XAPP242 (v1.1) August 23, 2000

Interfacing to Lara Networks Search Engine using Virtex Devices

Author: Stefanka Kitanovska

Summary

Due to rapidly expanding networking industry demands, there is a corresponding need for faster and faster search capabilities within Content Addressable Memory (CAM) devices. Every year new CAM devices emerge on the market. These devices have excellent capabilities and options, but they require an accompanying interface. Virtex™ devices have all the necessary features to interface with high-speed CAMs. This document describes a Virtex CAM controller for the Search Engine (a type of CAM device) from Lara Networks.

Introduction

CAMs have arrays designed to enable stored values to be located quickly by comparing input data to memory data to locate a match. If all matching data is found simultaneously in a search, the CAM is said to be performing at maximum efficiency, and all matches are found in a single clock cycle. A CAM Write mode is comparable to a RAM Write mode, but a CAM Read mode is different. In a RAM, the data at a specific location determined by an address is read. CAMs seek data that matches the input. When a match is found, the output is the address of the data.

In RAMs, address lines limit data storage capacity. In a typical RAM bank, a 10-bit bus addresses 1024 locations of 272-bit data. A CAM does not have this limitation because it does not use an address to read a location. To find a match of 272-bit data in 1024 locations, a 272-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 address lines are not needed to find CAM data the memory size is easily expanded. The width is determined by the storage and comparator size. Figure 1 compares a RAM and a CAM in Read or Search mode. The basic core of a CAM has a storage location and a comparator between the storage location data and the input data. This application note describes a basic CAM design optimized for speed, or density, or both.

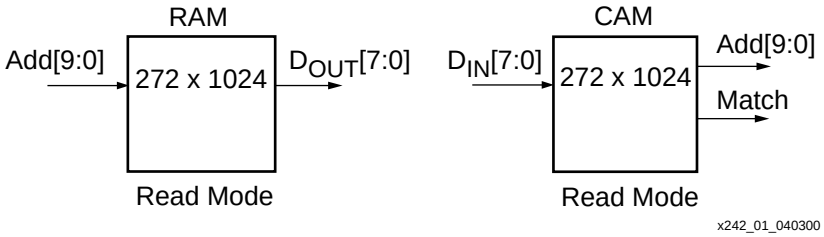

Figure 1: **Simple RAM and CAM Comparison**

Typical CAM Applications

CAMs are used in telecommunications, networking, and many diverse protocol applications. The correct CAM implementation for a particular application requires an analysis of the following parameters.

- Variable Word Size (Width)
- Number of Words (Depth)
- Match or Compare Time (Read)
- Binary or Ternary
- Significance of Write Speed

- Clock Frequency
- Masks and Outputs

CAM Controller Design

This CAM controller is for a Lara Networks LNI7010 Search Engine 1 Mb CAM. The datasheet is located at http://www.laranetworks.com. This document refers to the Lara Networks Search Engine as a CAM device.

The CAM controller offers a simple interface to the Lara Networks Search Engine. The CAM controller interface supports Read, Write, and Search operations in 16K x 68, 8K x 136, or 4K x 272 formats. The Search operation is performed in one clock cycle for the 68-bit and 136-bit data width implementation, and two clock cycles for the 272-bit implementation. A Single Read operation is executed in six clock cycles, and a Single Write operation requires three clock cycles. Table 1 shows the number of required clock cycles for all possible operations.

Table 1: **CAM Operation Modes and Required Clock Cycles**

Operation	Clock Cycles	Condition/Comment
Search	1	68-bit and 136-bit implementations
	2	272-bit implementation
Single Read	6	
Burst Read	4 + 2n	n = burst length
Single Write	3	
Burst Write	n + 2	n = burst length

The design uses three state machines operating at different times. Internal signals in Virtex devices can be in a high-impedance state, a feature used in this CAM controller design. The state machine performing the search operation executes at twice the clock rate of the state machine for the Read and Write state operation. A DLL is used to produce the 2x clock for the search state machine. Figure 2 shows a basic CAM controller design for an interface to one CAM device. The controller is capable of supporting up to 31 cascaded CAM devices. Controller performance and utilization are shown in Table 2.

