A Hardware Solution for Elimination of Rampant Speculations in Levo

Alireza Khalafi
Northeastern University

September 5, 2002

Introduction

Resource Flow Execution models has proved to be an effective way for extracting higher levels of ILP in sequential code. Although we anticipated that intermediate speculative values might help to boost the performance through a last-value prediction scheme, in practice this turned out to be less effective. The intermediate values, generated by Rampant Speculation, lead to unnecessary executions and writes on the forwarding buses. This will waste the available bandwidth, as well as taking away the opportunity for sophisticated value prediction schemes. The speculative values tend to overwrite the predicted values and eliminate their effect.

In this memo, I propose Early Notification as a technique for eliminating rampant speculation. I will show that the increase in hardware cost is minor considering the potential of the performance improvement.

Early Notification Mechanism

Figure 1 shows a portion of the Levo execution window with the associated sharing groups and forwarding buses. For simplicity only one of the forwarding buses is shown. In a resource flow execution model, each AS will forward its output operand value which is snooped and potentially snarfed by subsequent active stations with the same operand address. As an example in figure 1, assume that instructions in AS_1 and AS_3 both have the same output operand r_x. Further, assume that instruction in AS_7 also has r_x as one of its input operands. Now if AS_1 forwards a value $r_{x,1}$ on the forwarding bus, AS_7 will snarf this value, re-execute and forward its output operand value upon a change in its value. Later, AS_3 will forward a new value $r_{x,3}$ which will again is snarfed by AS_7 and will possibly cause another forwarding transaction. The main reason that AS_7 had to re-execute is because it did NOT know that AS_3 has the same output operand as AS_1.

I introduce the concept of Early Notification as a means by which AS_3 will notify the later instructions that it has the same output operand address as AS_1. This way, all the AS's after AS_3 will ignore the forwarded value and will wait for the forward from AS_3 or a later AS.

Implementation of EN is fairly simple. A single bit bus (the red lines in the figure 1) is originated from each sharing group *in a bus span* and extends to the bottom of the bus span. We call this single bit line an *EN signal*. Note that for each forwarding bus, there is a separate EN signal originating form each sharing group. If an AS snoops an earlier forwarded value with the same operand address as its output operand address, it will set its EN signal to one. The EN signals from pervious SG's in the *same* bus span are ORed together and checked by later active stations. If the value of the input EN signal to a SG station is 1, the AS's in that sharing group will ignore the forwarded value. The EN signals from all SG's are also ORed together and fed into the FU at the bottom of the bus span. When a FU receives the forwarded value, it also ignores the forwarded value.

An issue that needs to be addressed, is the timing requirement for EN signals. The EN signals can only be set after an AS snoops and verifies that the forwarded operands has the same address as its output operand. In Levo, this will take one cycle and therefore EN signal might not be set until end of the current bus cycles. For this reason, we assume that EN signal is snooped on the next bus cycle. In other words, each AS that snarfes an operand, will check the input EN signals on the next bus cycle. If the EN signal is 1, the AS will discard the snarfed value. This method, however, does not impose any extra delays. Even if the AS has already send the value for execution, it can either abort it or discard the result. In short, the EN signal can be safely snooped on the next bus cycle without imposing any extra delays.

In this proposal I have assumed that each EN signal originates from a sharing group. although in reality each active stations needs to send a EN signal. This is to reduce the cost of implementing EN. It is quite possible for the active stations in a sharing group to use a local simplified version of EN between themselves and use a shared output EN signal for the whole SG. Figure 2 shows one example of how this can be implemented.

The cost for implementing EN is fairly low. Assuming *n* sharing groups in each bus span, for each forwarding bus, we only need *n* EN signals. For larger bus spans, if we assume the bus delay is much larger than gate delay, it is possible to reduce the number of signals to *n/p* by inserting an OR gate between every *p* sharing groups.

Advantages of Using EN

The following are among main advantages:

- Reduced bus activity reduces the required bandwidth
- Reduction in the amount of re-executions. This is specially of interest because it might provide a reason to share PE's across columns which will reduce the cost and power consumptions.
- Providing opportunities for sophisticated value prediction techniques.

EN can be easily extended to support value prediction. Each time that an speculative value is forwarded, we need to stamp this transactions as speculative versus a normal transaction. The AS's will not raise an EN signal in response to an speculative transactions. As a result, the later instructions will use the speculative value to execute ahead of time and forward their results. If later on they receive a non-speculative data, they compare its value with the older value and if they are the equal, there is no need to forward a new result.

Predicated Execution

EN scheme should only be applied to Register and Memory operations. We would like to keep the speculative nature of predicated execution intact in order to allow for control independence. Note that due to the fewer speculative changes in the register values, we expect less intermediate changes in the predicate values as well. If an active station becomes disabled, we use the regular resource flow rules (nullifying and relay forwarding) to resolve dependencies.

Summary

Early Notification is a simple and cost effective enhancement for reducing the rampant speculation in Levo Microarchitecture. It also opens opportunities for applying other optimizations.

Figure 1

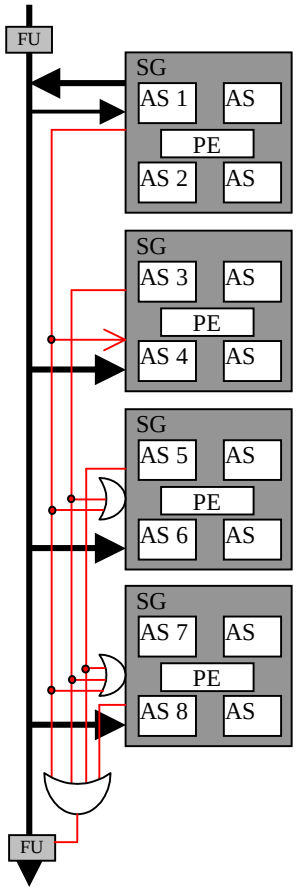

Figure 2