AFaultTolerantApproachtoMicroprocessorDesign

ChrisWeaver
ToddAustin
AdvancedComputerArchitectureLaboratory
UniversityofMichigan
{chriswea,taustin}@eecs.umich.edu

Abstract

Weproposeafault-tolerantapproachtoreliablemicroprocessordesign.Ourapproach,basedontheuseofanon-line checkercomponentintheprocessorpipeline,providessignificantresistancetocoreprocessordesignerrorsandoperationalfaultssuchassupplyvoltagenoiseandenergetic particlestrikes.Weshowthroughcycle-accuratesimulation andtiminganalysisofaphysicalcheckerdesignthatour approachpreservessystemperformancewhilekeepingarea overheadsandpowerdemandslow.Furthermore,analyses suggestthatthecheckerisafairlysimplestatemachinethat canbeformallyverified,scaledinperformance,andreused. Furthersimulationanalysesshowvirtuallynoperformance impactswhenoursimplecheckerdesigniscoupledwitha high-performancemicroprocessormodel.Timinganalyses indicatethatafullysynthesizedunpipelined4-widechecker componentin0.25umtechnologyiscapableofcheckingAlpha instructionsat288MHz.Physicalanalysesalsoconfirmthat costsarequitemodest;ourprototypecheckerrequiresless than6%theareaand1.5%thepowerofanAlpha21264processorinthesametechnology.Additionalimprovementstothe checkercomponentaredescribedwhichallowforimproved detectionofdesign,fabricationandoperationalfaults.

1Introduction

High-qualityverificationandtestingisavitalstepinthe designofasuccessfulmicroprocessorproduct.Designersmust verifythecorrectnessoflargecomplexsystemsandensurethat manufacturedpartsworkreliablyinvaried(andoccasionally adverse)operatingconditions.Ifsuccessful,userswilltrust thatwhentheprocessorisputtoataskitwillrendercorrect results.Ifunsuccessful,thedesigncanfalter,oftenresultingin seriousrepercussionsrangingfrombadpress,tofinancial damage,tolossoflife.Therehavebeenanumberofhigh-profileexamplesoffaultymicroprocessordesigns.Perhapsthe mostpublicizedcasewastheIntelPentiumFDIVbugwhere aninfrequentlyoccurringfunctionaldesignerrorcausederroneousresultsinsomefloatingpointdivides[1].Functional designerrorsarenottheonlysourceofproblems;theMIPS R10000microprocessorwasrecalledduetomanufacturing problemsthatresultedinunreliableoperation[2].

1.1ReliabilityChallenges

Thechallengesthatmustbeovercometobuildareliable microprocessordesignaregreat.Therearemanysourcesof errors,eachrequiringcarefulattentionduringdesign,verification,andmanufacturing.Webroadlyclassifythefaultsthatcan reducereliabilityintothreecategories:designfaults,manufacturingfaults,andoperationalfaults.

1.1.1Designfaults

Designfaults aretheresultofhumanerror,eitherinthe designorspecificationofasystemcomponent,thatrendersthe partunabletocorrectlyrespondtocertaininputs.Thetypical approachusedtodetectthesebugsissimulation-basedverification.Amodeloftheprocessorbeingdesignedexecutesa seriesoftestsandcomparesthemodel'sresultstoexpected results.Unfortunately,designerrorssometimesslipthrough thistestingprocessduetotheimmensesizeofthetestspace. Tominimizetheprobabilityofundetectederrors,designers employvarioustechniquestoimprovethequalityofverificationincludingco-simulation[4],coverageanalysis,random testgeneration[5],andmodel-driventestgeneration[6].

Anotherpopulartechnique, *formalverification* ,usesequalitycheckingtocompareadesignundertestwiththespecificationofthedesign.Theadvantageofthismethodisthatit worksatahigherlevelofabstraction,andthuscanbeusedto checkadesignwithoutexhaustivesimulation.Thedrawback tothisapproachisthatthedesignandtheinstructionsetarchitectureitimplementsneedtobeformallyspecifiedbeforethe processcanbeautomated.

Complexmoderndesignshaveoutpacedthecapabilitiesof currentverificationtechniques.Forexample,amicroprocessor with3232-bitregisters,8k-byteinstructionanddatacaches, and300pinscannotbefullyexaminedwithsimulation-based testing.Thedesignhasatestspacewithatleast 2^{132396} startingstatesandupto 2^{300} transitionedgesemanatingfromeach state.Whileformalverificationhasimproveddetectionof designfaults,fullformalverificationisnotpossibleforcomplexdynamicallyscheduledmicroprocessordesigns[7,8,9]. Todate,theapproachhasonlybeendemonstratedforin-order issuepipelinesorsimpleout-of-orderpipelineswithsmall windowsizes.Completeformalverificationofcomplexmodernmicroprocessorswithout-of-orderissue,speculation,and largeinstructionwindowsiscurrentlyanintractableproblem [10,11].

1.1.2Manufacturingdefects

Manufacturingdefects arisefromarangeofprocessing problemsthatmanifestduringfabrication.Forexample,step coverageproblemsthatoccurduringthemetallizationprocess maycauseopencircuits,orimproperdopinginthechannelof CMOStransistorsmaycauseachangeinthethresholdvoltage andtimingofadevice. *Nonconcurrenttesting* techniques, whichplacethepartintoaspecialtestingmode,aretheprimaryvehiclefordiagnosingthesetypeoferrors.Testingofthe systemisaccomplishedbyaddingspecialtesthardware.

ScantestingaddsaMUXtotheinputsofflip-flopsthat allowforreadingandwritingoflatchesduringtestmode.This methodprovidesdirectcheckingofflipflopoperationandindi-

rectcheckingofcombinationlogicconnectedtothescanlatch. Usingthescanchain,testvectorsareloadedintotheflipflops andthencombinationallogicisexercisedtodetermineifthe implementationisfaulty.Builtinselftest(BIST)addsspecializedtestgenerationhardwaretoreducethetimeittakestoload latcheswithtestvectors.BISTtestgeneratorstypicallyemploy modifiedlinearshiftfeedbackregister(LSFR)orROMsto generatekeytestvectorsthatcanquicklytestforinternallogic defectssuchassingle-stucklinefaults[12].Toobtainsufficientdesigncoverageincurrentdesigns,BISTcantakeas muchas15%ofthedesignarea[14,15,16].Amoreglobal approachistakenby *IDDQtesting*,whichusesonboardcurrentmonitoringtodetectifthereareanyshort-circuits.During testing,powersupplycurrentsaremonitoredwhilethesystem isexercised;anyabnormallyhighcurrentspikesareindicative ofshort-circuitdefects[13].TheadvantageofIddqisthatitis straightforwardtotestalargeareaallatonce,however,it requirescarefulregulationtogetthecorrectcurrentlimits.

1.1.3Operationalfaults

Operationalfaults arecharacterizedassensitivityofthe chiptoenvironmentalconditions.Itisusefultosubdivide thesetypeoferrorsintocategoriesbasedontheirfrequency [17]:permanent,intermittent,andtransientfaults. *Permanent faults*occurconsistentlybecausethechiphasexperiencedan internalfailure.Electrometalmigration[18]andhotelectrons [19][20]aretwoexamplesofpermanentfaultsthatcanrender adesignirrevocablydamaged.Wealsoclassifylatch-up, whichiscausedbyaunitygaininthebipolartransistorstructurespresentinaCMOSlayout,asapermanentfault;however,thisfaultcanbeclearedbypoweringdownthesystem.

Unlikepermanentfaults, *intermittentfaults* donotappear continuously.Theyappearanddisappear,buttheirmanifestationishighlycorrelatedwithstressfuloperatingconditions. Examplesofthistypeoffaultincludepowersupplyvoltage noise[21]ortimingfaultsduetoinadequatecooling[19]. Data-dependentdesignerrorsalsofallintothiscategory.These implementationerrorsareperhapsthemostdifficulttofind becausetheyrequirespecificdirectedtestingtolocate.