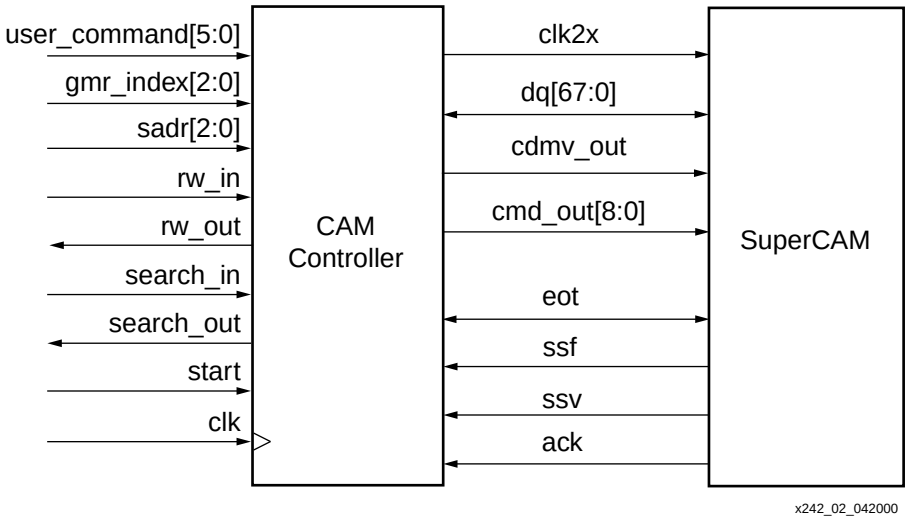

Figure 2: **CAM Controller Interface**

Table 2: **Controller Performance and Utilization**

Speed	Utilization	DLLs	GCLK
112 MHz	412 Slices	2	2

CAM Controller Details

This CAM controller design supports Read, Write, and Search modes in three different widths: 16K x 68, 8K x 136, or 4K x 272. Multiple widths are not supported concurrently. Appendix A details the steps to parameterize the design and to choose a specific width.

The Lara Networks CAM device has five identification pins (signal name id). When performing an operation, the designer is required to supply the ID of the CAM device. On the board, the ID pins on the CAM can be hardwired to the desired value.

Figure 2 details the CAM to Virtex device implementing the CAM controller connections. The signals required by the controller are described in Table 3.

Table 3: **CAM Controller Signal Descriptions**

Signal Name	Number of Bits	Description	
user_command	6	Command register. (Description of bits below.)	
user_command[5]	1	Burst	**Burst Read or Write**. When "1", this bit indicates Burst Read or Write. When "0", this bit indicates Single Read or Single Write.
user_command[4:3]	2	dmr	**Data, Mask, or Register.** dmr = 00: Data array access dmr = 01: MASK array access dmr = 10: External SRAM access dmr = 11: Internal Register access
user_command[2:1]	2	cmd[1:0]	**Command.** Cmd[1:0] is specified by the user and issued to the CAM controller.
user_command[0]	1	cmdv	**Command Valid.** The user must assert cmdv when the command on cmd[1:0] is valid.
start	1	**Start bit.** When set, the CAM initializes. This bit is automatically set to "1" after power up and "0" after initialization.	
cmd_out[8:0]	9	**Command.** Cmd_out[8:0] is generated by the controller and is issued to the CAM.	
cmdv_out	1	**Command Valid.** User must assert cmdv when the command on cmd[8:0] is valid. cmdv_out is issued to the CAM.	
rw_in[67:0]	68	**Read/Write In**. Input data to the Read/Write transmit register.	
address[13:0]	14	**Address**. Specified on rw_in[13:0]. Specifies starting address during Burst Write/Read. Specifies address during Single Write/Read.	
burst_length[8:0]	9	**Burst Length**. Specified on rw_in[22:14] during Burst Read/Write. Invalid during Single Read/Write.	

