About the Relevance of Multispeculation in High-Level Synthesis

***Abstract*—Nowadays circuits possess stringent area or power constraints. Nevertheless, the increase of performance is still an obligation while designing them. The recent appearance of Variable Latency Functional Units (VLFUs) has raised the possibilities for designers because they offer a good tradeoff. However, the VLFU behaviour depends on the inputs, so the use of many of them increases the probability of working in long latency mode. In this paper we propose the use of Multispeculative FUs to improve performance with a negligible area penalty. By judiciously speculating the carry vectors of concrete operations, and hiding the possible mispredictions, the probability of working in long latency mode is reduced. Hence, the average latency is diminished and thus performance is increased. Our experiments show a 24% improvement on execution time when considering logarithmic modules.**

Keywords: multispeculation, additive trees, HLS

I. INTRODUCTION

Additive structures are the basis of nowadays' datapaths. Addition chains or trees usually appear in signal processing applications such as: ECG, numerical integration methods, filters, multimedia applications, etc [2-3]. Therefore, it is crucial to improve the quality of adders and adder-dominated structures without incurring significant area or power overhead. The optimization of these structures has been historically performed with fast Fixed Latency FUs [1] or with Carry Save Adders (CSAs). Nevertheless, despite recent contributions towards Dataflow Graph (DFG) transformations to optimize the use of CSAs [4], the application of these structures requires the appearance of similar cluster of nodes in the DFG for reusing them and thus avoiding an excessive overhead. The introduction of Variable Latency Functional Units (VLFUs) has augmented the possibilities in the design space [5-7]. Opposite to the aforementioned fixed latency FUs, VLFUs are characterized by a low area/power and the varying total number of cycles to complete a computation, depending on the specific inputs. The performance of the datapath will be higher if the VLFUs work in the short latency mode as many times as possible. However, they are still subject to the long latency mode. Moreover, when utilizing several VLFUs the probability of working in long latency mode for any of them will increase, penalizing thus the datapath latency. Hence, it is critical to devise mechanisms able to hide possible penalties due to VLFUs working in long latency mode.

Multispeculative FUs (MSFUs) are VLFUs based on various forms of speculation over the carry signals. This carry speculation helps to shorten the critical path of the FU. The average case performance of these units is determined by the hit rate of the prediction. However, in spite of utilizing more than a predictor, none or only one additional cycle is enough for producing the correct result in the majority of the cases [5-6]. Our proposal consists of utilizing MSFUs to reduce the average latency of the circuit, and applying some techniques to hide the long latency penalties when possible.

II. MULTISPECULATIVE DATAPATHS

An n-bit Multispeculative Adder (MSADD) is composed of n/k k-bit fragments interconnected with n/k-1 predictors [7]. In order to implement datapaths, we apply Static Zero Prediction (SZP) in some points of the DFG, so in our case predictors are simple D-flipflops. These elements will be used to save the intermediate carries and pipeline them from a cstep to the following during the first cycles, avoiding thus the additional penalty cycles during this first set of operations. Finally in the last cycles, predictors will be utilized as usual, reducing the critical path of the last stage, which is the main limitation of CSA structures, and thus increasing performance. In this last stage, SZP will be assumed. Hence, in the last cycles a failure will be produced iff a carry-out from any fragment is '1', i.e. different from the prediction. In this last cstep/state a control mechanism similar to [7] is utilized, i.e. if the *hit* signal is false there will be a transition to an intermediate or *correction* state, and otherwise there will be a transition to the following state.

A comparison between a conventional flow and our proposal is illustrated in Figure 1. Let us consider non-speculative FUs and MSFUs. Note that MS-multipliers (MSMULs) are composed of a CSA tree and an MSADD in the last stage. And let´s suppose a latency of 4 and 2 cycles for the non-speculative FUs and 3 and 1 for the MSMULs and MSADD, respectively. Figure 1 a) shows the DFG of the Discrete Wavelet Transform [2], while figure 1 b) depicts its corresponding scheduling and binding with non-speculative FUs, which takes 28 cycles with 1 adder and 2 multipliers. Figures 1 c) to 1 e) show the application of our methodology. First, we identify the additive trees in the DFG, as it can be observed in figure 1 c). These additive trees are only composed of additions, but for the leaves, that can be products too. Second, a recovery addition per tree is introduced, as depicted in figure 1 d) with the operations labeled with '. Finally, the additive trees are scheduled and bound applying a modified combined resource constrained scheduling and binding, as shown in figure 1 e). Dotted lines indicate how the intermediate carries are pipelined from an operation to the following. Recovery additions will apply SZP until a correct result is reached. According to previous studies [5-7], they will be required none or once at the most, with a high probability. Hence, if the original operations (those labeled without a ') produce a hit, the csteps that are only composed of recovery additions can be skipped in execution time. In figure 1 e), these csteps are those labeled

with ‘. For instance, if ***Operation 7*** hits, cstep 10’ is not necessary. If the cstep contains conventional operations too, they will provide a *recovery slot* to hide one additional cycle. This is the case of csteps 6, 8 and 12 with ***Operations 4***, ***8*** and ***12***. Thereby, provided that a cycle is enough for correcting the results, the latency of our proposed implementation will range between 19 and 22 cycles, providing besides 3 recovery slots for 3 of the 6 recovery additions in the datapath.