Transientfaults appearsporadicallybutcannotbeeasily correlatedtoanyspecificoperatingcondition.Theprimary sourceofthesefaultsaresingleeventradiation(SER)upsets. SERfaultsaretheresultofenergizedparticlestrikesonlogic whichcandepositorremovesufficientchargetotemporarily turnthedeviceonoroff,possiblycreatingalogicerror [24][25].Whileshieldingispossible,itsphysicalconstruction andcostmakeitanunfeasiblesolutionatthistime[26].

Concurrenttesting techniquesareusuallyrequiredtodetect operationalfaults,sincetheirappearanceisnotpredictable. Threeofthemostpopularmethodsaretimers,codingtechniquesandmultipleexecution.Timersguaranteethataprocessorismakingforwardprogress,bysignalinganinterruptifthe timerexpires.Codingtechniques,suchasparityorECC,use extrainformationtodetectfaultsindata.Whileprimarilyused toprotectstorage,codingtechniquesalsoexistforlogic[27]. Finally,datacanbecheckedbyusingak-arysystemwhere extrahardwareorredundantexecutionisusedtoprovidea valueforcomparision[17,28,29,30].

Deepsubmicronfabricationtechnologies(i.e.,process technologieswithminimumfeaturesizesbelow0.25um) heightentheimportanceofoperationalfaultdetection.Finer featuresizesresultinsmallerdeviceswithlesscharge,increasingtheirexposuretonoise-relatedfaultsandSER.Ifdesigners cannotmeetthesenewreliabilitychallenges,theymaynotbe abletoenjoythecostandspeedadvantagesofthesedenser technologies.

1.2ContributionsofthisPaper

Inthispaper,wedetailanovelon-linetestingapproach, called *dynamicverification*,thataddressesmanyofthereliabilitychallengesfacingfuturemicroprocessordesigns.Our solutioninsertsanon-linecheckingmechanismintotheretirementstageofacomplexmicroprocessor.Thecheckermonitorstheresultsofallinstructionscommittedbytheprocessor. Ifnoerrorisfoundinacomputation,thecheckerallowsthe instructionresulttobecommittedtoarchitectedregisterand memorystate.Ifanyresultsarefoundtobeincorrect,the checkerfixestheerrantresultandrestartstheprocessorcore withthecorrectresult.Weshowinthispaperthatthechecker processorisquitesimple,lendingitselftohigh-qualityformal verificationandelectricallyrobustdesigns.Inaddition,we demonstratethroughsimulationandtiminganalysisofaphysicaldesignthatcheckercostsarequitelow.Theadditionofthe checkertothepipelinecausesvirtuallynoslowdownstothe coreprocessor,andareaandpoweroverheadsforacomplete checkerdesignarequitemodest.

Thesimplecheckerprovidessignificantresistanceto designandoperationalfaults,andprovidesaconvenientmechanismforefficientandinexpensivedetectionofmanufacturing faults.Designverificationissimplifiedbecausethechecker concentratesverificationontoitself.Specifically,ifanydesign errorsremaininthecoreprocessor,theywillbecorrected (albeitinefficiently)bythecheckerprocessor.Significant resistancetooperationalfaultsisalsoprovided.Weintroduce inthispaperalow-costandhigh-coveragetechniquefor detectingandcorrectingSER-relatedfaults.Ourapproachuses thecheckerprocessortodetectenergeticparticlestrikesinthe coreprocessor;forthecheckerprocessorwe'vedevelopeda re-execute-on-errortechniquethatallowsthecheckertocheck itself.Finally,wedemonstratehowthecheckercanbeusedto implementalow-costhierarchicalapproachtomanufacture testing.SimplelowcostBISTteststhecheckermodule,then thecheckercanbeusedasthetestertotestthecoreprocessor formanufacturingerrors.Theapproachcouldsignificantly reducethecostoflate-stagetestingofmicroprocessors,while atthesametimereducingthetimeittakestotestparts.

InSection2,wedetaildynamicverificationincludingthe architectureofthecheckerprocessoranditsoperation.Section 3givesresultsofadetailedanalysisoffourprototypechecker processordesignfortheAlphainstructionset.Theseanalyses includeperformanceimpactsonthecoreprocessor,andarea andpoweroverheadsduetotheadditionofthecheckerprocessor.Section4introducessimpleextensionstoourprototype designthatprovideadditionalresistancetooperationalerrors andimprovedmanufacturingtestcapabilities.Section5details relatedwork,andSection6givesconclusionsandsuggests futuredirections.

2DynamicVerification

2.1SystemArchitecture

Recentlyweproposedtheuseofdynamicverificationto reducetheburdenofverificationincomplexmicroprocessor designs[31][32][33]. *Dynamicverification* isanon-line instructioncheckingtechniquethatstemsfromthesimple observationthat *speculativeexecutionisfaulttolerant*.Considerforexample,abranchpredictorthatcontainsadesign error, *e.g.*,thepredictorarrayisindexedwiththemostsignificantbitsofthePC(insteadoftheleastsignificantPCbits). Theresultingdesign,eventhoughthebranchpredictorcontainedadesignerror,wouldoperatecorrectly.Theonlyeffect onthesystemwouldbesignificantlyreducedbranchpredictor

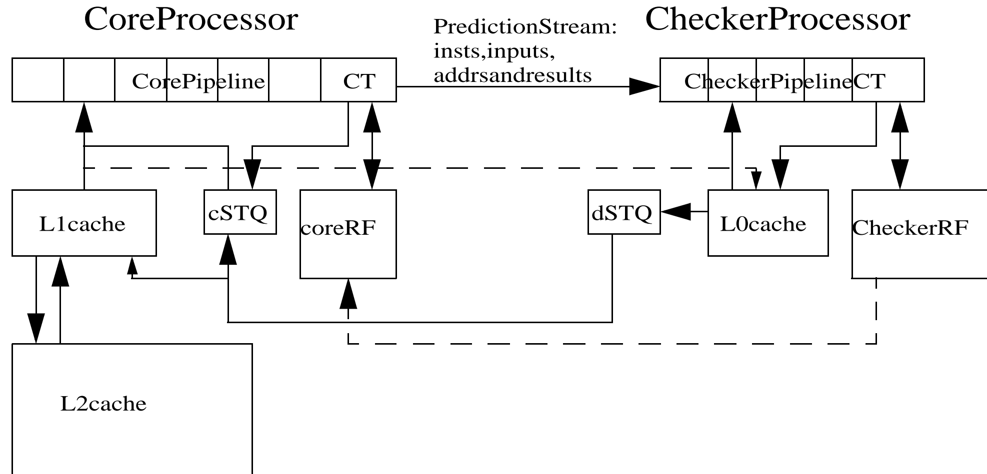

Figure 1: Dynamic Verification System Architecture.

accuracy (many more branch mispredictions) and accordingly reduced system performance. From the point of view of a correctly designed branch predictor check mechanism, a bad prediction from a broken predictor is indistinguishable from a bad prediction from a correct predictor design. Moreover, predictors are not only tolerant of permanent errors (*e.g.*, design errors), but also manufacturing defects and operational errors (*e.g.*, noise-related faults or natural radiation particle strikes).

Given this observation, the burden of verification in a complex design can be decreased by simply increasing the degree of speculation. Dynamic verification does this by pushing speculation into all aspects of core program execution, making the architecture fully speculative. In a fully speculative architecture, all processor communication, computation, control and forward progress is speculative. Accordingly, any permanent (*e.g.*, design error, defect, or failure) and transient (*e.g.*, noise-related) faults in this speculation do not impact correctness of the program. Figure 1 illustrates the approach.

