Microarchitecture-Aware Floorplanning Using a Statistical Design of Experiments Approach*

Vidyasagar Nookala Ying Chen David J. Lilja Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN
{vidya,wildfire,lilja,sachin}@ece.umn.edu

ABSTRACT

Since across-chip interconnect delays can exceed a clock cycle in nanometer technologies, it has become essential in high performance designs to add flip-flops on wires with multi-cycle delays. Although such a wire pipelining strategy allows higher operating frequencies, it can reduce the delivered performance of a microarchitecture, since the extra flip-flops inserted may increase the operation latencies and stall cycles. Moreover, the addition of latencies on some wires can have a large impact on the overall performance while other wires are relatively insensitive to additional latencies. This varying sensitivity suggests the need for a throughput-aware strategy for pipelining the interconnects that interacts closely with the physical design step, which determines the lengths of these multicycle wires. We use a statistical design of experiments strategy based on a multifactorial design, which intelligently uses a limited number of simulations to rank the importance of the wires. When applied at the floorplanning level, our results show improvements both in the overall system performance and in the total wire length when compared with an existing technique.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*

General Terms

Performance, Experimentation

Keywords

Wire pipelining, Microarchitecture, Floorplanning

1. INTRODUCTION

Industry trends indicate that the operating frequencies of leading-edge microprocessors have been doubling with every process generation [1], having broken the gigahertz barrier several years ago. These improvements in the clock frequency are complicated by the fact that even under aggressive optimization, the delay of a long inter-block interconnect may exceed the system clock cycle. To ensure that the system operates at the right frequency, the delay of such a global wire is distributed over several clock cycles by inserting memory elements such as edge-triggered flip-flops [2, 3]. However, it has been noted that this approach, which is referred to as *wire-pipelining* henceforth, can alter the functionality of a circuit [4] by arbitrarily changing the latency differentials along the paths in the circuit. Even if these differentials are corrected, an additional concern is the reduction in the throughput of the circuit due to the increase in the number of clock cycles per computation. In the microprocessor context, this can be explained as follows. The execution time of a particular program on a microprocessor can be expressed as the product of two terms:

$$T_{exec} = N_{cycles} \cdot T_{clk} \quad (1)$$

where N_{cycles} is number of clock cycles required to execute the instruction sequence of the program, and T_{clk} is the clock cycle time. The effects of pipelining the buses of a microprocessor can cause changes in data arrival times, which can be absorbed internally by the processor through, for instance, stalls in the instruction pipeline. This may result in an increase in some operation latencies, branch misprediction and cache miss penalties, etc., thus increasing N_{cycles}. This penalty depends on the locations at which these extra latencies are added: increasing the latencies on some buses can impact the throughput more than on others. This reduction in the system throughput is a concern that can negate the advantage of operating at higher frequencies.

For better throughput, it is imperative to reduce the extent of pipelining required by the critical buses, and therefore, physical design must focus on keeping the throughput-critical buses as short as possible. Traditional physical design, which typically focuses on minimizing the area and total wire length, can lead to throughput-suboptimal processor layouts in the wire-pipelining regime. Therefore, a throughput-aware strategy [5], which identifies and optimizes throughput-critical buses, particularly at early stages of the physical design flow, where the system bus delays are determined, is the need of the hour.

The clear bottleneck in the design flow is the estimation of N_{cycles}, or alternatively, the estimation of the throughput, which is inversely proportional to N_{cycles}. For a particular combination of bus latencies, this can be computed using cycle-accurate simulations on simulators such as SimpleScalar [6], on widely-used benchmark programs such as SPEC [7]. However, cycle-accurate simulations are inherently slow, and in extreme cases, a single run can take several days to complete. This, coupled with the large search space explored during physical design optimizations, makes it virtually impossible to use simulations for each layout that is to be evaluated. Specifically, if each of n wires on a layout can have k possible latencies, then the cycle-accurate simulator may have to perform up to n^k simulations to fully explore the search space.

*This work was supported in part by a gift from Intel Corporation, by the NSF under award CCCR-0205227, by the Minnesota Supercomputing Institute, and by the University of Minnesota Digital Technology center.

