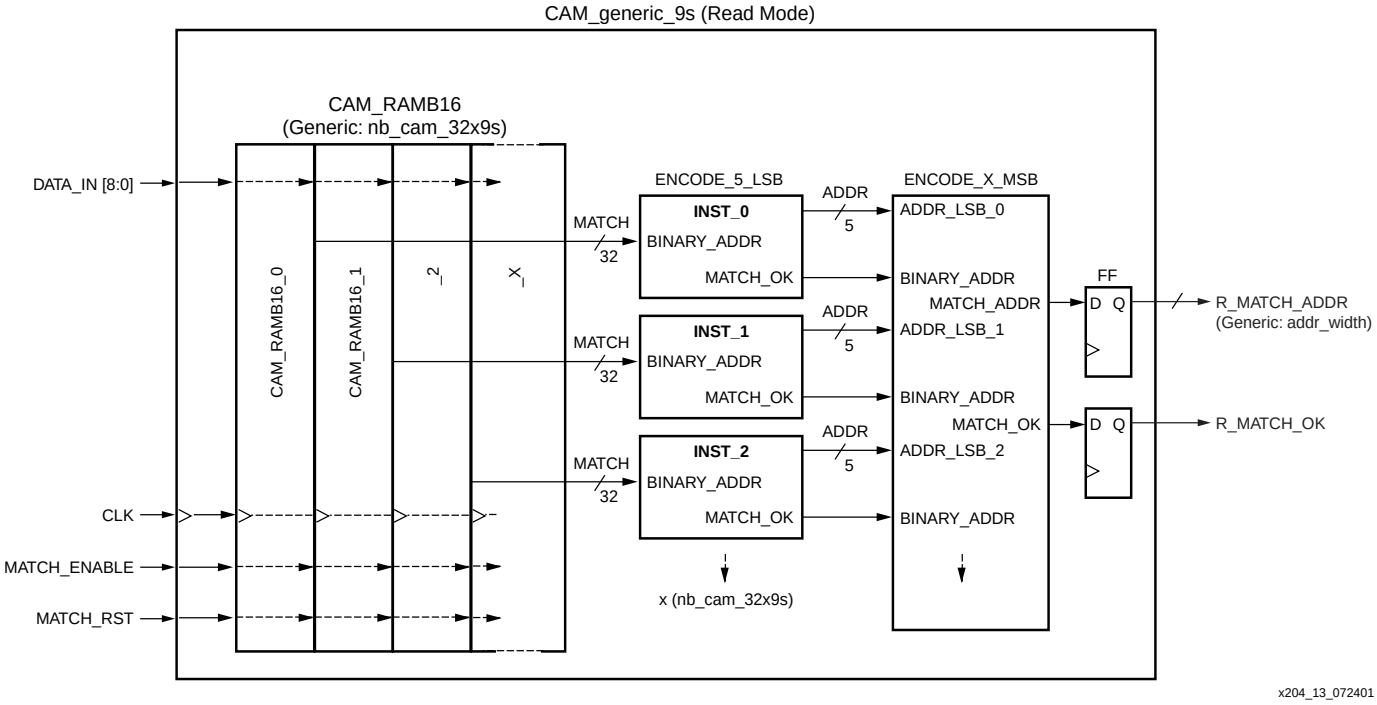

Figure 13: **CAM_generic_9s Read Path**

Module: CAM_top

This module is the top-level wrapper of the generic CAM (Figure 14). It registers all input and output signals.