To implement dynamic verification, a microprocessor is constructed using two heterogeneous internal processors that execute the same program. The *core processor* is responsible for pre-executing the program to create the *prediction stream*. The prediction stream consists of all executed instructions (delivered in program order) with their input values and any memory addresses referenced. In a baseline design, the core processor is identical in every way to the traditional complex microprocessor core up to (but not including) the retirement stage. In this baseline design, the complex core processor is ''predicting'' values because it may contain latent bugs that could render these values incorrect.

The *checker processor* follows the core processor, verifying the activities of the core processor by re-executing all program computation in its wake. The high-quality stream of predictions from the core processor serves to simplify the design of the checker processor and speed its processing. Pre-execution of the program on the complex core processor eliminates all the processing hazards (*e.g.*, branch mispredictions, cache misses, and data dependencies) that slow simple processors and necessitate complex microarchitectures. Thus is possible to build an in order checker without speculation that can match the retirement bandwidth of the core. In the event the core produces a bad prediction value (*e.g.*, due to a design errors), the checker processor will fix the errant value and flush all internal state from the core processor, and restart it after the errant instruction. Once restarted, the core processor will resynchronize with the correct state of the machine as it reads register and memory values from non-speculative storage.

To eliminate the possibility of storage structural hazards, the checker processor has its own register file and instruction and data caches. A small dedicated data cache for the checker processor, called the *L0 cache,* is loaded with whatever data is touched by the core processor; it taps off the output port of the L1 cache. This prefetching technique greatly reduces the number of misses experienced by the checker. However, if the checker processor misses in the L0 cache, it blocks the entire checker pipeline, and the miss is serviced by the core L2 cache. Cache misses are rare for the checker processor even for very small caches, because the high-quality address stream from the core processor allows it to manage these resources very efficiently. Store Queues are also added to both the core and checker (cSTQ and dSTQ in Figure 1) to increase performance.

The resulting dynamic verification architecture should benefit from a reduced burden of verification, as only the checker needs to be completely correct. Since the checker processor will fix any errors in the instructions that are to be committed, the verification of the core is reduced to simply the process of locating and fixing *commonly occurring* design errors that could affect system performance. Since the complex core constitutes a major testing problem, relaxing the burden of correctness of this part of the design can yield large verification time savings. To maintain a high quality checker processor, we leverage formal verification to ensure correct function and extensive checker processor Built-In-Self-Test (BIST) to guarantee a successful implementation.

2.2 Checker Processor Architecture

For dynamic verification to be viable, the checker processor must be simple and fast. It must be simple enough to reduce the overall design verification burden, and fast enough to not slow the core processor. A single-issue two-stage checker processor is illustrated in Figure 2a). The design presented assumes a single-wide checker, but scaling to wider or deeper designs is a fairly straightforward task (discussed later).

In the normal operation (as shown in Figure 2b), the core processor sends instructions (with predictions) at retirement to the checker pipeline. These predictions include the next PC, instruction, instruction inputs, and addresses referenced (for

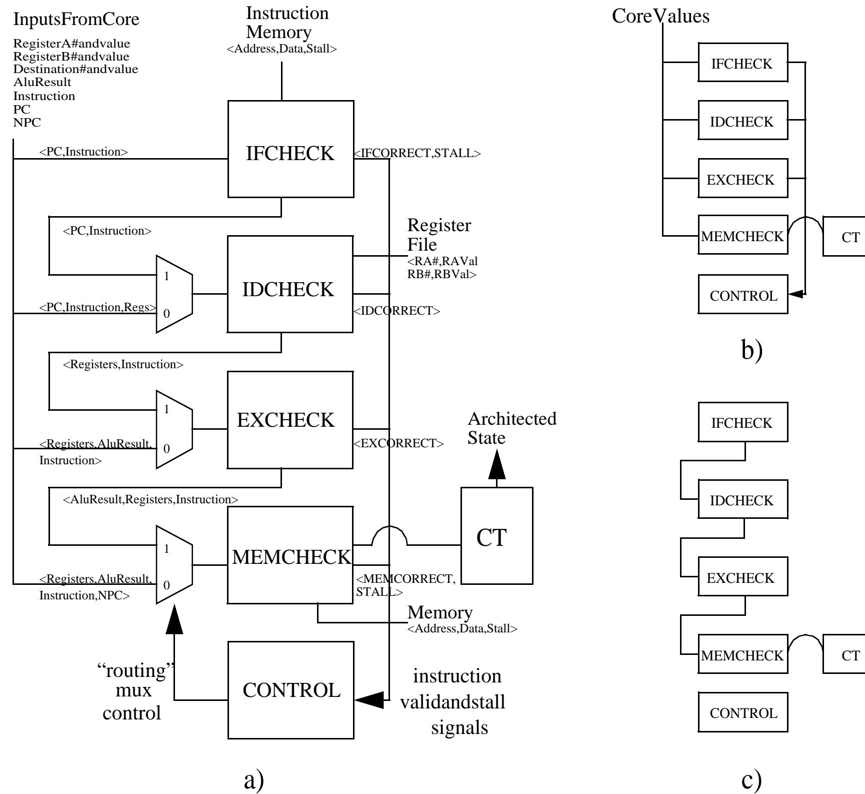

Figure2:CheckerProcessorPipelineStructurefora)asinglewidecheckerprocessor,b)acheckerprocessorin Checkmode,andc)acheckerprocessorinExecutemode

loadsandstores).Thecheckerprocessorensuresthecorrect-nessofeachcomponentofthistransferbyusingfourparallel stages,eachofwhichverifiesaseparatecomponentofthepre-dictionstream.Theseparationshownisdoneinamannersim-ilartotheclassicHennessyandPattersonfivestage pipeline[34].TheIFCHECKunitverifiestheinstructionfetch byaccessingtheinstructionmemorywiththecheckerprogram counter.IDCHECKverifiesthattheinstructionwasdecoded correctlybycheckingtheinputregistersandcontrolsignals. EXCHECKre-executesthefunctionalunitoperationtoverify corecomputation.Finally,theMEMCHECKverifiesanyloads byaccessingthecheckermemoryhierarchy.

Ifeachpredictionfromthecoreprocessoriscorrect,the resultofthecurrentinstruction(aregisterormemoryvalue)as computedbythecheckerprocessorisallowedtoretiretonon-speculativestorageinthe *commit(CT)* stageofthechecker processor.Intheeventanypredictioninformationisfoundto beincorrect,thebadpredictionisfixed,thecoreprocessoris flushed,andthecoreandcheckerprocessorpipelinesare restartedaftertheerrantinstruction.Coreflushandrestartuse theexistingbranchspeculationrecoverymechanismcontained inallmodernhigh-performancepipelines.

AsshowninFigure2b)and2c),theroutingMUXescanbe configuredtoformaparallelcheckerpipelineorarecovery pipeline,respectively.Inrecoverymodethepipelineisrecon-figuredintoaserialpipeline,verysimilartotheclassicH&P five-stagepipeline.Inthismode,stagecomputationsaresent tothenextlogicalstageinthecheckerprocessorpipeline, ratherthanusedtoverifycorepredictions.Unliketheclassic H&Pfive-stagepipeline,onlyoneinstructionisallowedto entertherecoverypipelineatatime.Assuch,therecovery pipelineconfigurationdoesnotrequirebypassdatapathsor complexschedulinglogictodetecthazards.Processingperfor-manceforasingleinstructioninrecoverymodewillbequite poor,butaslongasfaultsareinfrequenttherewillbenoper-ceivableimpactonprogramperformance[31].Oncethe instructionhasretired,thecheckerprocessorre-entersnormal processingmodeandrestartsthecoreprocessoraftertheerrant instruction.