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

There have been some recent attempts [8, 9] towards throughput-aware design at the floorplanning level. In [8], a throughput look-up table (LUT), indexed by the set of bus latencies, is constructed using cycle-accurate simulations. For a given layout (and the corresponding bus latencies), the throughput is evaluated from the LUT using some distance metrics. In contrast, the approach in [9] assigns weights to each of the system buses that are proportional to the amount of traffic seen on the buses, operating under the notion that the more often a bus is accessed, the more critical it is. The objective of the floorplanner then is to minimize a weighted sum of bus latencies, where the weights depend on the amount of traffic. While the two approaches indicate welcome progress in the quest for throughput-aware design, the accuracy of the strategies used to optimize the throughput-critical wires shows room for improvement. For instance, estimating the throughput of the thousands of layouts evaluated during floorplanning from an LUT of about 50 entries may not be accurate enough, no matter how well the LUT entries are chosen. In addition, the LUT has to be reconstructed if a different frequency is chosen. On the other hand, bus access frequencies may not exactly capture the quantitative impact of the bus latencies on the throughput. Specifically, the effect of extra latencies on the execution path of a particular operation is primarily determined by the dependencies the following instructions have on the data generated by that operation. Another recent work [10] uses a "one-at-a-time" approach to build throughput sensitivity models for a few selected critical paths, and these models guide the floorplanner to optimize the system throughput. However, such one-at-a-time approaches may not provide an effective sampling that captures the essence of a large solution search space.

The exponentially large search space prompts us to consider a design of experiments [11] strategy, a well-established approach that is particularly efficient at extracting the basic characteristics of a large design space through a small number of samples. Specifically, this paper proposes a strategy, based on a multifactorial design, to accurately identify the throughput-critical wires to be optimized in physical design, and then applies the methodology to floorplanning. The advantage of this approach is that the total number of simulations required to sample the space is proportional to n, compared to the $O(n^k)$ possible combinations of bus latencies. The throughput-critical wires are explicitly identified and appropriately weighted in the objective function for floorplanning. A vital ingredient of our approach that speeds up our design of experiments approach is a fast method for cycle-accurate simulation using the MinneSPEC reduced input sets [12].

The remainder of the paper is organized as follows. Section 2 describes the design flow, the wire-pipelining models used and the simulation strategy. Section 3 presents the experimental results obtained, along with a comparison with the access frequency based approach of [9], when our methodology is used at the floorplanning level. We conclude the paper in section 4.

2. THROUGHPUT-AWARE FLOORPLANNING

To incorporate wire-pipelining issues into floorplanning, we develop a design flow for throughput-awareness, as depicted in Figure 1. The first step is to quantify the impact of each system bus on the performance for each of the chosen benchmark programs, and accordingly assign weights to these wires. The weights may differ across the benchmarks, depending upon the instruction mix executed. The concept of using these weights is similar to [9], but the precise manner in which we choose these weights is different. A comparison between the two approaches is shown in Section 3.

The weights are then fed to the floorplanner, along with a target frequency. The objective of the floorplanner is to determine the positions of the blocks such that a weighted sum of bus latencies, in addition to traditional objectives such as area and aspect ratio, is minimized. The performance of the resultant layout is then estimated from cycle-accurate simulations. If frequency is a design variable, then the floorplanning may be repeated for several frequencies until an optimum design point or performance objective is achieved. In addition, the entire design flow of Figure 1 may be repeated for several microarchitectural block configurations to identify the optimal configuration [13]. For a general case, the weights to be used in floorplanning may be obtained by combining the weights obtained from optimizing the processor performance on a set of benchmarks. The rest of this section explains each step of the flow in detail.

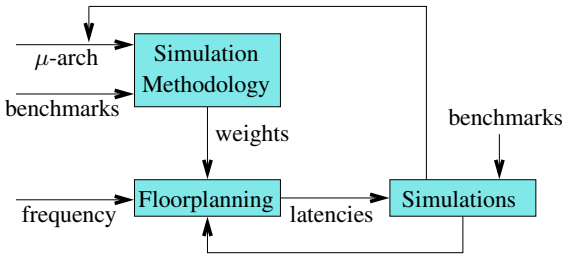

Figure 1: Throughput-aware floorplanning: design flow.

2.1 Wire-pipelining models and simulator

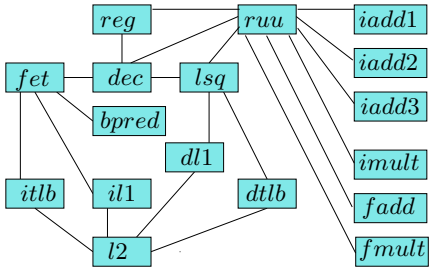

Figure 2: Microarchitecture and buses.

