LIST OF FIGURES

1.1. Execution model of a speculative multithreaded processor with helper threads. 12
1.2. Execution model of a speculative multithreaded processor with speculative multithreading. 13
2.1. Effective instruction window managed by speculative multithreaded processors. 26
2.2. Centralized Speculative Multithreaded Processor. 28
2.3. Clustered Speculative Multithreaded processor with three thread units. 31
2.4. A clustered speculative multithreaded processor with four thread units fully interconnected. 32
2.5. The thread order predictor. 35
2.6. Prediction accuracy of the thread order predictor. 36
2.7. Branch prediction accuracy. 37
3.1. Average number of input and live-in input values (through register and memory) for the loop-iteration spawning scheme. 46
3.2. Hit ratio of the loop iteration table with 8 entries. 52
3.3. The multi-value cache for a SM processor with four thread units. 62
3.4. Speed-up over single-threaded execution of a clustered speculative multithreaded processor with 16 thread units and perfect synchronization mechanism. 67
3.5. Performance potential of a speculative multithreaded processor with perfect value prediction for register and memory values and both thread ordering schemes for a) 4 thread units and b) 16 thread units. 69
3.6. Stride and increment predictors: a) 4 consecutive traces (TA, TA, TB and TA) which reference register Ri; b) predicted value of Ri4 using a stride predictor; c) predicted value of Ri4 using the increment predictor. .................................................. 72

3.7. Predicting register values of loop traces using PC-indexed predictors: a) trace input values; b) trace output values. ................................................................. 75

3.8. Predicting distance-3 values of loop traces using PC-indexed predictors: a) input values; b) output values. ................................................................. 76

3.9. Average number of inputs/outputs and distance-3 inputs/outputs per trace. ............... 76

3.10. Predicting distance-3 values of loop iterations using trace-based indexing: a) input values; b) output values. ................................................................. 77

3.11. Percentage of traces that have all their distance-3 output values correctly predicted. .... 78

3.12. Control-flow misprediction for the Path-based and the ideal gshare for loop iterations. . . 79

3.13. Speed-up for the different value predictors and for the loop-iteration spawning policy. . 82

4.1. Average number of iterations per loop execution. ................................................. 94

4.2. Average number of consecutive iterations that follow the same control-flow. ............. 96

4.3. Average number of different control flows in the last 8 iterations of innermost loops. .... 96

4.4. Percentage of code executed in parallel with other threads for each spawning policy. . . 98

4.5. Speed-ups for the three different spawning polices a) loop-iteration, b) loop-continuation and c) subroutine-continuation for the unrestricted thread ordering scheme. ......................... 99

4.6. Speed-up of the combination of heuristics compared with a single-threaded execution. . . 101

4.7. Average number of active threads per cycle. .................................................... 101

4.8. Percentage of parallelized code. ................................................................. 102

4.9. Steps of the profile-based spawning scheme. ................................................. 103
4.10. Number of pairs of basic blocks selected and number of selected pairs that have different spawning points. ................................................................. 105

4.11. Computed and Real reaching probability submatrix for a subroutine invoked from more than one place in the code. ................................................................. 107

4.12. Speed-up over a single-threaded execution obtained for 16 TU. ......................... 107

4.13. Average number of active threads per cycle. ................................................. 108

4.14. Percentage of code that is executed in parallel with some other code. ............... 109

4.15. Thread Unit Utilization for the Profile-based spawning scheme with the call-return pairs. 110

4.16. Percentage of time the spawning pair <9360-9361> of the go benchmark is executed simultaneously with another threads. ...................................................... 111

4.17. Speed-ups achieved by the different spawning pair removal scheme for different number of cycles executing alone. ......................................................... 112

4.18. Average number of spawning pairs removed by the cancellation policy. .............. 113

4.19. a) Speed-ups achieved by the different spawning pair removal scheme for different number of occurrences before cancelling for the 50-cycle removal scheme.
    b) Percentage of cancelled spawning pairs for the cancellation scheme after 8 and 16 occurrences. ................................................................. 114

4.20. Speed-up for the cancellation policies that remove spawning pairs are executing together with 2 or less parallel threads. ......................................................... 114

4.21. Speed-up of the cancellation policy that reconsiders an eliminated spawning pair after visiting it 8 times. ................................................................. 115

4.22. Thread Unit Utilization for the Profile-based spawning scheme with the call-return pairs for the best cancellation policy (50 cycles executing alone). .................. 116
4.23. Thread unit utilization for compress when the cancellation policy is applied after 200 cycles of execution alone. ................................................................. 117

4.24. Percentage of spawning pairs that create speculative threads (not taking into account the call-return pairs). ................................................................. 117

4.25. Speed-up for the reassign policy. ......................................................... 118

4.26. Speed-up of the reassign spawning policy compared with the 50-cycle removal policy (for compress, 200 cycles). ......................................................... 119

4.27. Average number of instructions between the spawning and the control quasi-independent point statically compared with the number of dynamic instructions executed at each thread unit. ................................................................. 120

4.28. Example that justifies that thread sizes are lower than the expected. .......... 120

4.29. Speed-up achieved when a minimum thread size is considered to spawn new speculative threads................................................................. 121

4.30. Thread size for the conventional removal policy and for the minimum thread size spawning policy. ................................................................. 122

4.31. Speed-up of the profile-based spawning policy over the combination of heuristics. .... 123

5.1. Branch prediction accuracy. ................................................................. 131

5.2. Slow-down when independent local branch predictors are used. .................. 131

5.3. Value prediction accuracy. ................................................................. 132

5.4. Speed-up with a perfect and a realistic value predictor. .......................... 133

5.5. Value prediction accuracy for the independent and the predictable profile-based spawning policies. ................................................................. 134

5.6. Speed-up obtained by the independent and the predictable profile-based spawning scheme with perfect and a realistic value predictor. .................. 135
5.7. Slow-down for an 8-cycle initialization overhead. ......................... 136

5.8. Average speed-ups for a 4-Thread Unit clustered processor. ................. 137