Figure2alsoillustratesthatthecheckingandrecovery modesusethesamehardwaremodules,therebyreducingthe areacostofthechecker.Eachstageonlyrequiresintermediate pipelineinputs-whetherthesearefromthecoreprocessorpre-dictionstreamorthepreviousstageofthecheckerprocessor pipeline(inrecoverymode)isirrelevanttotheoperationofthe stage.Thisattributeservestomakethecontrolandimplemen-tationofindividualstagesquitesimple.Inrecoverymode,the checklogicissuperfluousastheinputswillalwaysbecorrect, however,noreconfigurationisrequiredtothechecklogicasit willneverdeclareafaultduringrecovery.

Pipelineschedulingistrivial,ifanycheckerpipelineis blockedforanyreason,allcheckerprocessorpipelinesare stalled.Thissimplifiescontrolofthecheckerprocessorand eliminatestheneedforinstructionbufferingorcomplexnon-blockingstorageinterfaces.Sincetherearenodependencies betweeninstructionsinnormalprocessing,checkerprocessor pipelinestallswillonlyoccurduringacachemissorstructural (resource)hazard.Thislackofdependenciesmakesitpossible toconstructthecheckercontrolasathreestage(Idle,Check andExecute)Moorestatemachine.ThepipelinesitsintheIdle stateuntilthearrivalofaretiredcoreprocessorinstruction. ThepipelinethenentersnormalCheckmodeuntilallinstruc-tionsareexhaustedoraninstructiondeclaresafault.Ifafault isdeclaredthepipelineenterstheExecutestate,reconfiguring

the pipeline to single serial instruction processing. Once the faulty instruction has completed, the pipeline returns to Idle or Check mode, depending on the availability of instructions from the core.

Certain faults, especially those affecting core processor control circuitry, can lock up the core processor or put it into a deadlock or livelock state where no instructions attempt to retire. For example, if an energetic particle strike changes an input tag in a reservation station to the result tag of the same instruction, the processor core scheduler will deadlock. To detect these faults, a *watchdog timer (WT)* is added. After each instruction commits, the watchdog timer is reset to the maximum latency for any single instruction to complete. If the timer expires, the processor core is no longer making forward progress and the core is restarted. At that time, pipeline control transitions to recovery mode where the checker processor is able to complete execution of the current instruction before restarting the core processor after the stalled instruction.

3 Performance Implications

In this section we detail the performance of a complex out of order processor that is enhanced with a checker. Cycle accurate simulation of a number of benchmarks is used to demonstrate that the performance impacts are minimal.

3.1 Experimental Framework

The simulators used in this study are derived from the SimpleScalar/Alpha 3.0 tool set [35], a suite of functional and timing simulation tools for the Alpha AXP ISA. The timing simulator executes only user-level instructions, performing a detailed timing simulation of an aggressive 4-way dynamically scheduled microprocessor with two levels of instruction and data cache memory. A 4-wide 3-stage checker with 4k data cache and 0.5K instruction cache was used for the analysis. Simulation is execution-driven, including execution down any speculative path until the detection of a fault, TLB miss, or branch misprediction. To perform our simulation experiments, we collected results for nine of the SPEC95 benchmarks [36] and six of the SPEC2000 benchmarks. Of these program nine are primarily integer codes and six use floating point extensively.

Physical designs were constructed in synthesizable Verilog code. The code was compiled and simulated using Cadence's VERILOG-XL 2.1.3. The waveform data generated from the simulation was viewed using DAI Signalscan Version 6.4s4. Once the design had been debugged it was synthesized using Synopsys Version 2000.05, using the .25um Leda library. Synopsys designs were derived using a wire load from this library as well. The design was synthesized with timing as its primary optimization objective, and area as a secondary concern. After synthesis was complete, Synopsys timing, area and power reports were generated to assess the design.

3.2 Performance Impacts

With dynamic verification, there are two ways the checker processor can slow the progress of the core processor. First, checker processor stalls delay the retirement of instructions in the core processor, forcing the core processor to hold speculative state longer, possibly creating backpressure at core processor decode. If speculative state resources in the core processor fill, the decoder will have to stall as it will not be able to allocate re-order buffer and load/store queue resources to new instructions. The checker processor only stalls when an access to its data cache misses or when there are insufficient functional unit resources. Second, fault recovery will delay core progress as the core processor must be flushed and restarted after the errant instruction.

Integer Benchmark	Slowdown (Checked Vs Base)	Floating Point Benchmark	Slowdown (Checked Vs Base)
compress95	0%	applu00	-1.2779%
crafty00	0.013561%	hydro2d	-0.46554
gap00	-0.73826%	lucas00	0%
gcc	0.015443%	mesa00	-0.09291%
go	0%	tomcatv	0.19605%
ijpeg	-0.25113%	turb3d	-3.16497%
li	0.0556%		
perl	0%		
twolf00	0.05295		
average	-.095%	average	-.800%
OVERALL AVERAGE		-0.3771%	

Table 1: Performance Results of a Processor Enhanced with a Checker.

The performance impacts of dynamic verification are summarized in Table 1. These results reflect performance impacts without any core faults incurred during the simulation. The processor model with dynamic verification exhibits modest performance gains over the baseline model without a checker processor. This is due to the additional checker store queue (cSTQ), which eliminates store writeback misses at commit that can slow the baseline architecture (without a checker processor). This effect is demonstrated in the average decoder stalls percentage which went from 10.128% in the baseline to only 7.744% in the dynamically verified model.

An analysis of fault rates in a previous work indicated that recovery from faults had no perceivable impact on performance provided the faults were less than one every 1000 cycles [31]. Considering that faults are the result of design errors or infrequent transient faults, we don't expect fault rates to approach this threshold.

To summarize the performance results from [31,32,33], the checker processor keeps up with the core processor because the core processor clears the many hazards that would otherwise slow this very simple design. Branch hazards are cleared by the high quality next PC predictions from the core processor, generated using pre-execution of the program. Data hazards from long latency instruction are eliminated by the input value predictions supplied by the core processor. With these inputs values, instruction need not wait for earlier dependent operations to finish. As such, dependent long latency operation may execute in parallel in the checker pipeline. Finally, data hazards from cache misses are virtually non-existent in the core processor since the core processor runs ahead of the checker initiating cache misses (and tolerating their latency) in advance of checker execution. The checker cache need only hold the working set of the pipeline, i.e., data elements from the time they are first touched in the core processor pipeline until the time they are checked by the checker processor.

Indeed, it is this lack of pipeline hazards that also makes the checker processor design quite scalable. A deeper (or superpipelined) checker processor will process instructions faster because it can employ a higher clock rate. The lack of dependences makes increasing the depth of the checker processor pipe stages fairly straightforward. A wider (or superscalar) checker processor design will process instruction faster by allowing more instructions to be processed in a single cycle. For the most part, widening the checker processor stages requires only replicating the resources such that more instruc-

