Advanced features proposal and designs

4-way set associative L1 cache(may be 8-way set associative cache):

We plan to expand the current L1 cache into a 4-way 8-set associative cache with pseudo-LRU replacement policy. We will implement a 3-bit pseudo-LRU replacement policy. This policy keeps track of the usage history of each cache line in a set by maintaining a binary tree that has a height of 2. Each node in the tree corresponds to a cache line in the set. The root node represents the entire set. Each internal node represents a subset of the cache lines. The left child represents the subset of lines that were accessed less recently, while the right child represents the subset of lines that were accessed more recently. The leaf nodes are labeled 0 to 3 to indicate the cache line number within the set.

Victim Cache:

We plan to create a fully associative victim cache that consists of 4 cache lines. It is placed between a 4-way set associative L1 cache and the L2 cache. Whenever there is a L1 cache miss, the victim cache is checked first. If the address hits in the victim cache, the data is returned to the CPU and also promoted to the L1 cache by replacing its conflicting competitor. The data that was evicted from the L1 cache is then transferred to the victim cache. If there is a miss in the victim cache, the L2 cache is accessed, and the arriving data fills the line in the L1 cache while moving the current data to the victim cache. In this scenario, the replaced entry in the victim cache is discarded and, if dirty, written back to the L2 cache.

Parameterized L2 4-way cache system:

We plan to add a set-parameterized L2 cache design that allows for flexibility in its configuration by allowing us to specify the number of sets for the L2 cache. We want to have a flexible L2 cache unit. We'll not parameterize the number of ways for our L2-cache system, which means we will not parameterize our 3-bit pseudo-LRU logic for the 4-way cache system. When it needs a large cache unit, we can modify the parameter to achieve such an effect. We will define the parameters and pass the parameters into the modules for tag, data, valid and dirty arrays in our L2 cache datapath. By changing the size of these array modules, we could parameterize our L2 cache. The L2 cache system that will be connected between the L1 victim cache and the cache line adaptor.

Local Branch History Table:

We plan to create a separate module for dynamic branch prediction. In this module, we will create a state machine which has 4 states: SNT, WNT, WT, ST which correspond to strongly not taken, weakly not taken, weakly taken, strongly taken. This module will be initialized to WNT at the reset time. Once it receives a branch enable signal, it will update the state based on the value of branch enable signal. If the branch enable signal is 1, the state will transit in the following order: SNT -> WNT -> WT ->ST. If the branch enable signal is 0, the state will transit in the opposite direction. The outcome of this module will be sent to datapath to indicate whether the branch is likely to be taken or not.

Basic Hardware Prefetching:

We plan to create a separate module for prefetching. Once the cache system sends a cache miss signal to the prefetching module, the prefetching module looks into the cache system to see if the instruction at PC+4 is in the cache or not. If not, the prefetching module will fetch the instruction at PC+4 and store it in the cache system.