CSE434 Project Proposal

James Dicke, Ben Chavet

November 2, 2007

1 Summary

We propose to design a ternary Content Addressable Memory unit (CAM). Our CAM stores 16-bit words, and accepts an 8-bit key and an 8-bit mask. By applying the mask, we are able to handle "don't care" values in the key. The CAM can be queried by a key/mask pair, and the data word corresponding to the first matching key is returned.

Similarly, when writing to the CAM, the first storage location with a key that matches the key/mask input value is written. If there are no keys in the CAM that match the key/mask input value, the first vacant storage location is used. Finally, if there are no vacant locations, the *memfull* signal is asserted.

By using a modular design, we are able to chain multiple CAM units together to provide more storage space.

2 Background

Standard memory uses a static address space to access its contents. So, in order to access a value, you must know its address. In contrast, a CAM uses a key/value pair, much like how an associative array works in software. In order to access some data, you request it by using its key. In addition, a ternary CAM allows the use of "don't care" values when specifying the key. This adds flexibility in the search functionality of a CAM. [1]

3 CAM Functions

The basic functions of our CAM are: read, write, delete, reset, and pass. Except for reset and pass, all of the operations rely on searching the CAM for keys that match the key/mask input pair. This input generates a search key such that the bits of the input key that correspond to the bits of the mask that are set to "1" are used literally. The bits that correspond to the bits of the mask that are set to "0" are considered to be "don't cares."

For example, if the contents of the CAM are as follows:

Key	Value
01100100	0x9B
10010100	0xA4
10110000	0x3F
11101000	0xFFFF
10100000	0x5AB
11001000	0x6A

An input address specified as 1x1x0x00 matches both 10110000 and 10100000, but because 10110000 is the first listed, the CAM only returns the data associated with that key, namely 0x3F. In addition, when a match is found, the *foundout* output is asserted. When no match is found, *foundout* is not asserted. This is important for the Read operation, where it is necessary to be able to determine the difference between when a match is not found, and when a match is simply zero.

3.1 Read

Read uses the match function described above to send the value corresponding to the first matched key to the CAM's output. If no match is found, the *foundout* output is set to zero, and all of the output bits are set to zero.

3.2 Write

Write is easily the most complicated function of the CAM. It first checks for a match from the key/mask inputs. If one (or more) is found, the value corresponding with that match is overwritten with the input data. However, if no match is found, the input data word is stored at an unused storage space. If there are no unused spaces, the value is not written, and the *memfull* output is asserted.

Due to the complexity of the write operation, it is also slow. This is minimized by making the search function as fast as possible.

3.3 Delete

Delete works in the same manner as Write. It deletes the key and value corresponding to the match. If no match is found, nothing is done.

3.4 Reset

Reset is as simple as it sounds. When the *reset* input is asserted, the entire contents of the CAM are deleted, resulting in an empty CAM.

3.5 Pass

Pass is used for chaining (see below). When the *pass* input is asserted, the CAM simply passes the data input values directly to its data output. Other than for chaining, this doesn't really have much utility.

4 Inputs and Outputs

Inputs

- **8-bit key**
- **8-bit mask:** Specifies which bits of the key are "don't cares".
- **16-bit input word:** The word to be written
- **memwrite:** When asserted, the input key and input word are written to memory. When not asserted, a read operation is performed based on the input key and mask.
- **delete:** When asserted, the first entry with a key that matches the input key/mask pair is cleared to make room for new input.
- **reset:** When asserted, the entire contents of the CAM are cleared.
- **foundin:** Used for chaining. When asserted, signifies that a match has already been found in a previous CAM unit.

Outputs

- **16-bit output word**
- **memfull:** Asserted when all memory locations are in use.
- **foundout:** Asserted when an input address has been matched and its corresponding data is being output. This is important to differentiate between when a match is not found, and when an output value is 0x00.

5 Power Consumption

A custom method of reducing the design's power consumption is used. Once a word is detected, a signal is passed through the unit disabling the outputs from the remaining memory locations, thus saving power. Assuming an even distribution of keys that are read, the power consumption should be half that of a CAM with no power saving circuitry.

Additional power consumption issues will be considered as time permits. Several options exist for power conservation. The most popular methods to save power are bank selection and pre-computation. Bank selection divides the

readable memory into banks. When storing and retrieving data from the CAM, a number of input address bits are used to select which bank to use. This structural option greatly reduces the amount of active memory units and corresponding search circuitry, thereby greatly decreasing its power consumption. Pre-computation performs an operation on existing memory addresses so that only memory addresses which first pass the pre-computation are searched. One example of pre-computation is counting the number of 1's in an address. When an address is input, the number of 1's are counted and the input address is only compared to existing addresses that match the correct number of 1's. These comparisons require more hardware, but use less power due to the decreased amount of switching.

6 Chaining

Another feature of our CAM is the ability to chain multiple CAMs in series, allowing them to act as one continuous CAM unit. We specifically provide all of the inputs and outputs necessary to make this possible.

Each CAM unit shares the same key, mask, input word, *memwrite*, *delete*, and *reset* lines. Each CAM's *foundin* is connected to the previous CAM's *foundout*, connecting the first CAM's *foundin* to ground. *Foundout* on the last CAM indicates whether there was a match in any of the CAMs in the chain.

When a CAM matches a key, it asserts its own *foundout* in order to disable any further switching in all subsequent units. Additionally, the output corresponding to the match is then passed to the previous CAM in the chain, where it is then propagated all the way to the first CAM as the final output. This is accomplished by connecting the *foundout* output from one CAM to the *pass* input of the previous CAM.

References

[1] http://en.wikipedia.org/wiki/Content-addressable_memory