**ECE486/586**

**Computer Architecture**

**Final Project – Report for Branch and Branch Target Prediction**

**18th March 2013**

**Student Name Student ID**

**Ameya Deswandikar 970606576**

**Nikhil Patil 926474005**

**Portland State University**

*Electrical and Computer Engineering*

*Maseeh College of Engineering and Computer Science*

**CONTENT**

1. Project Specification …………………………………………………………………..
2. Assumptions………………………………………………………………………………..
3. ALPHA Predictor Implementation…………………………………………………
   1. Block Diagram……………………………………………………………………….
   2. Memory Management………………………………………………………….
   3. Flow Chart…………………………………………………………………………….
   4. Testing Strategy………………………………………………………………………..
   5. Test Results…………………………………………………………………………………….
   6. Benchmark Result………………………………………………………………….
4. Enhanced Alpha Predictor with BTB design……………………………………
   1. Flowchart & block diagrams…………………………………………………..
   2. Design Space (Graphs variations Vs performance)…………………
   3. Selection of optimum spec…………………………………………………….
   4. Benchmark Result………………………………………………………………….
5. Development environment…………………………………………………………..
6. Error Handling………………………………………………………………………………
7. Repository……………………………………………………………………………………
8. Appendix………………………………………………………………………………………
   1. Source code………………………………………………………………………….
   2. Trace file……………………………………………………………………………….
   3. Simulation results…………………………………………………………………

**1. Project Specifications**

**Branch and Branch Target Prediction**

1. Branch Predictor - Implementation of Alpha 21264 branch predictor
2. Brach Target Predictor – Design and simulation of branch target predictor

**Design Constraints:**

* ***Memory Limit:*** Maximum 8K bytes of storage (total, including the Alpha predictor)
* ***Table sizes:*** All tables must be sized as powers of two.
* ***Associative tables***: ≤ 8 way
* Random replacement must be reproducible.
* All multiplying or dividing numbers must be in powers of two.

**2. Assumptions**

**3. ALPHA Predictor Implementation**

The Alpha 21264 branch predictor uses both local correlation and global correlation. To incur this ALPHA predictor allocates dedicated memory to maintain local branch history and global branch history. To predict the direction of the current branch the 21264 implements a tournament branch prediction counter to select between the local predictor and global predictor. Together, local and global correlation techniques improves prediction rate.

1. **Block Diagram**

**GLOBAL\_PREDICTOR\_COUNTER**

**( 4096 x 2)**

**LOCAL\_PREDICTOR\_COUNTER**

**(1024 x 3)**

**LOCAL\_HISTORY\_TABLE**

**( 1024 x 10)**

**Program Counter**

**MUX**

**CHOICE\_PREDICTOR\_COUNTER**

**( 4096 x 2)**

**Branch Prediction**

**PATH\_HISTORY\_TABLE ( 1 x 12)**

**Tournament Branch Predictor:**

1. Local Branch Predictor:
   1. A local history table contains the past 10 outcome of the particular branch.
   2. A pattern history table contains 1024 entries.
   3. A 3-bit saturating counter for each entry in the pattern history table.
2. Global Branch Predictor:
   1. A global history register contains the outcome of the past 12 branches.
   2. A pattern history table contains 4096 entries.
   3. A 2-bit saturating counter for each entry in the pattern history table.
3. Choice Predictor Counter:
   1. A 2-bit saturating counter chooses which of the two branch predictors to use for each branch.
4. **Memory Management:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Memory Blocks** | **Indexed By** | **Size (bits)** |
| 1 | LOCAL\_HISTORY\_TABLE | **MEM**[ PROGRAM COUNTER] | 1024 x 10 |
| 2 | LOCAL\_PREDICTOR\_COUNTER | **MEM**[ LOCAL\_HISTORY\_TABLE] | 1024 x 03 |
| 3 | GLOBAL\_PATH HISTRORY\_TABLE |  | 1 x 12 |
| 4 | GLOBAL\_PREDICTOR\_COUNTER | **MEM**[ GLOBAL\_PATH\_HISTORY\_TABLE] | 4096 x 02 |
| 5 | CHOICE\_PREDICTOR\_COUNTER | **MEM**[ GLOBAL\_PATH\_HISTORY\_TABLE] | 4096 x 02 |
|  |  | **TOTAL MEMORY USED** | **29708** |

1. **Flow Chart:**

1. Get Prediction

Taken

Not Taken

Taken

Taken

Not Taken

**Get the global prediction counter value. Is the counter size < Size/2 ?**

**Get the local prediction counter value. Is the counter size < Size/2 ?**

**Choose the predictor ? Is the counter size < Size/2 ?**

**Get the local history Index**

**Branch Record**

**Is branch conditional or return or call?**

**Get the Global History Index**

**Program Counter**

No Yes

yes yes

No No

**2. Update Predictor:**

**Branch Record**

**Update the local history Index**

**Update the Global History Index**

**Program Counter**

**Is branch conditional or return or call?**

Yes

No

Do Nothing

Actual Result

**Update the counter value. If branch was taken, move towards left, otherwise move right**

**Update Choice Predictor Counter**

**Update Global Predictor Counter**

**Update Local Prediction Counter**

1. **Testing Strategy:**

To test the branch predictor code we have considered two cases.

**Case I:** Single branch with execution pattern: **T N N N N N N N N N -----**To Test Local Correlations

**Case II:** 3 correlative conditional branches A, B & C --------------------------To Test Global Correlation

If ( A==0)

{

//Code for A

}

If ( B==0)

{

//Code for B

}

If( A==B)

{

//Code for C

}

If A -> Taken & B-> taken then C-> must be Taken.

1. **Test Results:**

**--** Appendix b.

1. **Benchmark Result:**

**4. Enhanced Alpha Predictor with BTB design**

1. **Design Modifications in Alpha 21264 branch predictor:**
2. **Modified Flow chart**
3. **Design Space (Graphs variations vs performance)**

**5. Development environment**

**6. Error Handling**

**7. Repository**

We used Subversion for source control. The repository is hosted on Google Code and provides up to 5 GB space for hosting projects. All previous revisions can be easily obtained via the repository. It also contains an issue tracking system for managing defects.

**8. Appendix**

1. **Source code:**