In this paper, we utilize SimpleScalar, a widely-used out-of-order superscalar processor simulator that implements the DLX architecture. The microarchitecture of the processor that we use is shown in Figure 2. The instruction fetch and decode blocks are shown as fet and dec, respectively, while $il1$ and $dl1$ are the level-1 instruction and data caches, respectively. The instruction and data translation look-aside buffers (TLB) are indicated as $itlb$ and $dtlb$, respectively, while $l2$ represents the unified level-2 cache. The block ruu is the register update unit, which contains the reservation stations and instruction issue logic, while the block lsq represents the load store queue. The system register file is represented by reg, whereas $bpred$ consists of the branch predictor and the target buffer (BTB), which predict the direction and target address for a branch instruction, respectively. The blocks $iadd1$, $iadd2$, $iadd3$, $imult$, $fadd$ and $fmult$ are the functional units that execute arithmetic and logic instructions. The figure also shows the 22 system buses that can impact the processor performance, when pipelined.

The simulator is modified to include extra latencies on these buses as additional delays. To achieve this, we use 19 factors to model the 22 buses, as shown in Table 1. As can be seen in Table 1, most of the factors directly model the buses with the same name. The first exception is the factor $extra\_fet$, which models the sum of the latencies of three buses, as shown in Table 1. This factor represents the number of extra stages to be inserted in the *fetch* stage of

Bus	Parameter	ID
fet_il1		
$il1\_bpred$	$extra\_fet$	1
fet_bpred		
fet_itlb	fet_itlb	2
$itlb\_l2$	$itlb\_l2$	3
ruu_reg	ruu_reg	4
ruu_lsq	ruu_lsq	5
ruu_iadd1	$iadd\_add1$	6
ruu_iadd2	ruu_iadd2	7
ruu_iadd3	ruu_iadd3	8
ruu_imult	ruu_imult	9
ruu_fadd	ruu_fadd	10
ruu_fmult	ruu_fmult	11
lsq_dl1	lsq_dl1	12
$dtlb\_l2$	$dtlb\_l2$	13
lsq_dtlb	lsq_dtlb	14
$il1\_l2$	$il1\_l2$	15
$dl1\_l2$	$dl1\_l2$	16
dec_reg	dec_reg	17
fet_dec	fet_dec	18
dec_ruu		
dec_lsq	max_lsq_ruu	19

Table 1: Buses and factors.

the pipeline of the processor. The second and last exception is the factor max_lsq_ruu, which represents the maximum of the latencies of the buses dec_ruu and dec_lsq. We omit the details of the wire-pipeline models due to space constraints. Each of the factors is made completely configurable by modifying the SimpleScalar configuration file.

2.2 Simulation methodology

The benchmarks that are used in this work are selected from the SPEC 2000 benchmark suite. However, it is well known that the simulation of these benchmarks is very compute-intensive: for instance, the reported simulation time for the benchmark 177.mesa, when the reference input set used is about 3000 hours on a SPARC 333MHz machine. For our purposes, since the results of the simulation are used for purposes of optimization, we require fidelity rather than absolute accuracy[1]. The MinneSPEC reduced input sets [12] constitute a representative workload for the SPEC benchmarks, which correlates well with the full input set for these benchmarks and captures their global characteristics. The advantage of these test sets over conventional simulation speedup techniques such as fast-forwarding is the high accuracy in capturing the behavior of the reference input sets. Specifically, the throughput variation across different microarchitectural configurations for the reference input sets is most likely to be retained when MinneSPEC sets are used, while dramatically reducing the simulation times. The simulation time for 177.mesa, for instance, reduces to about 7 hours. Therefore, MinneSPEC is ideal for our purposes where fast simulation with high fidelity is adequate to determine the weights of various system buses.

We choose a set of nine benchmarks, which, along with the type and instruction count, are shown in Table 2. The benchmarks were chosen because of their distinct instruction mixes. For instance, 177.mesa has a high percentage of conditional branches, while 181.mcf has a very large number of memory operations, particularly "store" instructions. All benchmarks are complied at optimization level O3 using the SimpleScalar version of the gcc compiler and are run to completion.

Benchmark	Type	Instr. count (M)
164.gzip	Integer	1065
175.vpr	Integer	217
176.gcc	Integer	693
177.mesa	Floating-point	309
179.art	Floating-point	7700
181.mcf	Integer	175
183.equake	Floating-point	716
197.parser	Integer	914
256.bzip2	Integer	3800

Table 2: Benchmarks from the MinneSPEC suite.