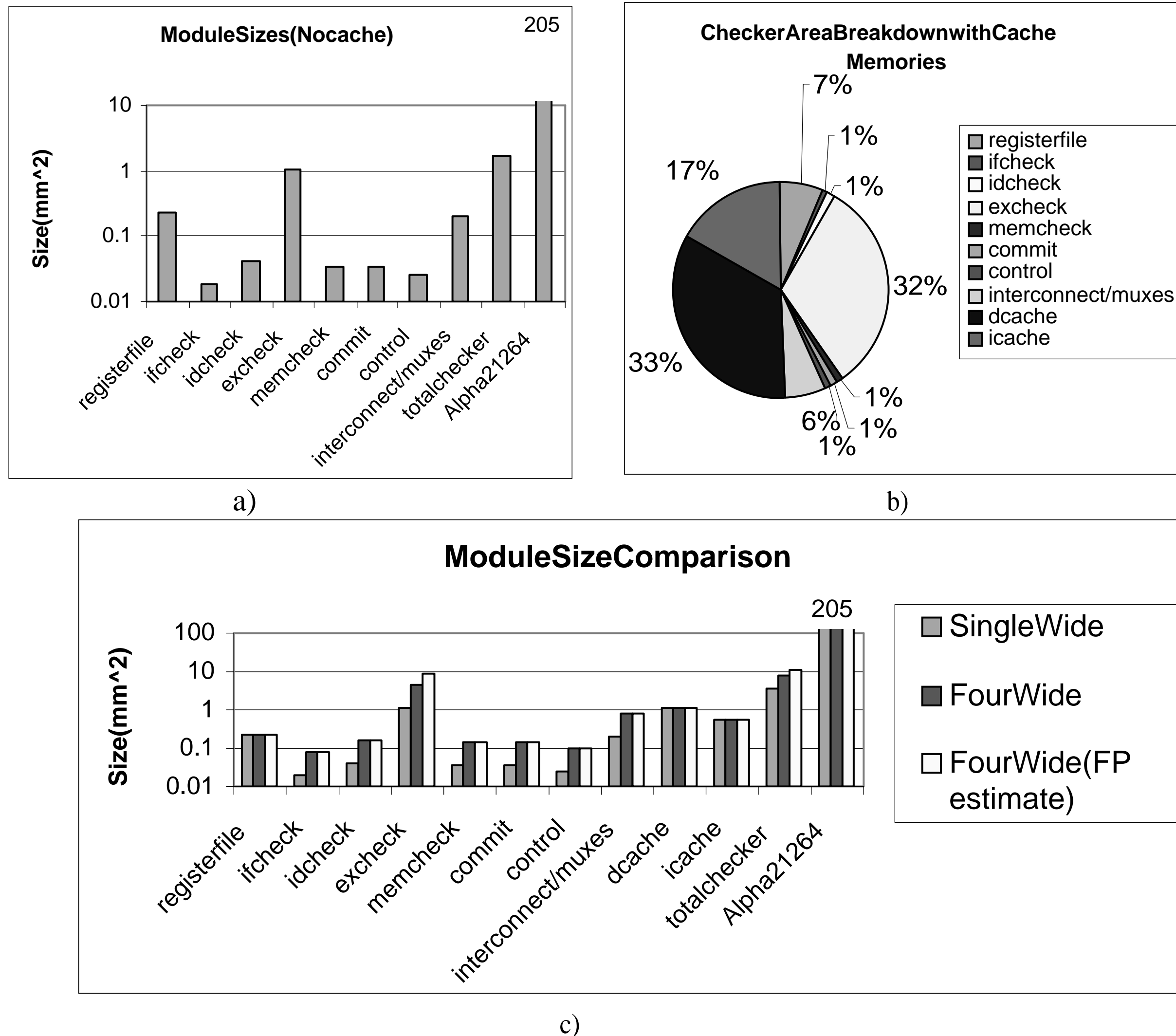

Figure4PhysicalCharacteristics:a)Theareaofthevariouscheckermodules.b)Abreakdownoftheareausedina singlewidechecker.c)Acomparisionofthesizesforasinglewidechecker,quadwidechecker,quadwidecheckwith floatingpointestimateandanAlpha21264.

tionscanbeprocessedinasinglecycle.Inputvaluechecks (forregistersormemory)aresomewhatmorecomplicated, however,asmultipleinstructionsmayattempttoreadfrom memoryorregistersinasinglecyclewhileatthesametime creatingvaluesthatmaybevisibletolaterinstructioninthe samecycle.Fortunately,thisisnomoredifficultthanthe dependencechecksthatoccurduringinstructionissue,renaming,andcommit.Wearecurrentlyinvestigatingthefull impactsofscalingonthecheckerprocessorandwillreporton thesefindingsinthefuture.

4PhysicalDesign

Whenconsideringanytypeoftestingtechniqueitisimportanttoquantifyitscostsinadditiontoitsbenefits.Thissection willanalyzetheperformance,areaandpowercostsofaprototypecheckerprocessor.

Toassesstheperformanceimpactsofacheckerprocessor, aVerilogmodelfortheintegersubsetoftheAlphainstruction setwasconstructed.Thedesignwastestedwithsmallhand codedassemblyprogramstoverifycorrectcheckeroperation. Floatingpointinstructionswerenotimplementedduetothe timeconstraints.Wedo,however,estimatefloatingpointoverheadsusingmeasurementsfromanunrelatedphysicaldesign inthesametechnology.TheSynopsystoolsetwasusedinconjunctionwitha0.25umlibrarytoproduceanunpipelinedfully synthesizeddesignthatranat288MHz.Ourhighlevelsimulationsshowthatpipeliningofthecheckerdoesnothavea severeimpactonperformance.Weareconsideringthis approachaswellassemi-customdesignsasmeanstomatching thespeedofcurrentmicroprocessors.

TheSynopsystoolsetwillgenerateareaestimatesforasynthesizeddesigngiventhecellsusedandtheestimatedinterconnect.Boundaryoptimization,whichexploitslogical redundancybetweenmodules,wasturnedoffsothatanassessmentofeachmodule'scontributiontotheoverallareacouldbe determined.A1.65mm^2single-widecheckersizewasproducedbythetoolset,whichisfairlysmallincomparisontothe 205mm^2sizeoftheAlpha21264microprocessor.Asinglewidecheckeronlyamountsto0.8%oftheareaoftheAlpha

chip.However,asmentionedbeforeonlyasubsetofthe instructionsetwasimplemented.Thecheckerareabreakdown chartshowsthatthe *excheck*module,whichhousesthefunc-tionalunits,contributesthemosttotheoverallsize.Thefloat-ingpointmodulesshouldbeslightlylargerthentheirinteger counterpartsduetotheneedtotracktheexponents.Themem-oryelementsinthecheckerdesignwereestimatedusingthe Cacti2cachecompiler[43].TheestimatesproducedbyCacti wereslightlypessimistic,butwithinafactorof2ofknown layouts.CactiprojectedthattheI-cacheandD-cachewouldbe 0.57mm^{2}and1.1408mm^{2}respectively.Thesevaluesarefor the512byteI-cacheand4KbyteD-cachethataredescribedin previoussection.Accountingforcaches,totalcheckerarea risestoroughly10mm^2,stillmuchsmallerthanacomplete microprocessorinthesametechnology.

4.1PowerAnalysis

Astransistorcountsgrow,powerisagrowingconcern sinceadequatedistributionandcoolingcancreatedesignprob-lems.Thus,checkingmechanismsmustbudgettheirpower usagetomeetoverallsystemgoals.Synopsystoolsarealso abletomodelpowerusageatahighlevel.Itestimatesswitch-ingfrequenciesforlogictodeterminethepowerconsumed.A 50% switchingestimateisthedefault.Generally,thisproduces aslightoverestimateofthepower,butitisusefulforacoarse grainanalysis.SynopsysestimatesthepowerofourVerilog designat941mW,smallincomparisontothe75wattsthatare consumedbyatypicalcoreprocessordesignliketheCompaq Alpha21264[42].Whilethisisapessimisticestimate,wehave isolatedafewmechanismstoimprovethepowerusage.To speedupthecheckerprocessorwearesendingouroperandsto allthefunctionalunitsatthesametime,thenselectingthe resultfromtheonewewant.Byincreasingthedepthofthe checkerpipeline,wecanpredecodethefunctionalunitused andonlychangetheinputstothatmodule.Thiswilldecrease theswitchinginthecheckerandtherebydecreasethepower requiredtoprocessaninstruction.