Table 3: **CAM Controller Signal Descriptions**

Signal Name	Number of Bits	Description
rw_out[67:0]	68	**Read Write Out.** Data received from the CAM when performing a Read.
search_in[67:0]	68	**Data.** Used during the search operation.
search_out[21:0]	22	**Index.** Specifies the index received after a successful search.
ready	1	**Ready.** When "1", indicates that the controller is ready for the next command. When "0", indicates that the controller is currently executing a command.
gmr_index	3	**Global Mask Register Index.** Must be specified during read, write or search operations.
id	5	**ID**. Indicates the ID of the CAM to be used.
sadr	3	**SRAM address.** Used for external SRAM access.
id_enable	1	**ID enable**. When "1", the next ID can be applied. Used during initialization.
match	1	**Match flag**. When "1", indicates a successful search.
match_valid	1	**Match Valid**. When "1", validates match.

Figure 3 is a block level diagram of the CAM controller design. There are four registers in this design: two transmit, and two receive. Two registers (one transmit, one receive) operate on the 1x clock (clk) and are used during reading and writing to the CAM device. The other two registers (one transmit, one receive) operate on the 2x clock (clk2x) and are used during the search operation. The master state machine is separated into three sub-state machines. The three sub-state machines are called Read/Write ("rw" module), Search ("search" module), and Initialize ("init" module). These state machines are mutually exclusive, and hence do not operate simultaneously. This is necessary to avoid output bus contention, as they all share common outputs, but only one state machine drives the outputs at a time.

