Lab 6 - Generic Flip-Flop CAM Design Space Exploration

Intro:

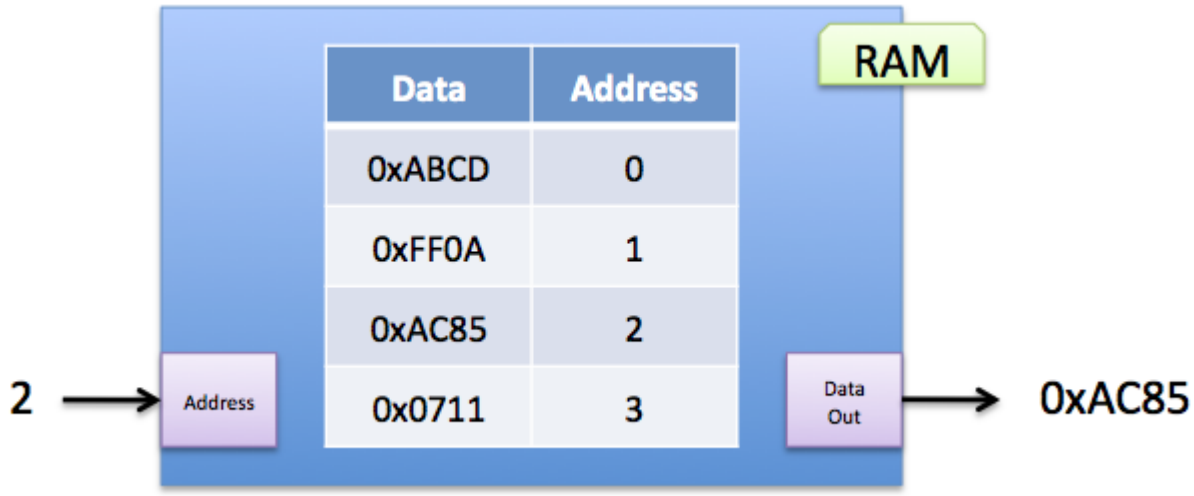
In this lab you will be implementing a generic Content Addressable Memory (CAM). This CAM will have the following parameters:

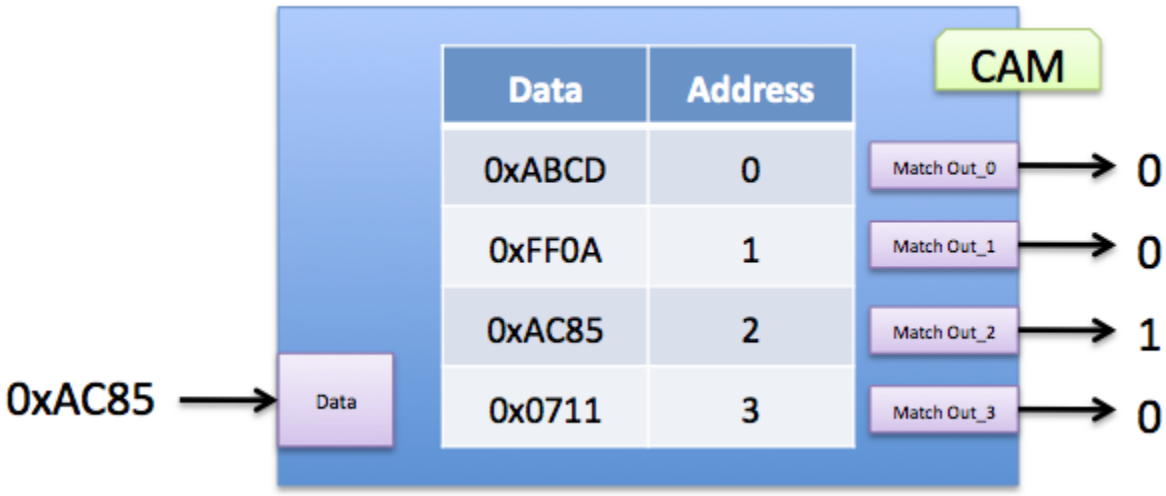
- M, being the number of words stored in the CAM
- N, the width of each word in the CAM. (you can assume all words in the CAM will have the same width)

After you have implemented the CAMs you will do some design space exploration. To do this you will fix one parameter of the CAM, and vary the other. Doing this you will recored the resources used on the FPGA, and its operating frequency.