5DesignImprovementsforAdditional FaultCoverage

Thedesignpresentedthusfaristargetedatdesignfaults. Whilethecheckerinherentlyaidsinthedetectionofopera-tionalandmanufacturingfaultsduetoitscheckingcapability andsimplificationofcriticalcircuitry,ithasnotbeenthepri-maryfocusthusfar.Wewillnowexplorehowdynamicverifi-cationcanfurtheraidinthedetectionofoperationaland manufacturingfaults.

5.1OperationalErrors

Severalmechanismsreducetheprobabilityofoperational errorsaffectingfunctionality,especiallythoseerrorsthatcan arisefromSERparticlestrikes.Ifthecheckerisproperly designed,thesystemisguaranteedtofunctioncorrectlygiven anyerrorinthecore.Thisguaranteecanbesafelymade becausethecheckerwilldetectandcorrectanyoperational faultsinthecore.Ensuringcorrectoperationismorechalleng-ingwhenweconsiderthepossibilityofoperationalerrorsin thecheckercircuits.Ideally,thecheckerwouldbedesigned suchthatitslifetimeandradiationtoleranceishigherthenthe core.Howeverthismaybetoocostlyorimpossibledueto environmentalconditions.

	erroroccurred	**errordidnotoccur**
error detected	**Functioningok** checkerdetectserror thatoccurred	**CaseA** checkerwillre-check andrealizeinstruction iscorrect
errornot detected	**CaseB** Smallprobabilityof occurringseeequation intext	**Functioningok** coreisworkingcor-rectlyandcheckerdoes notfindanyerrors

Table2:OperationalFaultsinCheckerCircuitry.

Table2enumeratesthepossiblefaultscenariosthatmay occurduringthecheckingprocess.CaseArepresentsthefalse positivesituationwherethecheckerdetectedafaultthatdid notreallyoccur.TosolveCaseA,wehaveaddedadditional controllogicthatcausesthecheckertore-checkcorevalues whenanerrorisdetected,beforerecoverymodeisentered. Thisreducesthelikelihoodofthiscaseoccurring,atthe expenseofslightlyslowerfaultrecoveryperformance.

CaseBcanoccurinoneoftwoways;thefirstiswhenan operationalerrorcausesanequivalenterrortooccurinboth thecoreandthechecker.Thisistheproductoftwosmallprob-abilities.Givenagooddesign,thelikelihoodofthiseventis probabilisticallysmall.However,replicationofthefunctional unitsmaybeappliedtofurtherreducethisprobability.The otherpossibilityisthateitherthecomparisonlogicorthecon-trollogicsufferanerror.WecanemployTMRonthecontrol logictohelpreducetheprobabilitythatthiserroroccurs.Ina systemwithachecker,theprobabilityofafailure,shownin thefollowingequation,isalwaystheproductofatleasttwo unlikelyevents.Forsystemsthatrequireultra-highreliability thecheckercanbereplicateduntiltheprobabilityisacceptable fortheapplication.Thisisalowcostredundantexecution approachasonlythesmallcheckerneedstobereplicated.

$$P_{failure} = P_{designerrorcore} * P_{maskingstrikeinchecker} + P_{designerrorcore} * P_{strikecheckercontrol}$$

Figure5illustrateshowTMRcanbeappliedtothecontrol logicofthecheckertoprovidebetterreliabilityintheeventof anoperationalerrorinthecheckercontrol.Again,adesign analysiswasdonebysynthesizingaVerilogmodel.Theareas estimatespreviouslygivenshowthatthesimplecontrollogic ofthecheckeronlycontributesasmallportiontotheoverall checkersize.Theadditionoftwomorecontrollogicunitsand votinglogiconlyconsumesanextra0.12mm^2.TMRstillhas asinglepointoffailurewithinthevoterlogic,butthechance ofastrikeisreducedduetotheareadifferenceascomparedto thecontrolunitlogic.Additionally,thevoterlogiccanbe equippedwithtransistorsthataresizedlargeenoughtohavea hightoleranceofenvironmentalconditions.

5.2ManufacturingErrors

Thecheckerprocessorcanimprovethesuccessrateofman-ufacturingtest,becausethecheckerdesignlendsitselftothese tests,andonlythecheckermustbecompletelyfreeofdefects forcorrectprogramoperation.Thestructureofthechecker, wherevaluesarelatchedinandcomparedtohardware, enhancesthecheckerssuitabilityfornonconcurrenttesting techniques.Asimplescanchaincanbesynthesizedinlatches

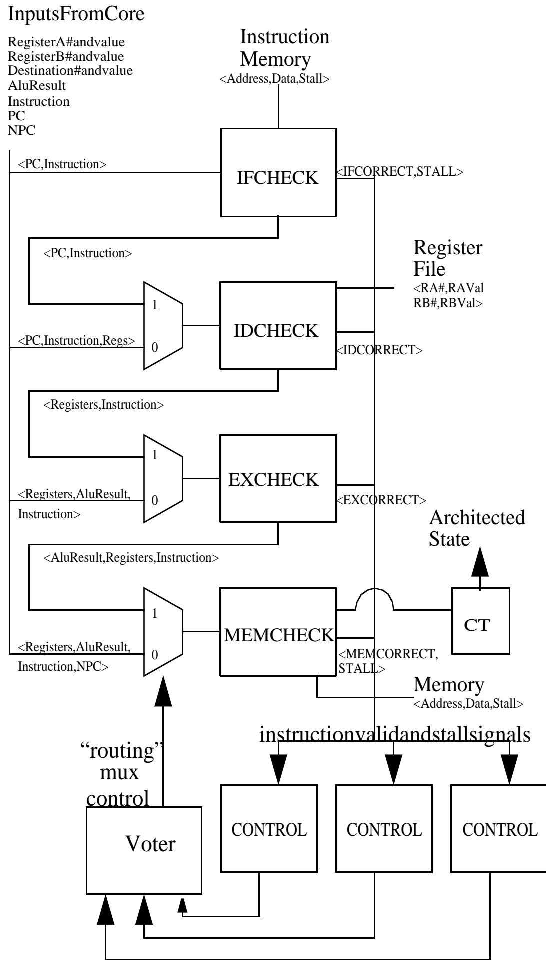

Figure5:CheckerProcessorPipelineStructurewith TMRontheControlLogic.

thatsupplythedatatothechecker.Thistypeoftestingcan achieveenhancedfaultcoverageoverthecore,sincethe checkercircuitryissignificantlysimpler.BISTisanother optionthatholdsgreatpotentialforthistypeofcircuit.

AsshowninFigure6,built-in-self-test(BIST)hardware canbeaddedtotestforcheckermanufacturingerrors.The BISThardwareusesaTestGeneratortoartificiallystimulate thecheckerlogicanditpassesvaluesintothecheckerstage latchesthatholdtheinstructionforthecheckertovalidate.The mosteffectivetestgeneratorwillbeaROMofopcodes,inputs andexpectedresults.Usingthisdataandtheerrorsignals alreadypresentinthesystem,anefficientnonconcurrenttest canbeperformed.Obviously,theROMshouldhavebothgood andbadsetsofdatasuchthatthecheckmodulesarefully tested.Thenumberoftestsrequiredtotestforallfaultsisa functionofthecheckerlogicsizeandstructure.Memories makeupanotherhugeportionofthedesign;fortunately,significantresearchhasbeendoneinhowtoeffectivelytestmemory.Marchingonesandzerosisjustonesimplewaytotesta memory.

