CSE 471 Autumn 1998

Computer Design and Organization

Verilog Program #2

Due: Friday November 6

This assignment is meant to make you understand the operation of a BTB. At the same time it serves as a refresher on the operation of caches.

You will design a BTB (branch target buffer) with 16 entries. The BTB is two-way set associative, i.e., it contains 8 sets of 2 entries each. Each entry consists of the following fields:

- Branch Instruction Tag (BrIT) is the tag allowing the BTB to recognize which branch instruction is to be predicted or whose statistics have to be updated. We assume that all instructions are aligned on word boundaries, hence an instruction address is 30 bits long. With the 3 bits needed to index the BTB, this leaves the BrIT to be 27 bits long.
- Prediction bits (Pr). This is a 2-bit field used for a 2-bit saturating counter (cf. H&P page 502).
- Branch Target field (BrTa). This field is 30 bits long. It stores the target address for a branch predicted taken.
- LRU bit (LRU). When an entry has to be put in the BTB and the corresponding set is full, the LRU entry is replaced.

Inputs to the BTB are:

- An operation field (see below), called OP: 2 bits.
- The PC of the instruction to be predicted or of the branch instruction whose fields in the BTB must be updated, called PC: 30 bits
- The Branch Target Address of the branch instruction whose fields in the BTB must be updated, called BTIN: 30 bits

Outputs from the BTB are:

- BTB hit/miss : HIT (0 = miss, 1 = hit)
- BTB prediction TAK (0 = not taken, 1 = taken)
- Branch Target Address BTOUT : 30 bits

The operations are:

- No-op OP = 0; The need for this OP will become clear for the second part of the assignment
- Predict OP = 1; Check the entry corresponding to PC.
  The possible outputs for the triple (HIT, TAK, BTOUT) are ('x' means undefined)
  - (0,x,x)
  - (1,0,x)
  - (1,1,BrTa)
- Update Not taken OP = 2;
  If the entry corresponding to PC exists in the BTB, "decrement" the saturating counter and set the LRU bit to "point" to the other entry in the set
  Else replace the entry pointed to by the LRU bit with an entry BrIT (corresponding to PC), Pr corresponding to "weak not taken", "x" for BrTa, and the LRU bit pointing to the other entry.
- Update Taken OP = 3;
  If the entry corresponding to PC exists in the BTB, "increment" the saturating counter, set the LRU bit to "point" to the other entry in the set, and set BrTa to BTIN
  Else replace the entry pointed to by the LRU bit with an entry BrIT (corresponding to PC), Pr corresponding to "weak taken", BrTa to BTIN, and the LRU bit pointing to the other entry.

What you have to do

1. Write a Verilog program for the design of the BTB. You should write your program at the behavioral level. In the first design, assume that the BTB handles only one operation in a given time unit. Test your design with various sequences of operations.
   The module should be called *btbseq* **and have parameters listed as below:**
   module btbseq(op,pc,btin,hit,tak,btout);
   A typical testing sequence would consist of a list of operations such as:

```
        1,8'H0000_0800,8'Hx          \\_ is here for readability
        1,8'H0000_0804,8'Hx          \\won't be updated; not a branch instruction
        2,8'H0000_0800,8'Hx          \\update not taken
        1,8'H0000_0800,8'Hx          \\should return (1,0,x)
        1,8'H0000_0808,8'Hx
        3,8'H0000_0808,8'H0000_1234   \\update taken
        1,8'H0000_0808,8'Hx           \\should return (1,1,8'H0000_1234)
etc..
```

   Be sure to test cases where you predict taken and not-taken and where your update confirms or does not confirm your prediction. Also be sure to test the LRU replacement, the setting of the saturating counters etc.

2. Comment your design by explaining what instructions can trigger the various outputs of "Predict" (OP = 1). Give the reason why BrTa has to be set to BTIN when OP = 3 and the entry already exists.

3. Now assume that the BTB can handle concurrently one "Predict" and one "Update" operation in the same time unit. You can assume that an "Update" operation (if any) follows its corresponding "Predict" after 2 time units and that there is no other "Predict" in this interval with the same PC. Modify your first design so that your new design reflects the potential concurrency.
   The module should be called *btbcon* **and have parameters listed as below:**
   module btbseq(op1,pc1,btin1,hit,tak,btout,op2,pc2,btin2);
   A typical testing sequence would consist of a list of operations such as:

```
        1,8'H0000_0800,8'Hx,0,8'Hx,8'Hx
        1,8'H0000_0804,8'Hx,0,8'Hx,8'Hx
        2,8'H0000_0800,8'Hx,1,8'H0000_0808,8'Hx
etc...
```