Transformers: the advent of Dark Silicon

Anshul Bansal, Girish Mururu
Georgia Institute of Technology

Introduction

1. Gordon Moore law & Denard Scaling law:
2. Performance trends 2x every 1.5 years
3. power requirements $\propto$ area for transistors
4. Denard scaling broke & Increase in frequency produces heat proportional to that of nuclear reactor
5. Current Trend: multi-core processors

The Advent of Dark Silicon

6. Infinite parallelism?? Infinite Power ??: No both are limited creating dark silicon[1]
7. **What do we have??** - lots of hardware which is not powered on
8. **What can we do??** - different types of micro-architecture on the same silicon
9. **What are we trying??** - hardware which adapts to the application running by changing its micro-architecture dynamically
10. **The rise of Transformers**

Transformers

1. Different applications show more affinity to some micro-architectures than others
2. The transformer is a mechanism of managing multiple micro-architectures in a given stage
3. Not all the micro-architectures are kept running
4. Transformer turns on the required micro-architecture for the application phase
5. Bumblee-bee is a Branch-prediction Transformer and Optimus-Prime is a Cache Replacement policy transformer

1. Bumblee-Bee - three different branch predictors, gshare, gskewed & bimode
2. Optimus-Prime - transforms between LRU and DRRIP[2]

Bumblee-Bee

1. On $misspredictionrate > threshold$, start all branch predictors to check which performs better
2. Shift to the one performing better and go to step2 on exceeding threshold
3. Problem of stale data in off branch-predictors is solved by Birthday Paradox [3]
4. Gives number of predictions needed to ensure the same entry is hit again on avg.

$$Tk = k\sqrt{K!} * \Gamma(1 + 1/k) * N(1 - 1/k)$$
$$T2 = 1.25\sqrt{N}$$

Experimental Results

1. We implemented our system using Macsim simulator [6]
2. The simulator is modeled on Intel Sandy Bridge type architecture
3. We used SPEC2006 workloads

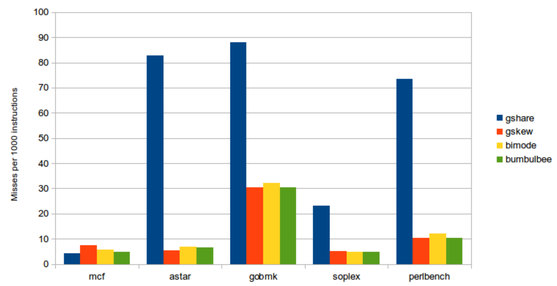

Figure 1: Branch Prediction Performance

4. The Bumble-bee transformer always try to achieve the performance of the best individual predictor as shown in Figure 1
5. The misprediction rate threshold considered here is 25%
6. The analysis of the number of times the branch predictor switch happens for different values of threshold is as shown in 2

Optimus-Prime

1. Best cache replacement policy is dependent on the type of the application being run [4]
2. In order to choose between LRU and DRRIP we use the concept of Set-Dueling[5]
3. Leader sets, (32 in our case) are allocated each to LRU and DRRIP
4. Meta data of the follower sets is not updated for the policy which is turned off
5. Set-dueling counter tells whether to continue using the same policy or shift to the other policy

Experimental Results -II

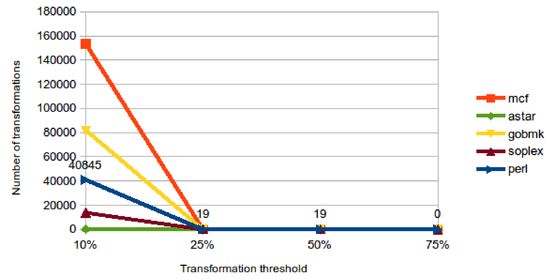

Figure 2: Number of branch predictor switches

1. The Optimus-Prime Cache Replacement Transformer was used for L3 cache replacement
2. It gets best of both LRU(Least Recently Used) and DRRIP[2] replacement policy; it is shown in figure 3

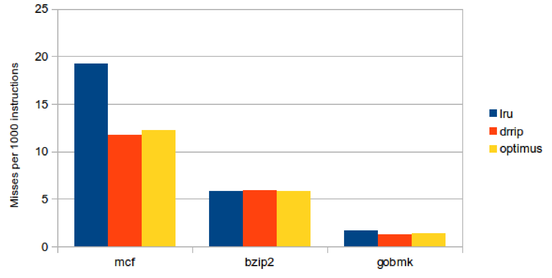

Figure 3: Cache Performance

Conclusion and Future Work

1. Another way of looking at the bright side of dark silicon
2. Leveraging it to achieve higher performance by adjusting to the application behavior
3. Future work can involve the analysis of the transformer when multiple applications are scheduled by OS for execution
4. Power analysis in detail
5. Extrapolation of the above design can be acclimated to other stages of pipeline like execution stage,fetch stage etc.

References

[1] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, pages 365–376, New York, NY, USA, 2011. ACM.

[2] Aamer Jaleel, Kevin B. Theobald, Simon C. Steely, Jr., and Joel Emer. High performance cache replacement using re-reference interval prediction (rrip). *SIGARCH Comput. Archit. News*, 38(3):60–71, June 2010.

[3] Philippe Flajolet, Danièle Gardy, and Loÿs Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Appl. Math.*, 39(3):207–229, November 1992.

[4] Aamer Jaleel, William Hasenplaugh, Moinuddin Qureshi, Julien Sebot, Simon Steely, Jr., and Joel Emer. Adaptive insertion policies for managing shared caches. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, pages 208–219, New York, NY, USA, 2008. ACM.

[5] Moinuddin K. Qureshi, Aamer Jaleel, Yale N. Patt, Simon C. Steely, and Joel Emer. Adaptive insertion policies for high performance caching. In *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, pages 381–391, New York, NY, USA, 2007. ACM.

[6] *Kim, H., Lee, J., Lakshminarayana, N. B., Lim, J., and Pho, T. 2012. MacSim: a simulator for heterogeneous architecture. https://code.google.com/p/macsim.*