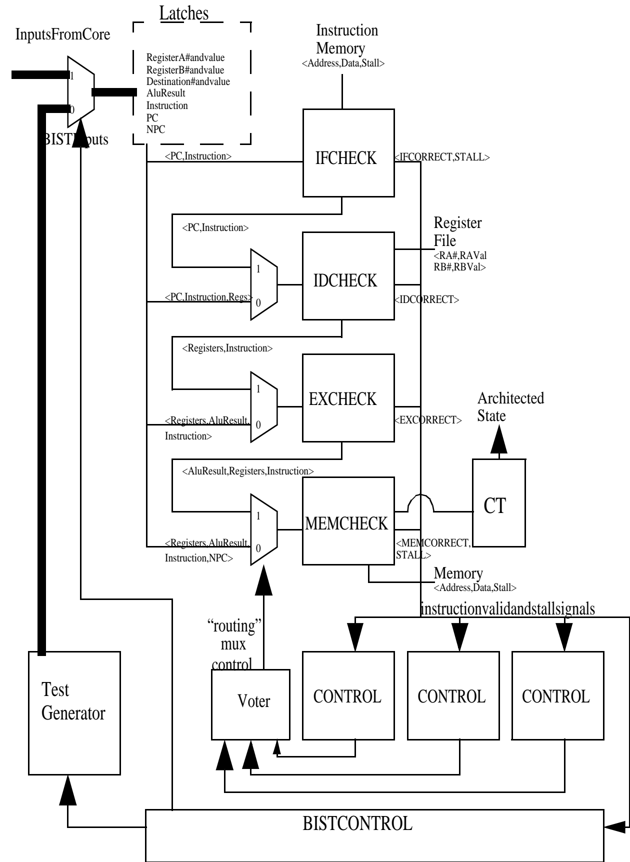

Figure6:CheckerProcessorPipelineStructurewith TMRonControlLogicandBIST.

ModuleName	#of faults	test coverage	test patterns	pattern size	memory
IFCHECK	2108	100%	204	97	2.4KB
IDCHECK	6063	100%	366	215	9.6KB
EXCHECK	82810	99.96%	764	327	30.5KB
MEMCHECK	5024	100%	290	194	6.9KB
COMMIT	388	100%	26	140	455B
CONTROL	2038	100%	80	97	790B
Total					51.8KB

Table3:BISTAnalysis.

ToassessthepotentialofapplyingBISTtothechecker,we ranoursynthesizeddesignthroughTetramax.Tetramaxisan ATPGtoolthatispartoftheSynopsyssuite.Verilogmodelsof thegatesinthesynthesizeddesignarereadintotheTetramax, allowingittoactivateandpropagatefaults.TheVerilogdesign isthenenteredintothetoolandanalyzed.Statisticsandpatternsarereportedfromtheanalysis.Inaddition,theATPGtool willtrytocompressthevectorssothatredundantpatternsare deleted.TheATPGstatisticsaresummarizedinTable3.The totalROMusageof51.8KBwaspluggedintotheCacti2tool, whichestimatedtheareaat0.057mm^2(assumingadensityof 10ROMcellsperRAMcell).

Although difficult to quantify, it is likely that the manufacturing process could benefit from the checker processor as well. In a fab where part production is limited by the bandwidth and latency of testers, the checker processor could improve testing performance. Once the checker is fully tested using internal BIST mechanisms, the checker itself can test the remaining core processor circuitry. No expensive external tester is required, only power and a simple interface with ROM to hold core testing routines and a simple I/O interface to determine if the checker passed all core processor tests.

6 Related Work

Rotenberg's AR-SMT [29] and more recently Slipstream [30] processors use multiple threads to eliminate hazards and verify correct operation. In Slipstream an advanced stream (A-stream) is used to aid a redundant stream (R-stream) by providing future knowledge to the trailing stream. As a result, performance can be increased and the redundancy can be used to detect transient faults. However, this technique does not provide total coverage or resistance to design errors. Other techniques for obtaining reliability include the redundant hardware approach, as in the IBM S/390 G5 microprocessor [38] and the redundant thread (or SRT) approach. In the SRT approach proposed in [10], redundant threads run concurrently and compare results for fault detection. Performance in SRT processors is improved with one thread prefetching cache misses and computing branch outcomes for other threads, similar to the core prefetching that the checker processor leverages.

A number of fault-tolerant processor designs have been proposed and implemented, in general, they employ redundancy to detect and/or correct transient errors. IBM's G4 processor [39] is a highly reliable processor design similar to the design in this paper in that it checks all instructions results before committing them to architected state. Checking is accomplished by fully replicating the processor core. An R-Unit is added to compare all instruction results, permitting only identical results to commit. If a failure in the processor is detected, it is addressed at the system level through on-line reconfiguration. The ERC32 is a reliable SPARC compatible processor built for space applications [40]. This design augments the microarchitecture with parity on all register and memory cells, some self-checking control logic, and control flow checking. In the event a fault is detected, a software interrupt is generated and the host program initiates recovery.

Unlike these designs, the checker can keep costs lower by only checking the function of the program computation. The G4 and ERC32 designs check both the function of the program and the mechanisms used to optimize program performance. This results in more expensive checkers, typically two times as much hardware in the core processor. Additionally, the checker can detect design errors. Simple redundant designs cannot detect design errors if the error is found in each of the redundant components.

Tamir and Tremblay [41] proposed the use of micro rollback to recover microarchitecture state in the event of a detected fault. The approach uses FIFO queues to checkpoint a few cycles of microarchitectural state. In this work, we use a similar parallel checking approach, but employ a global checking strategy to reduce checker cost. In addition, we use the existing control speculation mechanism to restore correct program state.

Blum and Wasserman discussed checkers with respect to the Pentium Division Bug [1]. In addition to describing complete checkers, they discussed how a partial checker could be utilized to achieve checking accuracy that was almost perfect. They also postulate that if error rates are kept small enough, correcting procedures can be very time consuming without impacting the performance of the system.

7 Conclusions and Future Directions

Many reliability challenges confront modern microprocessor designs. Functional design errors and electrical faults can impair the function of a part, rendering it useless. While functional and electrical verification can find most of the design errors, there are many examples of non-trivial bugs that find their way into the field. Additional faults due to manufacturing defects and operation faults such as energetic particle strikes must also be overcome. Concerns for reliability grow in deep submicron fabrication technologies due to increased design complexity, additional noise-related failure mechanisms, and increased exposure to natural radiation sources.

To counter these reliability challenges, we proposed the use of dynamic verification, a technique that adds a checker processor to the retirement phase of a processor pipeline. If an incorrect instruction is delivered by the core processor the checker processor will fix the errant computation and restart the core processor using the processor's speculation recovery mechanism. Dynamic verification focuses the verification effort into the checker processor, whose simple and flexible design lends itself to high-quality functional verification and a robust implementation.

We presented detailed analyses of a prototype checker processor design. Cycle-accurate architectural simulation of four design monitoring an Alpha 21264-like core processor resulted in virtually no performance penalities due to on-line instruction checking. The simple checker can easily keep up with the complex core processor because it uses pre-computation in the core processor to clear the branch, data, and communication hazards that could otherwise slow the simple checker pipeline.

We presented analyses of four prototype physical checker design. Timing analyses indicate a fully synthesized and unpipelined 4-wide checker processor design in 0.25um technology is capable of running at 288MHz. We are currently hand optimizing this design through better layout and stage pipelining. We fully expect our follow on efforts will demonstrate that the checker design is also quite scalable. In addition, area and power analyses of our physical design were presented. Overall, the checker processor requires less than 6% the area and 1.5% the power of an Alpha 21264, confirming that our approach is low cost. Finally, we presented novel extensions to our baseline design that improve coverage for operational faults and manufacturing fault detection.

We feel that these results strengthen the case that dynamic verification holds significant promise as a means to address the cost and quality of verification for future microprocessors. Currently, we continue to refine our physical checker processor design, optimizing its layout and investigating technique to scale its performance through pipelining. In parallel with this effort, we are also examining how we might further leverage the fault tolerance of the core processor to improve core processor performance and reduce its cost. One such approach is to leverage the fault tolerance of the core to implement self-tuning core circuitry. By employing an adaptive clocking mechanism, it becomes possible to overclock core circuitry, reclaiming design and environmental margins that nearly always exist. We will report on this and other optimizations in future reports.