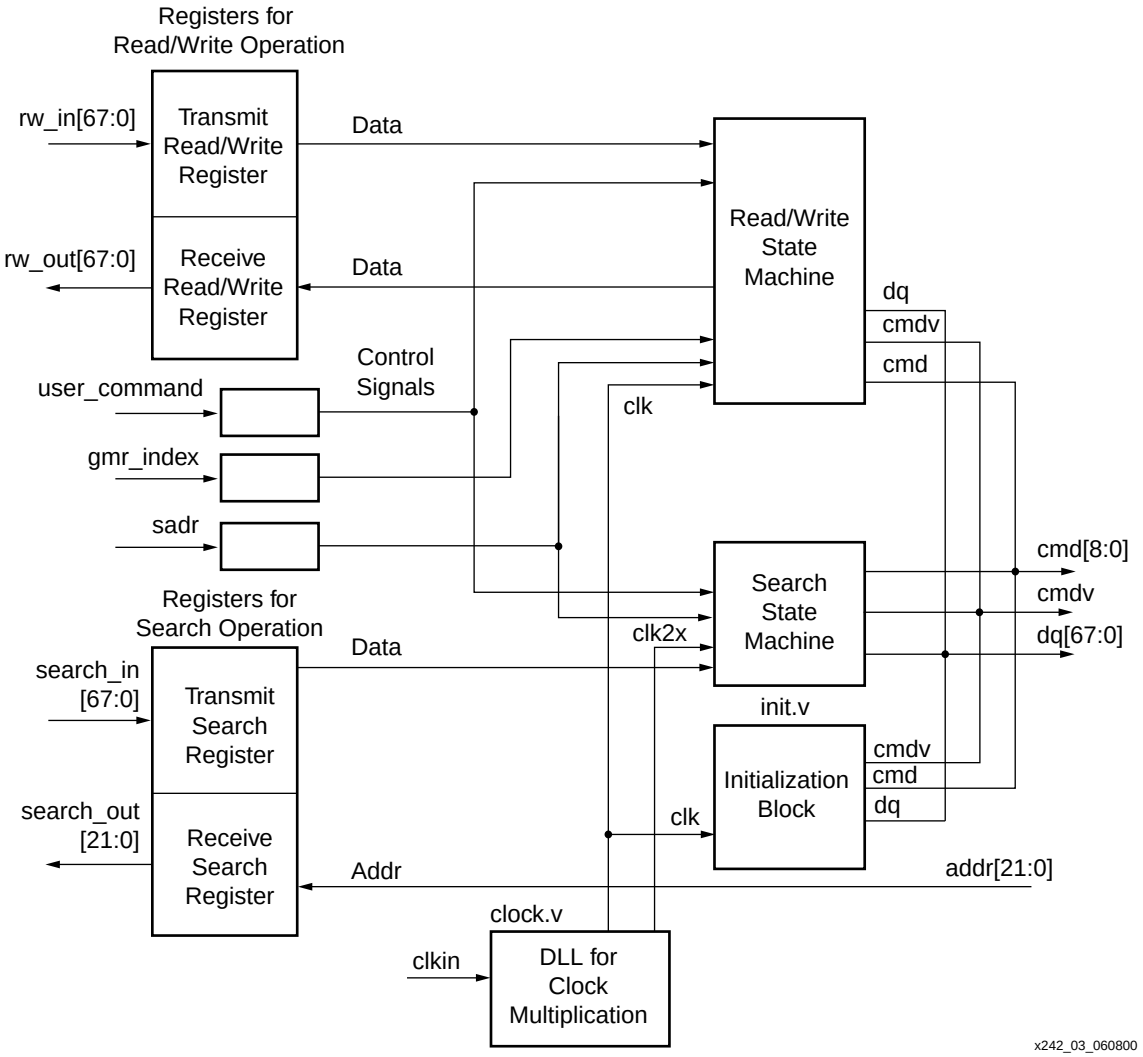

Figure 3: **CAM Controller Block Diagram**

The state machines are executed in the following order. The Initialize state machine becomes operational immediately after start-up with the start bit controlling the operation. Before initialization of the CAM, the Virtex device must be configured, and the DLL locked. At this time, the start bit should be held at "0". To start initialization, the start bit must be set to "1" and held for two to five clock cycles, and afterwards set to "0". Therefore, the Initialize state machine does not run again without another power up or user reset of the Virtex device. The Initialize state machine initializes all the CAMs in the system. To initialize a CAM, the ID of that CAM must be supplied. The id_enable signal will indicated when the to supply the next CAM ID. Figure 4 shows the Initialize state machine.

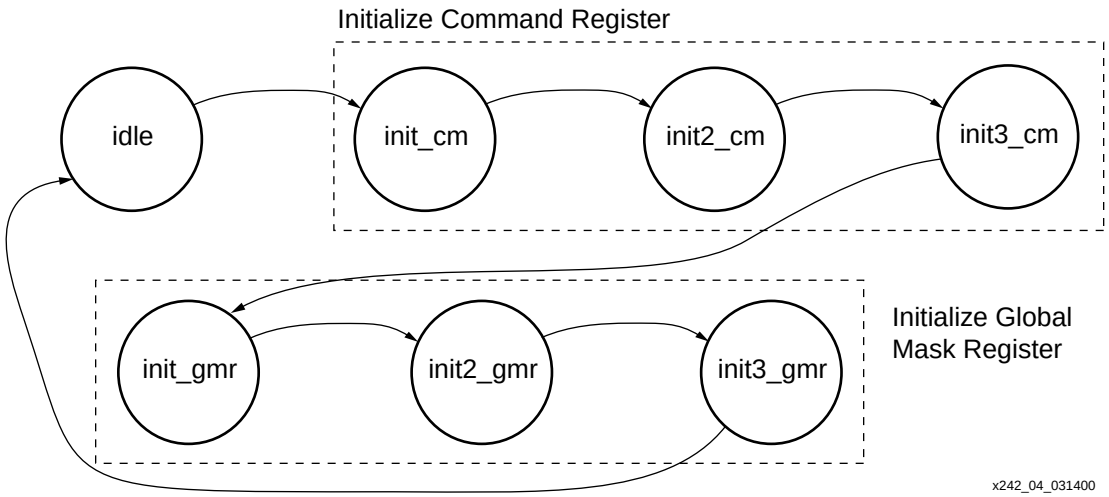

Figure 4: **Initialize State Machine**

After the Initialize state machine completes execution, the controller waits for a command to perform either a write to or a read from the CAM. The following steps are necessary immediately after initialization.

1. Program the Information register
2. Setting the CAM data and mask arrays to "1"

Programming the Information Register

The Information register is programmed by a Write to that register. Table 4 shows the user_command format for a register write. In the first clock cycle, rw_in must have the value shown in Table 5. In the second clock cycle, rw_in must specify the data to be written into the Information register. Table 6 shows the programmable fields of the Information Register.