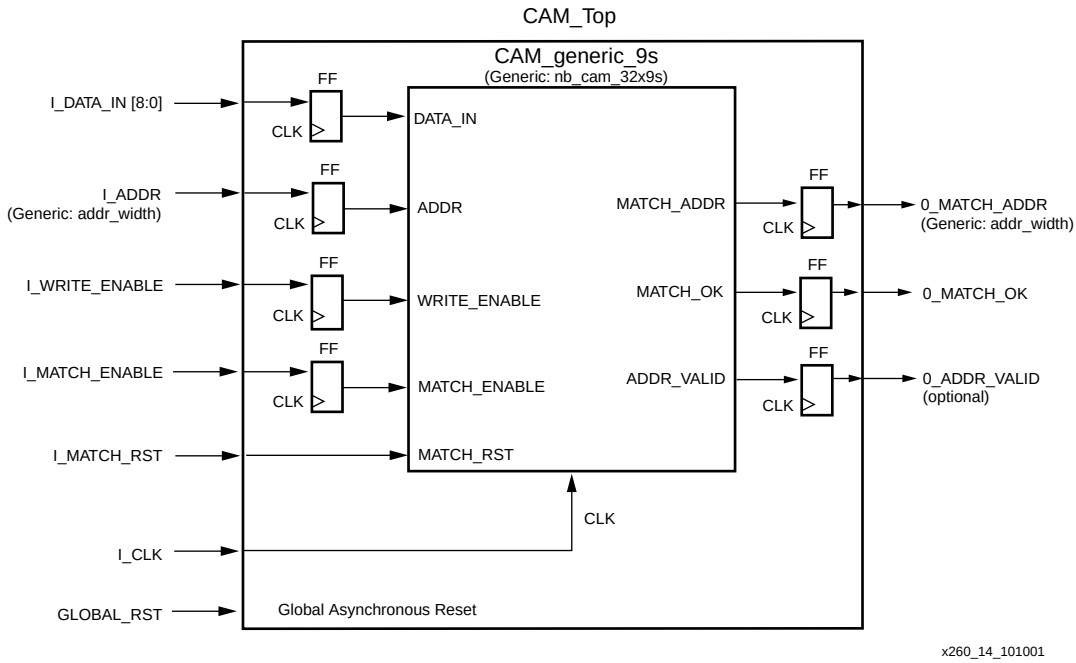

Figure 14: **CAM Design Top Level Block Diagram**

Pinout: (all I_XXX signals are inputs, O_XXX are outputs)

- I_DATA_IN is the 9-bit data input bus used by both read and write operations.
- I_ADDR is the generic input address bus used only to write new data into a selected location.
- I_WRITE_ENABLE is a two clock cycle active High signal (Erase then Write)
- I_MATCH_ENABLE enables a read access (active High)
- I_CLK is the clock. As an option, it can be routed through a DLL.
- O_MATCH_ADDR is the generic output address valid only in read mode.
- O_MATCH_OK is High when a match is found (read operation).
- O_ADDR_VALID is an optional signal when multiple matches occur. Active High indicates O_MATCH_ADDR is valid for a single match.

A recommendation when synthesizing this reference design is to keep the hierarchy for eventual floorplanning and to facilitate static timing analysis. A constraint file (UCF) should define the clock period. For optimal performance, the UCF files can constraint the location of each block SelectRAM+ memory in a column and the eight RAM32x1s primitives in the adjacent Slices column.

This reference design can be easily adapted to different CAM widths. As an example, four CAM32x9 macros produce a 32-word by 36-bit CAM. The 32 additional 4-input AND gates used to generate the global decoded address fit in 32 LUTs.

Reference Design Files

The reference design files can be downloaded from the Xilinx ftp site at:
ftp://ftp.xilinx.com/pub/applications/xapp/

Conclusion

The unique Virtex-II block SelectRAM+ true dual-port features allow an innovative technique to implement embedded fast CAMs. The CAM32x9 macro provides single clock cycle “one-hot” decoded address, even if several macros are cascaded to generate deeper or wider CAM blocks. The small size (32-word by 9-bit) of the basic CAM32x9 macro offers flexible choices in CAM size.

The reference design demonstrates a complete solution including an encoded address output, a match flag, and in addition to fast read access, a fast 2- clock-cycle erase/write access. Expandable CAM depth provides real flexibility.

The CAMs accelerate memory data search. Because of the various requirements among CAM-based applications, solutions must be flexible. In addition to the block SelectRAM+ solution, other approaches based on Virtex-II slices are presented in the application notes XAPP202 “Content Addressable Memory (CAM) in ATM Applications” and XAPP203 “Designing Flexible, Fast CAMs with Virtex slices”.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/30/01	1.0	Initial Xilinx release.
02/27/02	1.1	Updated for Virtex-II Pro devices.