8References

[1]M.BlumandH.Wasserman,“ReflectionsonthePentiumDivision Bug”,IntelCorporation,Oct.1997.

[2]M.Kane,“SGIstockfallsfollowingdowngrade,recallannouncement.”PCWeek,Sept.1996

[3]M.Williams,“FaultyTransmetaCrusoeChipsForceNECtoRecall 300Laptops.”TheWallStreetJournal,Noc30,2000

[4]P.Bose,T.Conte,andT.Austin,“Challengesinprocessormodelingandvalidation,”IEEEMicro,pp.2-7,June1999.

[5]A.Aharon,“Testprogramgenerationforfunctionalvericationof PowerPCprocessorsinIBM,”inProceedingsofthe32ndACM/IEEE DesignAutomationConference,pp.279-285,June1995.

[6]R.Grinwald,“Userdefinedcoverage,atoolsupportedmethodologyfordesignverication,”inProceedingsofthe35ndACM/IEEEDesign AutomationConference,pp.1-6,June1998.

[7]Anonymous“ScalableHybridVerificationofComplexMicroprocessors,”submittedforpublication

[8]M.K.SrivasandS.P.Miller,“FormalVerificationofanAvionics Microprocessor,”SRIInternationalComputerScienceLaboratoryTechnicalReportCSL-95-04,June1995.

[9]M.C.McFarland,“FormalVerificationofSequentialHardware:A Tutorial,”IEEETransactiononComputer-AidedDesignofIntegratedCircuitsandSystems,Vol12,NO.5.May1993.

[10]J.BurchandD.Dill,“Automaticvericationofpipelinedmicroprocessorscontrol,”ComputerAidedVerication,pp.68-80,1994.24

[11]J.Sawada,“Atablebasedapproachforpipelinedmicroprocessor verfication,”inProceedingsofthe9thInternationalConferenceonComputerAidedVerication,June1997.

[12]H.Al-AsaadandJ.P.Hayes,“Designverificationviasimulation andautomaatictestpatterngeneration.”InProceedingsoftheInternational ConferenceonComputer-AidedDesign.IEEEComputerSocietyPress, LosAlamitos,CA,174-1801995.

[13]E.Bohl,Th.Lindenkreuz,R.Stephan,“TheFail-StopController AE11,”InternationalTestConference,pp567-577,1997.

[14]B.T.MurrayandJ.P.Hayes,“TestingICs:GettingtotheCoreof theProblem,”Computer,Vol.29,No.11,pp.32-45,Nov.1996.

[15]M.Nicolaidis,“TheoryofTransparentBISTforRAMs,”IEEE Trans.Computers,Vol.45,No.10,pp.1141-1156,Oct.1996.

[16]M.Nicolaidis,“EfficientUBISTImplementationforMicroprocessorSequencingParts,”J.ElectronicTesting:TheoryandApplications, Vol.6,No.3,pp.295-312,June1995.

[17]H.Al-Asaad,B.T.MurrayandJ.P.Hayes,“OnlineBISTfor EmbeddedSystems,”IEEEDesignandTestofComputers,pp.17-24, October-December1998.

[18]D.G.PierceandP.G.,“Electromigration:Areview,”MicroelectronicReliability,Vol.37.No.7,pp.1053-1072,1997.

[19]R.S.MullerandT.L.Kamins,“DeviceElectronicsforIntegrated CircuitsSecondEdition,”JohnWiley&Sons,NewYork,NY1986.

[20]S.Wolf,“SiliconProcessingfortheVLSIEraVolume3-SubmicronMOSFET”LatticePress,SunsetBeach,CA1995.

[21]K.Seshan,T.Maloney,andK.Wu,“Thequalityandreliabilityof Intel'squartermicronprocess.”IntelTechnologyJournal,Sept.1998.

[22]M.Bohr,“Interconnectscaling:thereallimitertohigh-performanceULSI,”inProceedingsoftheInternationalElectronDevicesMeeting,pp.241-244,Dec.1995.

[23]N.WesteandK.Eshragian,PrinciplesofCmosVLSIDesign:A SystemsPerspective.Addison-WesleyPublishingCo.,1982.

[24]J.Z.etal,“IBMexperimentsinsoftfailsincomputerelectronics,”IBMJournalofResearchandDevelopment,vol.40,pp.3-18,Jan. 1996.

[25]P.Rubinfeld,“Managingproblemsathighspeed,”IEEEComputer,pp.47-48,Jan.1998.

[26]J.Ziegler,“Terrestrialcosmicrays,”IBMJournalofResearchand Development,vol.40,pp.19-39,Jan.1996.

[27]D.P.Siewiorek,R.S.Swarz,“ReliableComputerSystems, DesignandEvaluationSecondEdition,”DigitalPress,BurlingtonMA 1992.

[28]S.K.ReinhardtandS.S.Mukherjee“TransientFaultDetectionvia SimulataneousMultithreading,”Proceedings27thAnnualInt’lSymp.on ComputerArchitecture(ISCA),June2000.

[29]E.Rotenberg,“AR-SMT:AMicroarchitecturalApproachtoFault ToleranceinMicroprocessors,”Proceedingsofthe29thFault-Tolerant ComputingSymposium,June1999.

[30]Z.Purser,K.Sundaramoorthy,andE.Rotenberg,“AStudyofSlipstreamProcessors,”33rdInterantionalSymposiumonMicroarchitecture, December2000.

[31]T.Austin.“DIVA:AReliableSubstrateforDeepSubmicron MicroarchitectureDesign”.In *Micro-32*,Nov99.

[32]T.Austin.“DIVA:ADynamicApproachtoMicroprocessorVerification.” *TheJournalofInstruction-LevelParallelismVolume2,* 2000.

[33]S.Chaterjee,C.WeaverandT.Austin.“EfficientCheckerProcessorDesign.”InMicro-33,Dec2000.

[34]J.HennessyandD.Patterson. *ComputerArchitectureaQuantitativeApproach* .MorganKaufmannPublishers,Inc.1996.

[35]SPECnewsletter.Fairfax,Virginia,Sept.1995.

[36]D.C.BurgerandT.M.Austin.Thesimplescalartoolset,version 2.0.TechnicalReportCS-TR-97-1342,UniversityofWisconsin,Madison, June1997.

[37]H.Al-Assad,J.P.HayesandB.T.Murray“Scalabletestgenerators forhigh-speeddatapathcircuits,”JournalofElectronicTesting:Theoryand Applications,Vol.12,Nos.1/2,Feburary/April1998

[38]T.JSlegeletal.“IBM’sS/390G5MicroprocessorDesign.” *IEEE Micro*,pp12-23,March/April1999.

[39]L.SpainhowerandT.Gregg,“G4:Afault-tolerantCMOSmainframe,”inProceedingsofthe28thFault-TolerantComputingSymposium, June1998.

[40]J.Gaisler,“Evaluationofa32-bitmicroprocessorwithbuilt-in concurrenterrordetection,”inProceedingsofthe27thFault-Tolerant ComputingSymposium,June1997.

[41]Y.TamirandM.Tremblay,“High-performancefaulttolerantVLSI systemsusingmicrorollback,”IEEETransactionsonComputers,vol.39, no.4,pp.548-554,1990.

[42]CahnersMicroprocessorReport,vol15,pp.50,August2000.

[43]GlennReinman,NormJouppi.“AnIntegratedCacheTimingand PowerModel.”http://research.compaq.com/wrl/people/jouppi/cacti2.pdf