Table 4: **Internal Registers Write user_command Format**

burst	dmr	cmd[1:0]	cmdv
0	11	01	1

Table 5: **rw_in Bus for Register Write**

Unused	Address
[67:6]	[5.0]

Table 6: **Information Register Description from the Lara Networks LNI7010 Data Sheet**

Field	Range	Initial Value	Description
RVSN	[3:0]	0001	**Revision Number**. This is the current device revision number. Numbers start from one and increment by one for each revision of the device.
IMPL	[6:4]	000	This is the CAM device implementation number.
	[7]	0	Reserved.

Table 6: **Information Register Description from the Lara Networks LNI7010 Data Sheet**

Field	Range	Initial Value	Description
Device ID	[15:8]	00000001	This the is device identification number
MFID	[31:16]	1101 1100 0111 1111	**Manufacturer ID**. This field is the same as the manufacturer ID and continuation bits.
	[67:32]		Reserved.

Table 7 specifies the addresses for the internal registers of the Lara Networks Search Engine. The value given is in decimal.

Table 7: **LNI7010 Register Overview**

Address	Abbreviation	Type	Name
0-31	COMP0-31	R	32 Comparand Registers. Stores comparands from the DQ bus for later learning.
32-47	MASKS	RW	16 Global Mask Register Arrays
48-55	SSR0-7	R	8 Search Successful Index Registers
56	COMMAND	RW	Command Register
57	INFO	R	Information Register
58	RBURREG	RW	Burst Read Register
59	WBURREG	RW	Burst Write Register
60	NFA	R	Next Free Address Register
61-63	-	-	Reserved

Setting the Data and Mask Array of the CAM

The CAM data and mask arrays should be set to "1" immediately after initialization. The easiest way to set all data is to perform a Burst Write. Refer to the Write section in How to Use the CAM Controller for additional details.

How to Use the CAM Controller

The CAM controller design supports multiple functions including Write and Burst Write to a data or mask array, Read and Burst Read from a data or mask array, and Search of a data or mask array. It performs Read or Write to internal registers as well as Single Read and Single Write to an external SRAM device. The necessary steps for these functions are described in this section.

Write

A Write can be either a Burst Write or a Single Write to the CAM data array, mask array, or an internal register. To specify a Burst Write, set the burst bit to "1". When this bit is set to "0", only a Single Write is executed.

The controller command format for a Single Write to the data array is shown in Table 8. For specific dmr values, see Table 3.

Table 8: **Single Write user_command Register Format**

burst	dmr	cmd[1:0]	cmdv
0	00	01	1

The data array or mask array location address for the data Write is on the rw_in[13:0] bus is show in Table 9.

Table 9: **Single Write rw_in Bus Status**

unused	address
[67:14]	[13:0]

Figure 5 shows the Read/Write state machine.