2.3 Our design of experiments based strategy

Statistical *design of experiments* is a design approach to characterize the response of a system in terms of changes in the factors which influence the system. The basic idea is to conduct a set of experiments, according to a given prescription, in which all factors are varied systematically over a specified range of acceptable values, such that the experiments provide an appropriate sampling of the entire search space. The subsequent analysis of the resulting experimental data will identify the critical factors, the presence of interactions between the factors, etc. The influence of the individual factors is expressed as *main effects*, while *interaction effects* describe the influence of interactions. For a system affected by N factors, there are N main effects, $\binom{N}{2}$ two-factor interaction effects, and so on. In all, there are $2^N - 1$ effects that must be estimated. The simplest design, commonly referred to as *full factorial design*, permits estimation of all of the main and interaction effects. However, such a design involves experimenting over all combinations of the possible values subscribed by the factors. As noted earlier, the number of possible bus latency configurations in floorplanning is an exponential function of the number of factors. Even though the number of factors N is relatively small ($N = 19$) for this microarchitecture, given the high simulation times (even under MinneSPEC), it is impractical to use cycle-accurate simulations for each of the allowable configurations to determine the response, which in this case is the number of clock cycles to execute a program, that needs to be minimized (to maximize the throughput, its reciprocal).

We address the problem of reducing the number of simulations with a few assumptions. Each of the factors is restricted to have two levels : the minimum and the maximum possible values for the factor, thereby permitting us to employ a two-level factorial design. The idea is that, by stimulating the system with inputs at their extreme values, we provoke the greatest response for each input. The assumption is that the system response is a monotone function of changes in the inputs (factor levels). While this assumption cannot be guaranteed in these types of systems, it works quite well in practice[2]. Besides, higher level designs exhibit a complex effect structure and require more simulations, which make them unreasonable for studies like ours. As is shown in [14], the two-level approach can be effectively used to design simulation strategies for microarchitectural optimizations. Since the factor levels represent bus latencies, the extreme (high and low) values can be obtained by assuming worst-case and best-case scenarios for the corresponding wire lengths. The high/low value for a bus latency may be determined by placing the connecting blocks as far/close as possible. A

[1] In the context of physical design, this concept is not new: the Elmore delay metric has been used widely because of its ability to determine the relative impact of a change, even though the absolute accuracy may be limited.

[2] Although this is not a proof, it seems intuitively acceptable to believe that increasing the latency of a bus will decrease the system throughput.

valid assignment may, for example, be 0 for the low value, and the latency corresponding to a corner-to-corner connection across the chip for the high value.

Interactions: We have identified a few potential significant interactions, which resulted from the nature of wire-pipelining models integrated into the simulator.

- We have incorporated functional unit scheduling in the simulator. Specifically, the number of latencies inserted on the three buses ruu_iadd1, ruu_iadd2 and ruu_iadd3 can be different, and while issuing an integer add instruction, of all the available units, the one with the least latency is chosen. This indicates possible significant (two and three factor) interactions, which need to be estimated.
- In the decode stage, the number of extra pipeline stages to be inserted is modeled as a maximum function of three factors dec_ruu, max_lsq_ruu and ruu_reg (refer to Table 1). Such a nonlinear function can result in significant (two and three) factor interactions among these three factors.

We assume that all of the other interactions are negligible, allowing us to utilize a resolution III fractional factorial design [11], which provides the logically minimum number of experiments to determine the main effects of the factors. For N factors, the number of experiments required is equal to the nearest highest power of 2, which turns out to be 32 for our work, since $N = 19$. The design is captured by a simulation matrix M of size 32xN, where each of the 32 rows corresponds to a simulation run. In general, in a two-level design, the two levels of each factor are represented as {+1, –1}, and the idea is to estimate the effect of changing the level of the factor from "+1" to "–1". Each level (± 1) is contained in exactly half of the simulation runs, indicating that each column (which corresponds to a factor) of M has 16 "+1"s and 16 "–1"s. The effect of a factor is determined as the difference of the responses where its level is "+1" and those where its level is "–1". We refer the reader to [11] for further details.

Although, a resolution III design is useful only for cases where there are no interactions, due to the *aliasing* of the main effects with the interaction effects, a few flexibilities permit us to estimate a few selected interactions, such as those of the previous paragraph. Specifically, a resolution III design of size (number of experiments) 32 can actually be used for up to 31 factors. Since we only have 19 factors, the remaining 12 can be treated as dummy factors. The desired interaction effects can be estimated by aliasing them with some of the dummy factors, which typically have negligible (or zero) effects. In addition, it can be noted that the two-factor interaction effects of $\{ruu\_iadd1, ruu\_iadd2, ruu\_iadd3\}$ must be equal due to symmetry. Therefore, estimating one of them will suffice, and this is true for the corresponding main effects as well.