RAMs vs CAMs

Random Access Memories (RAMs) will read in one address. Find the data stored at that single point in its memory, and return the data it finds. A CAM will do something entirely different. A CAM will take a data point search its **ENTIRE**memory for that data point, and return all address locations that match the data point. As can be seen in the following figures:





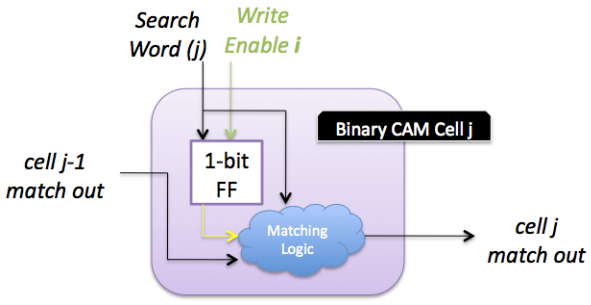
CAM design

The CAMs you will be building for this lab can be thought of as having three levels. The top level (what the user will ultimately use) is the CAM Array level. The CAM Array level consists of M CAM rows. Each CAM row will then consist of N CAM cells.

CAM Cell

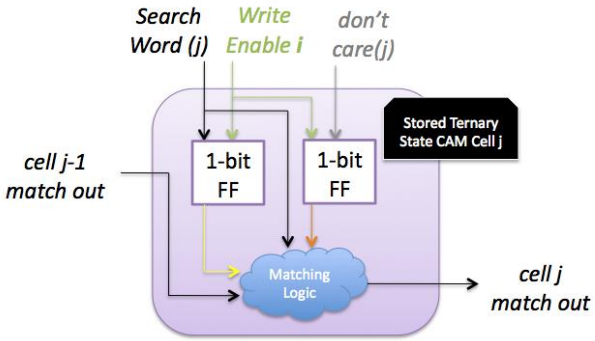
The CAM cell will hold one bit of data in a Generic Flip Flop. This bit is continually being read (on the rising edge of the clock), unless the bit is being written. There are three different CAM cells you should implement for this lab.

The first CAM cell, **Binary CAM Cell**, is the simplest. This CAM cell will store only one bit of data. This bit of data can be changed by setting the 'search_word' port to a value, and setting write_enable high. The jth binary CAM cell in a CAM row has an input from the (j-1)th cell. This input says whether the last j-1 cells have a match up to this point. The jth cell determines if it has a match, and propagates its result to the (j+1)th cell.

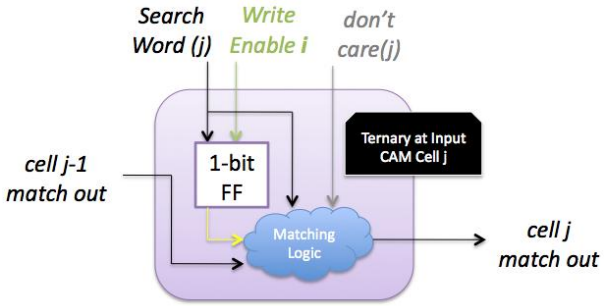


The next CAM cell, **Stored Ternary State CAM Cell** works similar to the binary CAM cell. The main difference being a don't care bit. The stored ternary state CAM cell will keep a second flip-flop that stores this don't care bit. When this bit is high, it doesn't matter what value is stored

in the CAM register. The don't care bit is written in the same fashion as the CAM register.

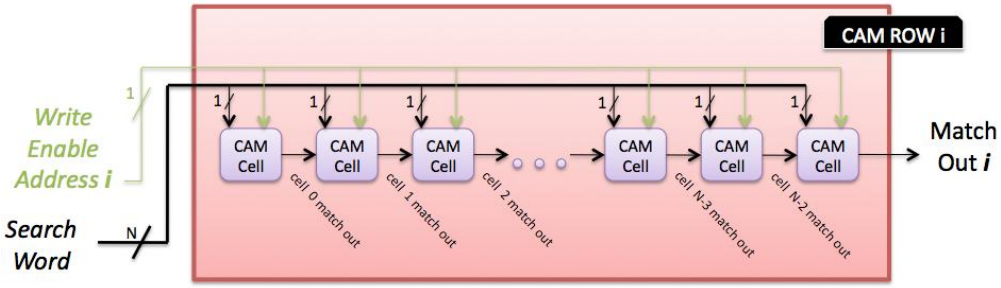


The last CAM cell, **Ternary at Input CAM Cell** will not store a don't care bit. Instead each search into the CAM is accompanied by a don't care bit. The match logic will consider the don't care bit on a search, by search basis.