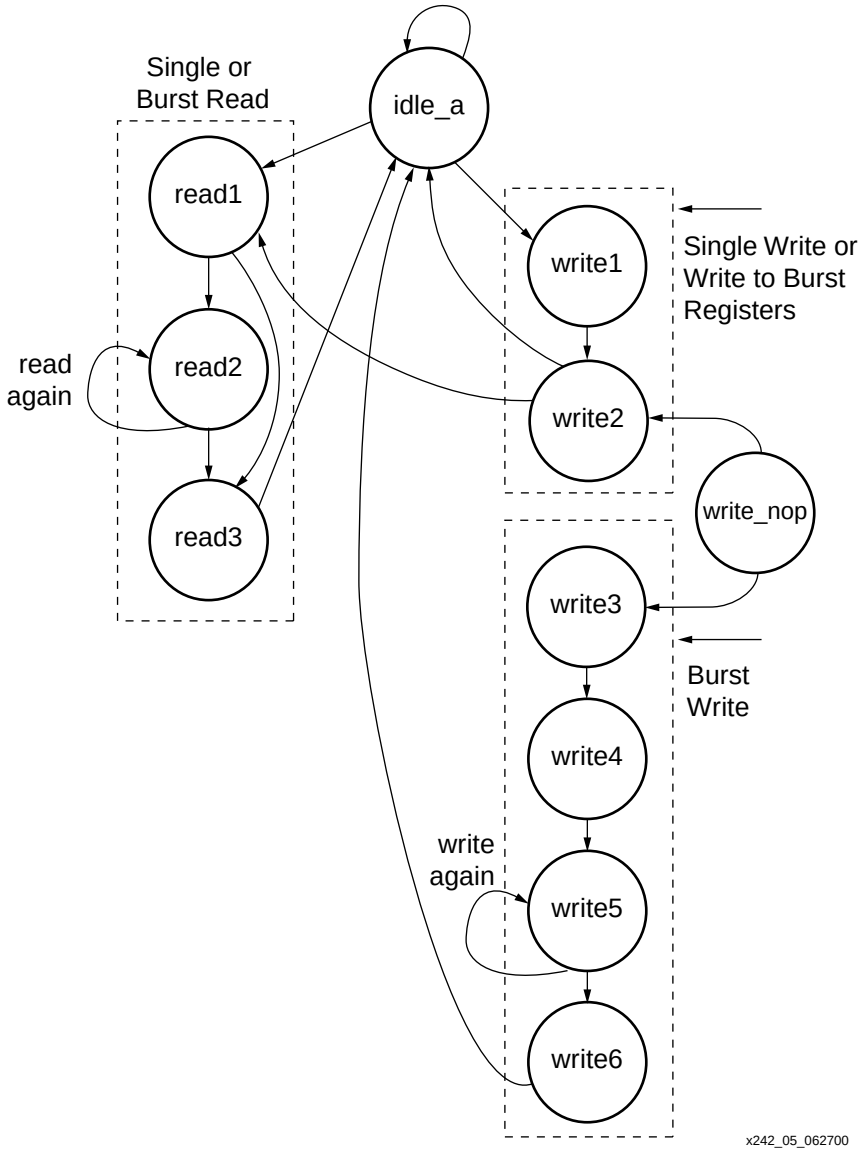

Figure 5: **Read or Write State Machine**

When cmdv is asserted, the controller receives the command supplied on cmd[1:0] and starts executing. The data or address necessary for execution of the specified command must be ready at the outputs of the transmit Read/Write register at this time. When cmdv is valid and cmd[1:0] = 01, the Read/Write state machine starts executing from the idle_a state shown in Figure 5. For a Single Write, it fetches the address from the transmit Read/Write register in the first clock cycle. In the second clock cycle, it fetches the data from the transmit Read/Write register.

To perform a Burst Write, the controller must first program the Burst Write register inside the CAM. Table 10 shows the format of the user_command register for performing a Burst Write to the data array. To Write to the mask array, set dmr to a "01".

Table 10: **Burst Write Command Register Format**

burst	dmr	cmd[1:0]	cmdv
1	00	01	1

The designer is required to specify the burst length and starting burst address. These values are sent to the CAM. The values must be stored together in the transmit Read/Write register in the format shown in Table 11. Bits [13:0] specify the starting address for the Burst Write, and bits [22:14] specify the burst length to be written into the Burst Write register. This must be done prior to execution of the Burst Write command.

Table 11: **rw_in Format for Specifying Burst Length and Starting Burst Address**

unused	burst_length	addr
[67:23]	[22:14]	[13:0]

In the first clock cycle, the user_command in Table 10 is supplied. In the second clock cycle, the burst length and starting burst address must be sent to the Read/Write register (rw_in). In subsequent clock cycles, the data to be written to the CAM must be supplied on the Read/Write register (rw_in), one data bit per clock cycle until the Burst Write command is executed. By asserting the ready flag, the controller indicates that it is ready for the next command.

Read

As with the Write command, the Read command can also be a Single Read, or a Burst Read. The Burst bit in the user_command register specifies whether the Read is Burst Read or a Single Read. Figure 5 shows the state diagram of the state machine used during the Read operation. Table 12 shows the format of the command register for performing a Single Read from the data array. Table 3 shows all of the dmr values.

Table 12: **Single Read Command Register Format**

burst	dmr	cmd[1:0]	cmdv
0	00	00	1

In the same clock cycle that cmdv is asserted, the controller fetches the Read address from the transmit Read/Write register (rw_in), so this address must be ready at the outputs of the transmit Read/Write register. Table 13 shows the format of the rw_in signal for Single Read. With a 2-cycle latency, the data is sent back from the CAM. This data is then written into the receive Read/Write register (rw_out).