The advantage of fractional factorial resolution III designs over other screening designs such as Plackett and Burman (PB) [15], which is employed in [14], is the well defined aliasing structure. This attribute can be used to estimate a few required interactions, as is done in this project, at the expense of a few additional simulations[3]. Finally, if more interactions need to be considered in the design, and the number of dummy factors is inadequate, then one option is to perform appropriate additional orthogonal runs [16]. If this is not sufficient, then a higher resolution (IV or more) design can be utilized, if the associated simulation cost[4] can be tolerated.

[3]For N factors, the number of experiments required in a PB design is equal to the next highest multiple of 4 (20 for 19 factors in this work), unlike the nearest highest power of 2 in a resolution III fractional factorial design used in this work.

[4]A resolution IV fractional factorial design for the microarchitecture of this work has a minimum size of 64.

2.4 Floorplanning

The factor and interaction effects obtained from the approach described in the previous section are now inserted as factor weights into the floorplanner to obtain an arrangement of blocks that optimizes the throughput. Since floorplanning plays a major role in determining of the global wire lengths for a design, this is an ideal phase at which these weights may be used. The throughput-aware floorplanner must have the ability to pipeline the buses if the delays exceed the system clock cycle, which is provided as input.

Our implementation uses PARQUET [17], a simulated annealing (SA) based floorplanner available in the public domain. The advantage of this SA-based approach is that it allows easy integration of our weights into the cost function. The original cost function implemented in PARQUET is a weighted sum of area ($Area$), total wire length (TWL), and aspect ratio (AR), as shown below, where W_A, W_{WL}, and W_{AR} are user defined weights for $Area$, TWL, and AR, respectively.

$$cost = W_A \cdot Area + W_{AR} \cdot AR + W_{WL} \cdot TWL \qquad (2)$$

For throughput-aware floorplanning, we replace TWL with the new cost, weighted sum of factor latencies, $WSFL$, which also considers interactions. For each interaction, its weight is multiplied with minimum of the latencies of the involving factors, since the influence of latencies other than the minimum on the system performance must be attributed to the individual factor effects. If $\mathcal{J}$ is the set of interactions, then,

$$cost = W_A \cdot Area + W_{AR} \cdot AR + W_{WL} \cdot WSFL$$
$$WSFL = \sum_{i=1}^{19} (w_i \cdot lat(i)) + \sum_{j \in \mathcal{J}} (w_j \cdot min(j)) \qquad (3)$$

where $lat(i)$ and w_i are the latency and the weight, of factor i, respectively, w_j is the weight of interaction $j \in \mathcal{J}$, and $min(j)$ is the minimum of the latencies of its associated factors.

For the factor $extra\_fet$, the latency $lat()$ is computed as the sum of the buses fet_il1, $fet\_bpred, il1\_bpred$, as shown in Table 1, and the latency of max_lsq_ruu is the maximum of the latencies of dec_ruu and dec_lsq buses. For all other factors, which directly model the buses with the same name, the latencies of the buses are used directly.

Parameter	Value
Fetch width	4 instrs/cycle
Issue width	4 instrs/cycle
Commit width	4 instrs/cycle
RUU entries	64
LSQ entries	32
IFQ entries	8
Branch pred	bimod, 4K table 2-lev 1K table, 10-bit 4K BHT
IL1	64K, 32B, 4-way LRU, latency: 1
DL1	64K, 32B, 4-way LRU, latency: 1
L2	2M, 64B, 8-way latency: 12
ITLB, DTLB	128 entries Miss latency: 30

Table 3: Block configuration of the processor.