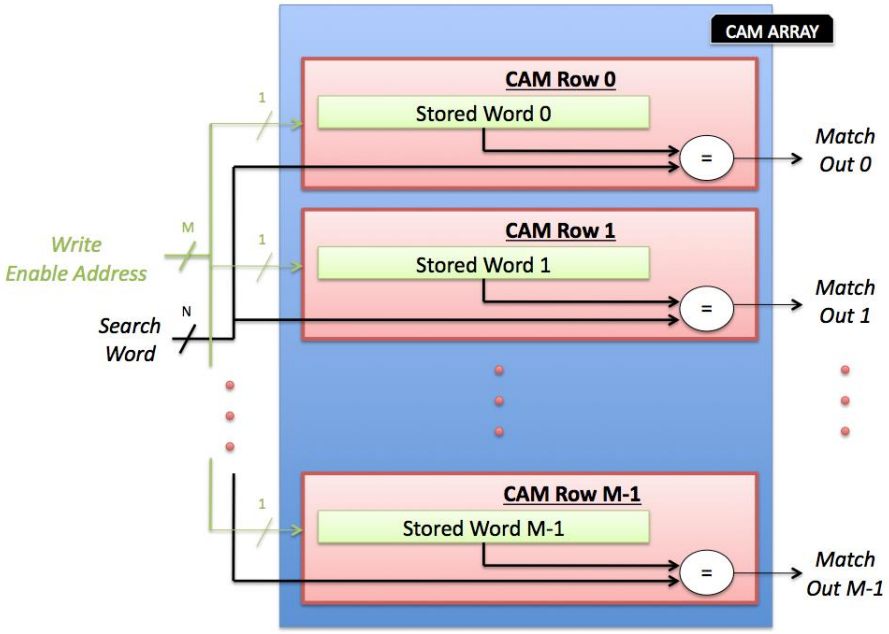
CAM Row

The CAM row should consist of N CAM cells. The CAM row will take a search word, N bits in length, and compare each of these N bits to their corresponding CAM cells. If all the cells match then this CAMt row sets match_out to high. The CAM row also has a 1-bit signal for write_enable when write_enable is set to high the CAM row will write the N-bits of search_word to the N CAM cells.



CAM Array

The CAM array will hold M CAM rows. The CAM array will read in one search_word, N bits in length, and send this N-bit search_word to all M CAM rows. The CAM array will also have an M-bit match_out port (one for each CAM row), to show which rows matched the search. The CAM array will have a M-bit write_enable_address that will be used to specify what CAM rows will be written to.



Lab Details:

For this lab you are given skeleton code. These 4 files are all you need to complete this lab.

- CAM Cells - Implement the 3 CAM cell types as described above. All should have the same input and output ports (Even if you don't use all of them in your design).
  - BCAM Cell
  - TCAM Cell
  - STCAM Cell
- CAM Row - You should create N CAM cell, based on the CAM_WIDTH generic. You should also connect all N CAM cells together in this file. You should be able to change the CAM cells being used in your design from this file.
- CAM Array - You should create M CAM rows, based off the CAM_DEPTH generic. You should also connect all M CAM rows to the necessary CAM Array ports.
- CAM Wrapper - This file you do not have to modify, but all lower files should work with this file as the top level.

Some hints and specifications for this lab.

- When searching for a word it should take two cycles to return the correct rows that match.
- Not all parts of the CAM cell need to run based on the clock
- All flip flops will run based on the clock

Turn-in:

Each group should turn in one tar file to iLearn The contents of which should be:

- A README file with the group members names, and any incomplete or incorrect functionality
- All VHDL files for your design