Table 13: **Single Read rw_in Bus Status**

unused	address
[67:14]	[13:0]

When performing a Burst Read, the controller must first program the Burst Read register inside the CAM. The format of the user_command register for a Burst Read to the data array is shown in Table 14. To Read from the mask array, set dmr to "01".

Table 14: **Burst Read Command Register Format**

Burst	dmr	cmd[1:0]	cmdv
1	00	00	1

To program the Burst Read register, the controller performs a Write to the Burst Read register automatically. The designer must supply the burst length and the starting burst address. The format for the data entered in the transmit Read/Write register (rw_in) is shown in Table 15. One clock cycle after cmdv is asserted by the user, the Read/Write state machine fetches the burst address and burst length from the transmit Read/Write register (rw_in). Two clock cycles later, the data is sent from the CAM. The data is then stored in the Read/Write receive register (rw_out).

Table 15: **Burst Read rw_in Bus Status**

unused	burst_length	addr
[67:23]	[22:14]	[13:0]

Search

The Search command operates on clk2x, and therefore uses two different transmit and receive registers, denoted as search registers in Figure 2. The search operation is a 2- or 4-cycle operation depending on the width selected. However, since it operates on clk2x, the search operation appears to be a single clk clock cycle operation (for 68-bit and 136-bit wide data). If the data width is 272 bits, the search command uses two clk clock cycles to execute. Figure 6 shows the state diagrams of the search state machine. Only one state machine is implemented, based on the width selected.

