July 1993

UILU-ENG-93-2229 CRHC-93-16

Center for Reliable and High-Performance Computing

APPLICATION OF COMPILER-ASSISTED MULTIPLE INSTRUCTION ROLLBACK RECOVERY TO SPECULATIVE EXECUTION

N.J. Alewine W. K. Fuchs W.-M. Hwu

Coordinated Science Laboratory College of Engineering UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Approved for Public Release. Distribution Unlimited.

REPORT SECURITY CLASSFICATION Ib. RESTRICTIVE MARKINGS Unclassified None SCURTY CLASSFICATION AUTHORITY 3. DSTINUTION / AVAILABUTY OF REPORT Approved for public release; distribution unlinited FERFORMING ORGANIZATION REPORT NUMBER(S) 5. MONITORING ORGANIZATION REPORT NUMBER(S) UTLU-ENG-93-2229 CRRC-93-16 NAME OF FERFORMING ORGANIZATION 6b. OFFICE SYMBOL Coordinated Science Lab M/A UNIVERSITY of Illinois N/A NAME OF FERFORMING ORGANIZATION 6b. OFFICE SYMBOL Coordinated Science Lab N/A University of Illinois N/A NAME OF FUNDING (FERNSONING ORGANIZATION 7a 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER ID10 W. Springfield Avenue 0. SUPICE OF FUNDING NUMBERS CORDES (Cip, State, and ZPCode) 10. SOURCE OF FUNDING NUMBERS The Endude Security Classification) Applicable) ADDRESS (Cip, State, and ZPCode) 10. SOURCE OF FUNDING NUMBERS The Endude Security Classification) N/A Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution 2. RESONAL AUTHOR(S) 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)				REPORT DOCUM	MENTATION	PAGE		
Unclassified Nome JBCLASSIFICATION AUTHORITY 3. DiSTRBUTION/AVALABUTY OF REPORT Approved for public release; distribution unlimited C. DECLASSIFICATION /DOWNGADING SCHEDULE 3. DISTRBUTION/AVALABUTY OF REPORT Approved for public release; distribution unlimited FERORMING ORGANIZATION REPORT NUMBERS) S. MONITORING ORGANIZATION REPORT NUMBERS) S. MONITORING ORGANIZATION (M sppikeNP) NAME OF FRORMING ORGANIZATION Coordinated Science Lab Bb. OFFICE SYMMOL (M sppikeNP) Ta. NAME OF MONITORING ORGANIZATION (M sppikeNP) NAME OF FRORMING ORGANIZATION Coordinated Science Lab Bb. OFFICE SYMMOL (M sppikeNP) Ta. NAME OF MONITORING ORGANIZATION NATIONAL APPLOACH ADDRESS (CP, Stee, and JP Code) NA Tb. ADDRESS (CP, Stee, and JP Code) NA IIIOL W. Springfield Avenue Urbana, IL Sb. OFFICE SYMMOL (M sppikeNP) Sp. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER TAGE OF FUNDING (SPONSONING ORGANIZATION 7a Bb. OFFICE SYMMOL (M sppikeNP) Sp. ROUGEC OF FUNDING HUMBERS TOLE (Include Security Cassification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution No. ACCESSION NC 2. SUBJECT TEAMS (Continue on reverse if necessary and identify by block number) 1903 Tip 1912 SPAGE COUNT 1923 Task: a repair rollback recovery, compiler-assisted multiple instruction recompiler-assisted multiple instru	REPORT SEC	URITY CLASSI	FICATION		16. RESTRICTIVE	MARKINGS		
SECURITY CLASSRCATION AUTHORITY 3. DISTRUUTION /AVALABLITY OF REPORT ADDECS (SC) Approved for public release; distribution unlimited Approved for public release; distribution unlimited S. MONITORING ORGANIZATION REPORT NUMBER(S) LILU-ENC-93-2229 CRRC-93-16 S. MONITORING ORGANIZATION REPORT NUMBER(S) LILU-ENC-93-2229 CRRC-93-16 S. MONITORING ORGANIZATION REPORT NUMBER(S) LILU-ENC-93-2229 CRRC-93-16 N/A NAME OF FROMMIC ORGANIZATION 6b. OFFICE SYMBOL Coordinated Science Lab N/A Urbana, IL 61801 N/A NAME OF FUNDING/SPONSORING 8b. OFFICE SYMBOL ORGANIZATION 7a 10. SOURCE OF FUNDING NUMBERS PROGRAM PROJECT TRAS ADDRESS (CHy, State, and ZPC Code) 10. SOURCE OF FUNDING NUMBERS The Brocked Security Compliant-Assisted Multiple Instruction Rollback Recovery to Speculative Execution TASK 2. PERSONAL AUTHOR(S) ALEWINE, N. J. W. K. Fuchs, and WM. Have 3. DIFFLEMENTARY NOTATION 10. SUBECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES FIELD GROUP 18. SUBECT TERMS (Continue on reverse if necessary and identify by block number) 7	line lassified				None			