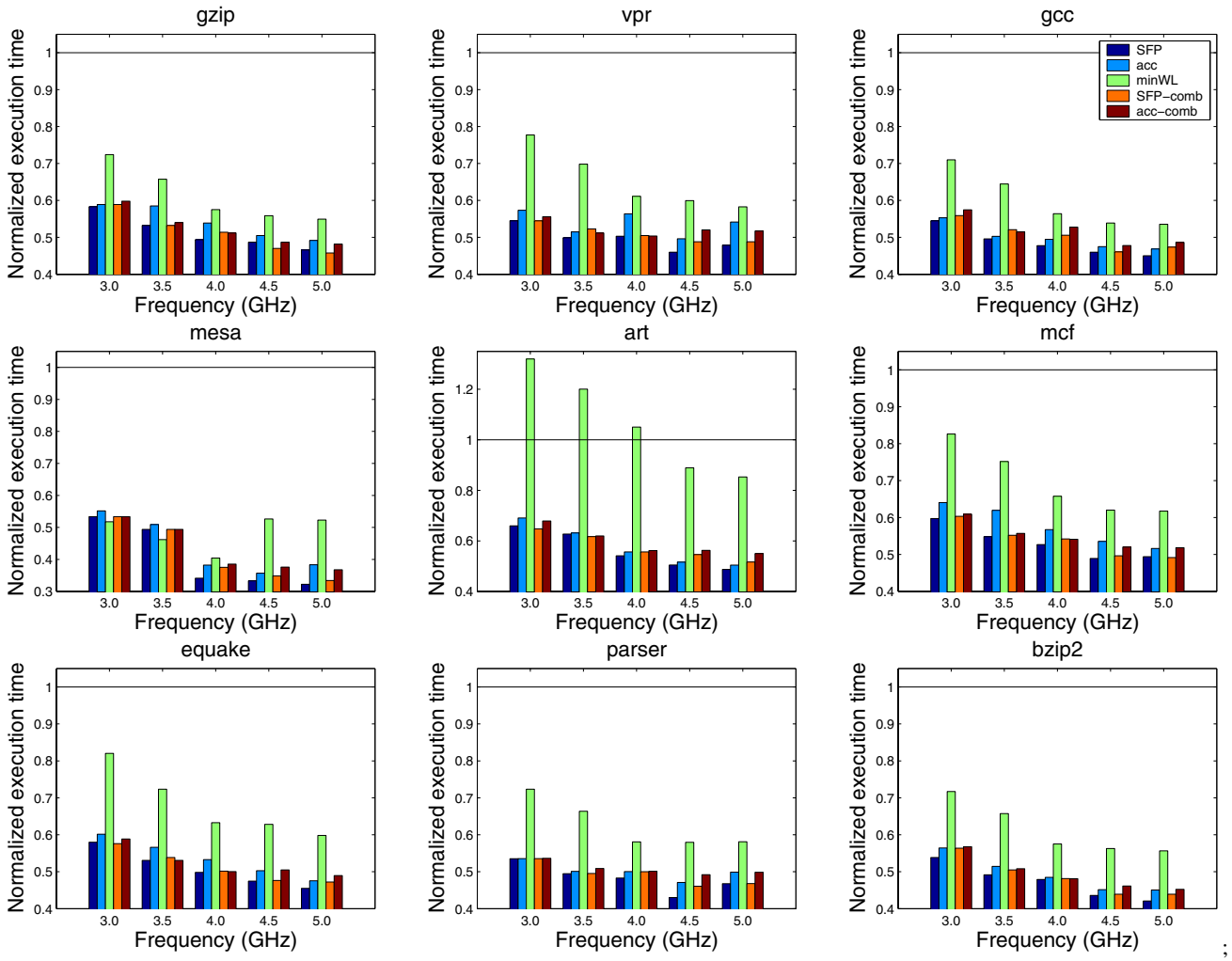

Figure 3: Results for nine benchmarks for five different frequencies. The execution times are normalized to the baseline case, where wire-pipelining is not employed and the frequency cannot exceed 1.6GHz, and this indicated by the horizontal line in each graph.

3. EXPERIMENTAL RESULTS

The configuration of the parameters and functional units for the microarchitecture is listed in Table 3 for the set of blocks shown in Figure 2. The configuration, as well as the areas of the individual blocks, are taken from [8]. The low value for a bus latency is chosen to be 0, depicting the best case placement of the connecting blocks. For the high value, we pick the corner-to-corner latency in the chip, which is found to be 10 clock cycles for a frequency of 5.0GHz in 65nm technology, based on projections from [18]. For each of the nine MinneSPEC benchmarks of Table 2, 32 cycle-accurate simulations are performed, for the resolution III design employed. Although the floorplan can be optimized for each of the individual benchmarks, in reality, one must generate a single floorplan for the processor that is, on average, optimal over all benchmarks. For this case, referred to as "comb" henceforth, the factor and interaction weights are determined by combining the weights[5] of the factors across all nine benchmarks.