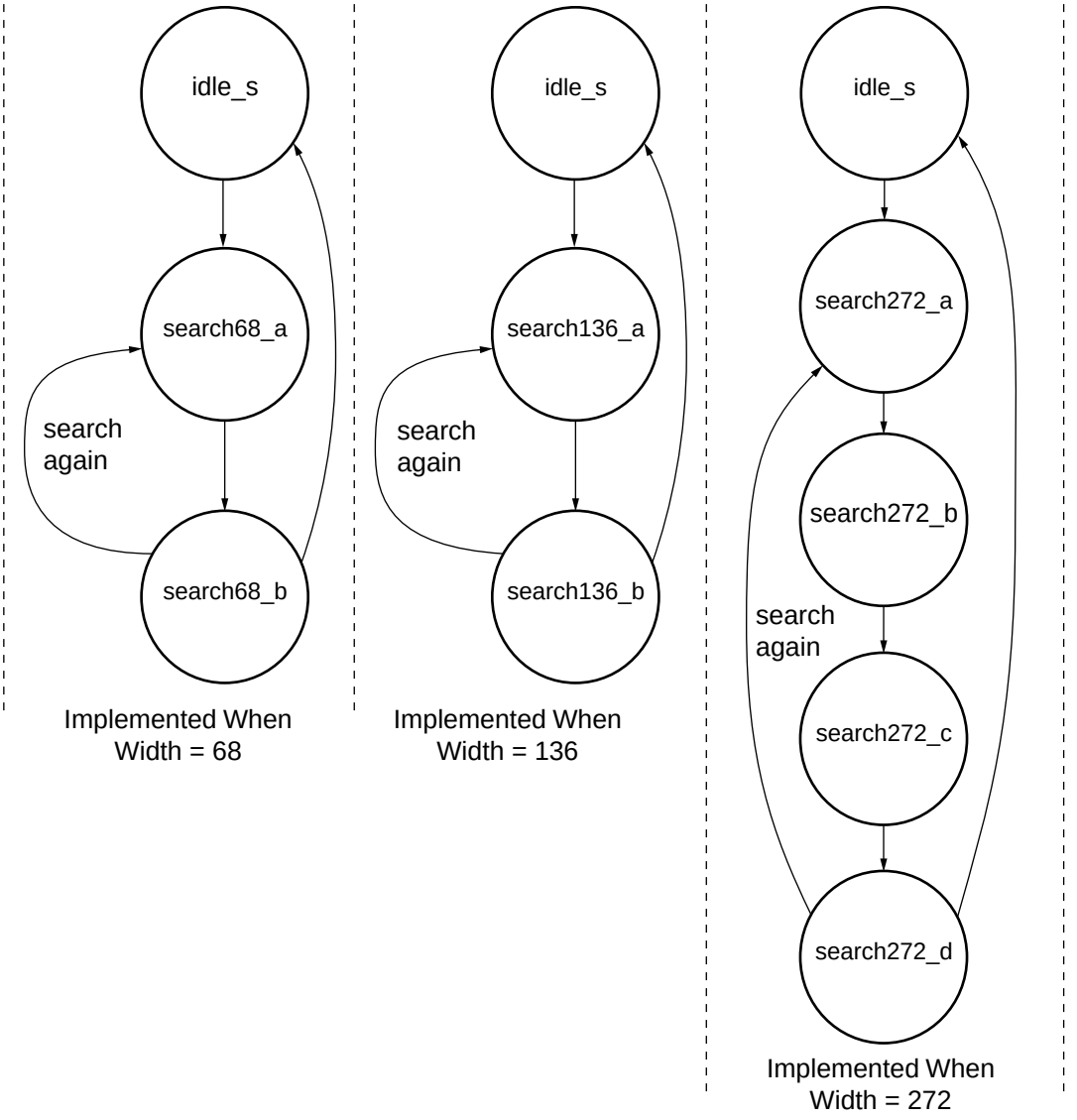

Figure 6: **State Diagrams for the Search State Machines**

The CAM data returns with some latency depending on the number of CAMs. Latency varies with the number of CAMs, as shown in Table 16.

Table 16: **Data Latency**

# of Devices	Latency in Clock Cycles
1	4
2-8	5
9-31	6

Table 17 shows the command register format for a search operation.

Table 17: **Search Command Register Format**

Burst	dmr	cmd[1:0]	cmdv
0	00	10	1

In the same cycle that cmdv is asserted, the data to be sent to the CAM is being read from the transmit search register. There are three possible data widths, and therefore three possible search implementations. The width parameter is defined in the defines.v file. See Appendix A for more information. The transmit search register is 68-bits wide. In a 68-bit width implementation, the data on the transmit search register must be held for two clk2x cycles. In a 136-bit implementation, the upper 68-bit portion of the 136-bit data must be supplied in the first clk2x clock cycle, then the lower 68-bit portion is supplied in the second clk2x clock cycle. For the 272-bit implementation, 272-bit data is split into four 68-bit portions, and sent in four different clk2x cycles starting with the uppermost portion first. The data must be sent to the search transmit register using the clk2x clock.

If a match is found, the match flag is asserted one clock cycle later. The address is sent back on sadr[13:0].

Conclusion

The flexible Virtex families are high performance partners for CAM controllers applications. Lara Networks Search Engine devices are some of the most efficient CAM devices available. Maximum performance from the Search Engine is achieved when the CAM controller is implemented in Virtex devices.

Appendix A

Verilog Code

The reference design files are available for download on the Xilinx web site in PC (**xapp242.zip**) and UNIX (**xapp242.tar.gz**) formats.

The CAM controller design can be parameterized to allow different width and depth configurations. This is done by editing the defines.v file. To specify the desired width, remove the comment designator. The following example has a desired width of 272.

```
//Please comment out the define commands that you are not using
//`define width68   2'b00     //for 68-bit data width
//`define width136  2'b01     //for 136-bit data width
`define width272    2'b10     //for 272-bit data width
```

For the desired depth the variable depth must specify the number of cascaded CAMs in binary. Remove the comment designator on the define command line specifying the required depth of the configuration. The following example is for depth of two CAMs. The variable depth is defined with a value of 2 in binary. The appropriate depth2_8 line is not commented.

```
//Please comment out the define commands that you are not using
`define depth       5'b00010  //User defined cascading depth. Specify
                              //depth in binary.
//`define depth1    2'b00     //Un-comment if depth is 1 device
`define depth2_8    2'b01     //Un-comment if depth is between
                              //2 and 8 devices
//`define depth9_31 2'b10     //Un-comment if depth is between
                              //9 and 31 devices
```

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
6/27/00	1.0	Initial Xilinx release.
8/23/00	1.1	Replaced Lara Networks part number to LNI7010