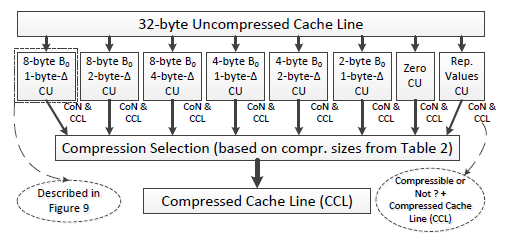
**Base-Delta-Immediate Cache Compression**

**REFERENCE**

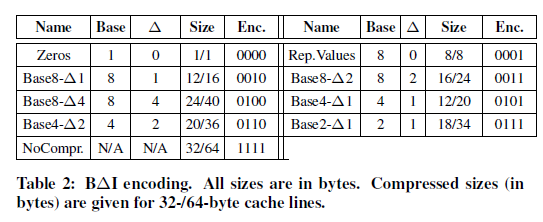
The main reference paper referred for implementation is “**Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches**” and other materials can be found here: <https://github.com/shreya231191/CacheCompression/tree/master/Report/ReferencePaper>

**COMPRESSION**

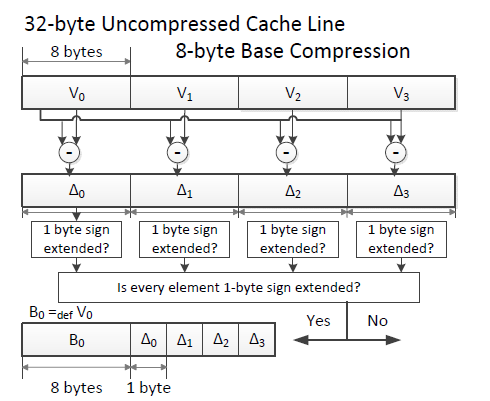
The below block diagram containing a total of 9 methods of compressing a 32bytes cache line was implemented in Xilinx ISE and the compressed cache sizes can be found in the table below.



The base used for calculation of deltas is the first 8,4 or 2 bytes of the cache line.



In order to decompress the deltas accurately flag bits are calculated at the compressor end and appended with the compressed cache line. These flag bits suggest whether the base was greater than the other cache segments or not while calculating the delta. Using this information, the delta is either added or subtracted from the base at the decompressor end.



**MODIFIED BDI**

1. **DYNAMIC BASE:** The cache line could contain incoherent data wherein if the first segment is chosen as the base then not many segments can be compressed. That is as per the original BDI concept, the entire cache line would remain uncompressed as it would get compressed only if all the deltas are uniformly 1,2, or 4 bytes in size.

Now, a base is chosen such that maximum number of segments are compressed using it. This concept is thereby dynamic in the selection of the base in a cache line.

1. **DYNAMIC COMPRESSION**: In order to benefit fully from the concept of dynamic base, the cache segments whose deltas are not uniform with the majority of compressible segment deltas, could be stored in their original length. That is, a dynamic base is chosen that can compress the majority of the segments in a cache line and the ones that cannot be compressed have their entire length deltas stored. To do so the compression status of all the segments is stored along with the flag bits.

The above methods generate meta data bits. Hence, it needs to be considered in the calculation of compressed size of the cache line and then the decision of whether or not to employ them should be taken instead of implementing this extra portion of processing.

Hence, it is decided not to be employed for cache lines with bases of 2 bytes as there would be a total of 16 segments. And meta data of 16 bits for flag bits and 16 for compression status would add an extra of 4 bytes to the compressed cache line.

1. **REPEATED VALUES OF BASE 4**

**RESULTS**

**I)** 8 cache lines of each type were included in the uncompressed data file to get the below result:

No. of uncompressed lines = 40

No. of uncompressed bytes = 2.58KB

No. of compressed bytes= **1.25KB**

Compression Ratio = **2.064**

These results are reflected in the *uncompdata, compdata* and *decompdata* files can be found here:

<https://github.com/shreya231191/CacheCompression/tree/master/DynamicBaseDynamicComp_Base4>

**II)** In order to draw a comparison, the type of cache lines that were benefited by the modified BDI were placed in the uncompressed cache file.

No. of uncompressed lines = 18

No. of uncompressed bytes = 1.16 KB

No. of compressed bytes by original BDI= **0.818KB**

Compression Ratio of original BDI = **1.418**

No. of compressed bytes by modified BDI= **0.688KB**

Compression Ratio of modified BDI = **1.686**

**PENDING WORK:**

1. The modified BDI was only implemented for Base 4 data here: <https://github.com/shreya231191/CacheCompression/tree/master/DynamicBaseDynamicComp_Base4>

Hence, the implementation for Nase 8 is pending.

1. The above concepts need to be tested for benchmark cache data and on FPGA as well.