This paper proposes a mechanism to reduce power spent in fetch and decode operations by trying to reuse the already decoded instructions stored in a Decode Filter Cache. The decode cache thus prevents, to an extent, the repeated fetching and decoding of the same instructions, thus saving power consumption. A predictor mechanism is also used to further improve the likelihood of successfully accessing the DFC.

Strengths of the Paper:

1) The authors of the paper state that the proposed mechanism will provide a 34% reduction in power consumption on average at a performance cost of only 1%. This improvement is important in embedded systems where power is at a premium.

2) Using the IFC along with the DFC helps to better exploit the spatial locality. The IFC will, to a certain extent, avoid degradation from DFC misses, when the next instruction is not decoded yet.

3) The prediction mechanism proposed in the paper tries to do away with the extra pipeline stage and the branch predictor of the Micro-Operation cache. By not having a trace cache, it greatly simplifies the implementation of the DFC.

4) The researchers have tested the mechanism in a simulated environment and have produced meaningful results. The available data can be analyzed further and better improvements can be made to the proposed mechanism.

Weaknesses of the Paper:

1) The paper fails to give more details on certain aspects, thus leaving the reader to wonder about many of the finer points of the cache mechanism details and motivations behind the design. For example, the paper does not detail exactly how profiling was accomplished on the instruction set of the modeled processor (see section 3.2). It states, “execution frequencies of all instructions are obtained from a set of benchmarks”, but it does not explicitly elaborate on this. Were the benchmarks, the same benchmarks used to test the mechanism? The paper is full of rather large leaps like this.

2) Possible implementation details of the proposed mechanism are not provided.

3) The benchmark results are not properly explained. The users are left to wonder why a particular benchmark performs better than the other or what a particular benchmark aims to test.

4) The paper does not consider or identify applications, which might be more dependent on the set of “uncacheable” instructions.

5) The impact on the cycle time, due to the switching between fetching from I-cache and fetching from IFC or DFC is not discussed.
6) The overhead in memory for the operation of the Next Fetch Prediction Table is not mentioned in the experimental setup.
7) The means of selecting values for width and frequency for optimal cacheable/non-cacheable classification is not provided.
8) For the prediction mechanism, the authors explain scenarios where mis-predictions occur, but they do not specify any methods to handle mis-predictions.