III. Experimental Results

Several benchmarks have been simulated and synthesized with a 65 nm library, utilizing *Design Compiler* by *Synopsys*. Our results considering non-speculative and MS logarithmic modules can be observed in figure 2, where the percentage variation in execution time, area and Area Delay Product (ADP) are shown. The baseline flow is composed of conventional list-scheduling and left-edge binding algorithms using non-speculative FUs [2-3].

As it is shown, MS results improve execution time by 24% on average (44% in the best case). The average area penalty is close to zero because the reduction in FUs area is compensated for the additional routing and control. Finally, when considering ADP, there is a 24% reduction.

IV. Conclusions

This work describes the application of multispeculation over additive trees. The main idea consists of pipelining the carry vectors in the inner nodes of the trees. Thanks to the associative property of addition, carry vectors that cannot be consumed by active adders can be accumulated at a later stage. The last stage delay will be reduced by multispeculating these carry vectors. Results show that multispeculation open a new

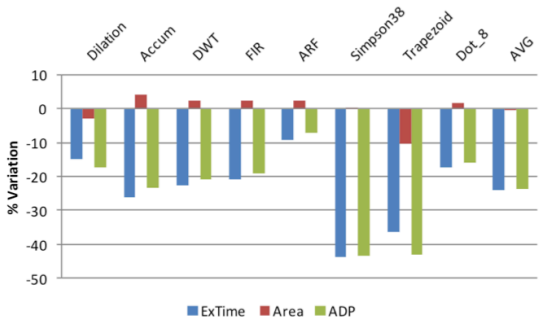

Figure 2. Execution time gain, area penalty and area per delay variation percentages for logarithmic FUs

window of possibilities in datapath design, as the study of the number of required predictors or the development of new MSFUs able to provide new optimizations to the datapaths.

References

[1] I. Koren, “Computer Arithmetic Algorithms”, A K Peters, 2nd ed, 2002.

[2] S.P. Mohanty, N. Ranganathan, E. Kougianos, P. Patra. “Low-Power High-Level Synthesis for Nanoscale CMOS”, Springer, 2008.

[3] P. Coussy, A. Morawiec, “High-Level Synthesis. From Algorithm to Circuit Design.”, Springer, 2008.

[4] H. Parandeh-Afshar et al., “Improving FPGA Performance for Carry-Save Arithmetic”, IEEE TVLSI, Vol. 18, no. 4, pp. 578-590, April 2010.

[5] S.M. Nowick, “Design of a low-latency asynchronous adder using speculative completion”, IEE Proc. Comput. Digit. Tech., 1996, vol 143, no. 5, pp. 301-307.

[6] A.K. Verma et al., “Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design”, DATE 2008, pp. 1250-1255.

[7] A.A. Del Barrio et al., “Multispeculative Addition Applied to Datapath Synthesis”, IEEE TCAD, 2012 (on press).

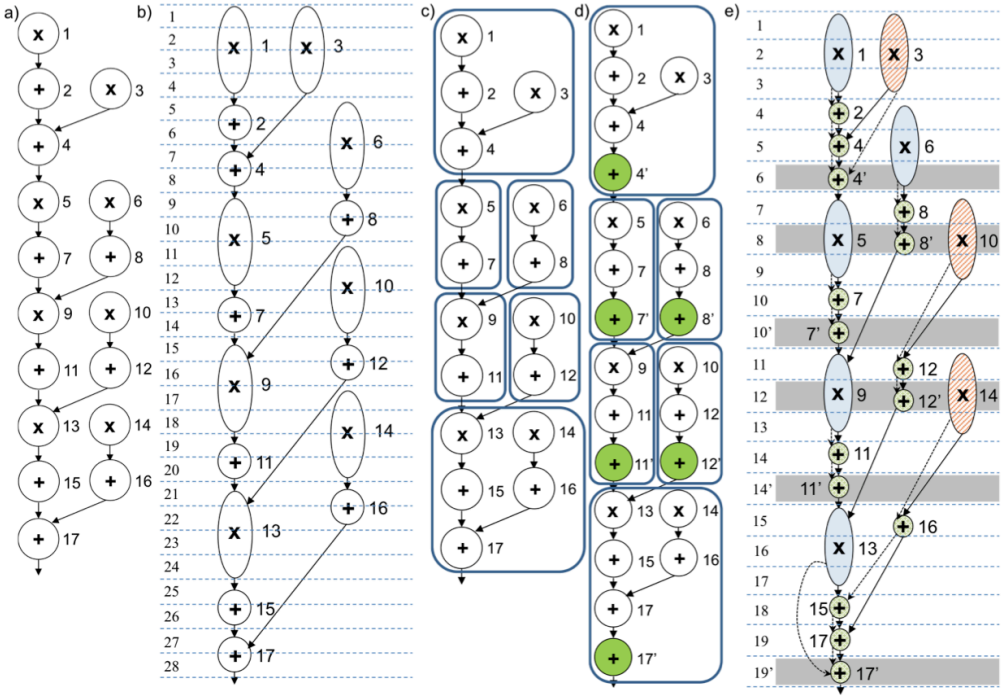

Figure 1. a) Discrete Wavelet Transform (DWT) DFG [2], b) conventional scheduling with non-speculative FUs, c) additive structures in the DWT DFG without and d) with the recovery additions, e) scheduling and FU-binding with multicycle MSFUs