DECLASSFICATION/DOWNGRADING SCHEDULE Approved for public release; distribution unlimited DECLASSFICATION/DOWNGRADING SCHEDULE Approved for public release; distribution unlimited PERFORMING ORGANIZATION REPORT NUMBER(S) S. MONITORING ORGANIZATION REPORT NUMBER(S) UILU-ENC-93-2229 CRRC-93-16 Coordinated Science Lab University of Illinois 6. OFFICE SYMBOL (# applicable) 7a. NAME OF MONITORING ORGANIZATION NATIONAL OF MONITORING ORGANIZATION National Aeronautics and Space Administral NA ADDRESS (GV, Stee, and JP Code) NA 7b. ADDRESS (CV, Stee, and JP Code) 101 W. Syningfield Avenue Urbana, IL 61801 NA 7a. Noffit Field, CA NAME OF FUNDING / SPONSORING ORGANIZATION 7a 8b. OFFICE SYMBOL (# applicable) 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER PROGRAM Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution 2. PERSONAL AUTHOR(S) ALEWINE, N. J, W. K. Fuchs, and WM. Hwu 14. DATE OF REPORT FROM 19. DATE COVERED FROM 14. DATE OF REPORT function 1993 15. PAGE COUNT 18 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) rollback recovery, compiler-assisted multiple instruction transient processor failures, instructional level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechana	SECURITY C	LASSIFICATION	AUTHORITY		3. DISTRIBUTION / AVAILABILITY OF REPORT			
DecArsming ORGANIZATION REPORT NUMBER(S) distribution unlimited PERFORMING ORGANIZATION REPORT NUMBER(S) S. MONTORING ORGANIZATION REPORT NUMBER(S) UTILU-ENG-93-2229 CRHC-93-16 NAME OF FROMMING ORGANIZATION Eb. OFFICE SYMBOL (# applicable) NAME OF FINONIG/SPONMING ORGANIZATION Eb. OFFICE SYMBOL (# applicable) NAME OF FINONIG/SPONMING ORGANIZATION Eb. OFFICE SYMBOL (# applicable) NAME OF FINONIG/SPONSORING ORGANIZATION 7a Bb. OFFICE SYMBOL (# applicable) NAME OF FINONIG/SPONSORING ORGANIZATION 7a Bb. OFFICE SYMBOL (# applicable) NOT TO INTIL (include Security Cassification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution 2. PERSONAL AUTHOR(S) ALLENTINE, N. J, W. K. Fuchs, and WM. Hwu 3a. TYPE OF REPORT 13b. TIME COVERED FROM 14. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT 7. COSATI CODES FRED 14. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT 18 7. COSATI CODES FRED 18. SUBECT TERMS (Continue on reverse if necessary and identify by block number) 193 119 18 7. COSATI CODES FRED 18. SUBECT TERMS (Continue on reverse if necessary and identify by block number) 193 119 18 7. COSA					Approved	for public	c release;	
PERFORMING ONGANIZATION REPORT NUMBER(5) 5. MONITORING ORGANIZATION REPORT NUMBER(5) UILU-ENG-93-2229 CRHC-93-16 NAME OF PERFORMING ONGANIZATION 66. OFFICE SYMBOL (7 applicable) UILU-ENG-93-2229 CRHC-93-16 NAME OF PERFORMING ONGANIZATION 66. OFFICE SYMBOL (7 applicable) UILU-ENG-93-2229 CRHC-93-16 NAME OF PERFORMING ONGANIZATION 66. OFFICE SYMBOL (7 applicable) NAME OF FUNDING/SPONSORING ORGANIZATION 68. OFFICE SYMBOL (7 applicable) NAME OF FUNDING / SPONSORING ORGANIZATION 18. OFFICE SYMBOL (7 applicable) NAME OF FUNDING / SPONSORING ORGANIZATION 18. OFFICE SYMBOL (7 applicable) 7 TAB 2. ADDRESS (CRY, State, and ZIP Code) 10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 7 TAB 3. TYPE OF REPORT 18. TIME COVERED FROM 7. TAB 3. TYPE OF REPORT 18. TIME COVERED FROM 7. 18. SUBJECT TERMS (Continue on reverse If Recessary and identify by block number) 7. COSATI CODES 7. 18. SUBJECT TERMS (Continue on reverse If Recessary and identify by block number) 7. COSATI CODES 7. 18. SUBJECT TERMS (Continue on reverse If	DECLASSIFIC	CATION / DOW	NGRADING SCHED	ULE	distribu	tion unlim	ited	