The weights, both for the individual benchmarks and the combined case, thus obtained are used to guide our Statistical FloorPlanner (SFP), which uses a resolution III fractional factorial design methodology to obtain the weights. We present results for a number of clock frequencies, ranging from 3.0GHz to 5.0GHz. The weights can be used for each of these frequencies, since the bus latency ranges are valid for all of the frequencies less than or equal to 5.0GHz. We compare the results of SFP with those of traditional floorplanning, where the cost function to be minimized is the total wire length, and we refer to this floorplanner as *minWL*. In addition, we also compare our results with the access frequency-based floorplanning of [9], which will be referred to as *acc* henceforth. Based on [19], we have implemented the algorithm in [9] that gathers the bus access information by incorporating access counters for each bus in SimpleScalar.

We assume that the operating frequency of the chip is constrained only by the bus delays, and the maximum of the delays of the buses is the minimum possible clock period when wire-pipelining is not employed. The corresponding maximum frequency, obtained by minimizing the maximum of wire lengths of the global wires in the floorplanner, is determined to be about 1.6GHz, and this forms the baseline unpipelined design.

Figure 3 presents the results obtained from floorplanning for the nine MinneSPEC benchmarks. The graphs plot the execution

[5]Normalized to the maximum of the factor (and interaction) weights for that benchmark.

times[6] of the programs for five different frequencies ranging from 3.0GHz to 5.0GHz. As mentioned earlier, the baseline processor with no wire-pipelining operates at 1.6GHz. All execution times are normalized to that of this baseline processor, which is shown as a horizontal line in each of the plots. The bars "SFP-ind" and "acc-ind" represent the cases, respectively, for SFP and acc, where the floorplan is specifically optimized for a benchmark using the corresponding weights. The corresponding general cases, where the weights across all benchmarks are combined to obtain a single floorplan for the processor, are represented as "SFP-comb" and "acc-comb", respectively.

The figures indicate that, for most cases, as expected, individual weights result in floorplans with better performance than when the combined weights are used. The graphs show that "SFP-comb" outperforms "minWL" by a large margin for each benchmark over all, particularly high, frequencies. In addition, "SFP-comb" performs better than "acc-comb" for almost all frequencies. On an average, as compared to "acc-comb", "SFP-comb" reduces the execution time by 2%-4% for the nine benchmarks. It must be noted that the lower improvements (execution time reductions) are due to the contribution of the smaller frequencies such as 3GHz, where the relatively less amount of pipelining indicates less reductions in the execution times. For higher frequencies such as 5GHz, a good amount of improvements are observed. For instance, improvements of about 10% and 7% are obtained for 177.mesa and 197.parser, respectively. Besides, the results obtained from floorplanning using individual weights, i.e., the cases "SFP-ind" and "acc-ind", indicate similar improvements for SFP in the execution time. The average improvements seen are in the range 4%-6%, while improvements are on the higher side for higher frequencies (the maximum and minimum improvements of about 18% and 4% can be seen for 177.mesa and 179.art, respectively, for a frequency of 5GHz.). It can be observed that the execution times decrease as the frequency increases for all benchmarks, except for 175.vpr, where it increases slightly for the case "SFP-ind", as the frequency increases from 4.5GHz to 5GHz. In addition, the "minWL" solution for 179.art results in highly inferior layouts, with higher execution times than the baseline case for most frequencies. The reason behind this is that one of the most critical buses, ruu_fadd, is penalized (kept long) in the floorplanning solution, in the process of minimizing the total wire length.

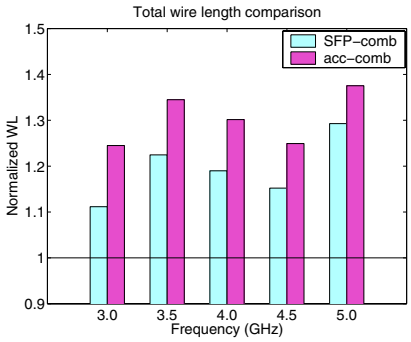

Figure 4: Total wire lengths for "SFP-comb" and "acc-comb" cases. The wire lengths are normalized to the "minWL" case, which is shown by the horizontal line.

Besides maximizing throughput, another equally important issue is the wire length associated with each design. Even if the objective at the floorplanning level is to maximize performance, higher wire lengths may not be desirable as they can lead to problems such as congestion and detours in the later stages of physical design. Since both SFP and acc minimize the weighted sum of bus latencies, it is possible that the floorplanning results in layouts where the lengths of some low ranked wires, because of their low weights, are significantly increased. Figure 4 shows the total wire length trends, normalized to the "minWL" case, for the "SFP-comb" and "acc-comb" approaches. For all frequencies, "SFP-comb" results in lower total wire lengths as compared to "acc-comb". On an average, "SFP-comb" results in layouts with about 10% less wire lengths than "acc-comb". We believe that the reason for this is that the magnitudes of the low ranked factors in the access ratios based approach are far less than those of the SFP approach, forcing the floorplanner to treat those wires as free variables.

[6]Given as T_{exec} in (1). The term N_{cycles} is the reciprocal of throughput that is optimized by the floorplanner, and T_{clk} is the corresponding frequency.

4. CONCLUSION AND FUTURE WORK

This paper proposed a methodology to based on a statistical design of experiments approach to identify the performance critical buses in a microarchitecture. The performance impact of each bus and is quantified by assigning weights, and the approach is applied at the floorplanning level. A comparison of the results with an existing approach, which uses bus access frequencies as weights, indicates that our proposed methodology produces better performance with a lower total wire length. As future work, we intend to further optimize the approach, and incorporate other critical objectives such as power consumption.

5. REFERENCES

[1] S. Borkar, "Obeying Moore's law beyond 0.18 micron," in *Proc. IEEE ASIC/SOC*, pp. 26–31, Sep. 2000.

[2] P. Cocchini, "Concurrent flip-flop and repeater insertion for high performance integrated circuits," in *Proc. IEEE/ACM ICCAD*, pp. 268–273, Nov. 2002.

[3] S. Hassoun *et al.*, "Optimal buffered routing path constructions for single and multiple clock domain systems," in *Proc. IEEE/ACM ICCAD*, pp. 247–253, Nov. 2002.

[4] V. Nookala and S. S. Sapatnekar, "Correcting the functionality of a wire-pipelined circuit," in *Proc. ACM/IEEE DAC*, pp. 570–575, Jun. 2004.

[5] L. Scheffer, "Methodologies and tools for pipelined on-chip interconnect," in *Proc. IEEE ICCD*, pp. 152–157, Oct. 2002.

[6] D. C. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0," Technical Report CS-TR-97-1342, The University of Wisconsin, Madison, Jun. 1997.

[7] J. L. Henning, "SPEC CPU 2000: Measuring CPU performance in the new millennium," *IEEE Computers*, vol. 33, pp. 28–55, Jul. 2000.

[8] C. Long *et al.*, "Floorplanning optimization with trajectory piecewise-linear model for pipelined interconnects," in *Proc. ACM/IEEE DAC*, pp. 640–645, Jun. 2004.

[9] M. Ekpanyapong *et al.*, "Profile-guided microarchitectural floorplanning for deep submicron processor design," in *Proc. ACM/IEEE DAC*, pp. 634–639, Jun. 2004.

[10] A. Jagannathan *et al.*, "Microarchitecture evaluation with floorplanning and interconnect pipelining," in *Proc. ACM/IEEE ASPDAC*, pp. 32–35, Jan. 2005.

[11] D. C. Montgomery, *Design and analysis of experiments*. New York, NY: John Wiley, 1991.

[12] A. J. KleinOsowski and D. J. Lilja, "MinneSPEC: A new SPEC benchmark workload for simulation-based computer architecture research," *IEEE Computer Architecture Letters*, vol. 1, Jun. 2002.

[13] J. Cong *et al.*, "Microarchitecture evaluation with physical planning," in *Proc. ACM/IEEE DAC*, pp. 32–35, Jun. 2003.

[14] J. Yi *et al.*, "A statistically rigorous approach for improving simulation methodology," in *Proc. ACM HPCA*, pp. 281–291, Feb. 2003.

[15] R. Plackett and J. Burman, "The design of optimum multifactorial experiments," *Biometrika*, vol. 33, pp. 305–325, Jun. 1956.

[16] C. F. J. Wu and M. Hamada, *Experiments: Planning, analysis, and parameter design optimization*. New York, NY: John Wiley, 2000.

[17] S. N. Adya and I. L. Markov, "Fixed-outline floorplanning through better local search," in *Proc. IEEE ICCD*, pp. 228–334, Oct. 2001.

[18] J. Cong, "An interconnect-centric design flow for nanometer technologies," *Proc. IEEE*, vol. 89, pp. 505–528, Apr. 2001.

[19] M. Ekpanyapong. Private communication, 2004.