11ULU-ENG-93-222 CRRC-93-16 INAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of 11Linois 6: OFFICE SYMBOL (f' #ppicable) 7a. NAME OF MONITORING ORGANIZATION National Aeronautics and Space Administrat N/A ADDRESS (CRy, State, and ZP/Code) N/A National Aeronautics and Space Administrat N/A Iniversity of 11Linois N/A National Aeronautics and Space Administrat N/A INDRESS (CRy, State, and ZP/Code) NAME OF FUNDING /SPONSORING ORGANIZATION 7a 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER NAME OF FUNDING /SPONSORING ORGANIZATION 7a B. OFFICE SYMBOL (f' applicable) 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER TTELE (include Security Cassification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution Yes 2. FRESONAL AUTHOR(S) 2. FRESONAL AUTHOR(S) ALEWINE, N. J, W. K. Fuchs, and WM. How 2. Tree of REPORT 13b. TIME COVERED 193. Tuly 12 15. PAGE COUNT 18 3. SUPLEMENTARY NOTATION 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Tollback recovery, compiler-assisted multiple instruction transient processor failures, instructional level paral- lelian 3. ABSTRACT (Continue on reverse if necessary and identify by block number) 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	PERFORMING	ORGANIZATI	ON REPORT NUMB	ER(S)	5. MONITORING	ORGANIZATION	REPORT NUMB	ER(S)
TILD-ENC-93-2229 CRRC-93-10 INAME OF PERFORMING ORGANIZATION COORDINATED State, and ZUP Code) 6b. OFFICE SYMBOL (# applicable) NA IN AME OF PERFORMING ORGANIZATION COORDINATED State, and ZUP Code) NA National Aeronautics and Space Administral NATIONAL PROVIDENCE (# applicable) IIO W. Springfield Avenue Urbana, IL 61801 N/A The ADDRESS (City, State, and ZUP Code) NAME OF FUNDING/SPONSORING ORGANIZATION 7a Bb. OFFICE SYMBOL (# applicable) 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER ORGANIZATION 7a D 10. SOURCE OF FUNDING NUMBERS PROGRAM PROGRAM PROJECT TASK NO. ADDRESS (City, State, and ZUP Code) 10. SOURCE OF FUNDING NUMBERS PROGRAM PROGRAM PROJECT TASK NO. WORK UNIT ACCESSION NC 1. TITLE (Include Security Classification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution PROGRAM 14. DATE OF REPORT (Yesr, Month, Day) 15. PAGE COUNT 18 2. PERSONAL AUTHOR(S) ALEWINE, N. J, W. K. Fuchs, and WM. Hwu 193 Tuly 12 15. PAGE COUNT 18 3. TYPE OF REPORT FELD GROUP 13. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 193 Tuly 12 15. PAGE COUNT 18 7. COSATI CODES 13. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)								
NAME OF FEROMANIA ORGANIZATION 6. OFFICE SYMBOL (# applicable) 7. NAME OF MONITORING ORGANIZATION Coordinated Science Lab University of Illinois N/A National Aeronautics and Space Administrai ADDRESS (Chy, State, and ZP Code) N/A National Aeronautics and Space Administrai 1010 V. Springfield Avenue Urbana, IL 61801 NA PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Moffitt Field, CA NAME OF FUNDING /SPONSORING ORGANIZATION 7a 8b. OFFICE SYMBOL (f applicable) 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER PROGRAM Application of Complementation of Continue on reverse if necessary and identify by block number) 2. PEROCAL PERORT 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (UILU-ENG-	-93-2229	CRHC-93-16					
Coordinated Science Lab (if applicate) NA National Aeronautics and Space Administration is the paralelism which can be exploited by both super-solar and VLIW architectures. The key to a successful general estimation or reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES <td< td=""><td>NAME OF F</td><td>PERFORMING C</td><td>ORGANIZATION</td><td>6b. OFFICE SYMBOL</td><td>7a. NAME OF N</td><td>MONITORING OR</td><td>GANIZATION</td><td></td></td<>	NAME OF F	PERFORMING C	ORGANIZATION	6b. OFFICE SYMBOL	7a. NAME OF N	MONITORING OR	GANIZATION	
University of lilinois N/A ADDRESS (CRy, State, and ZP Code) The ADDRESS (CRy, State, and ZP Code) Ilol W. Springfield Avenue Woffitt Field, CA Urbana, IL 61801 NAM NAME OF FUNDING/SPONSORING ORGANIZATION Bb. OFFICE SYMBOL (# applicable) 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER ADDRESS (CRy, State, and ZIP Code) 0. SOURCE OF FUNDING NUMBERS WORK UNIT FOR SOURCE OF FUNDING NUMBERS ADDRESS (CRy, State, and ZIP Code) TASK WORK UNIT FOR SOURCE OF FUNDING NUMBERS TITLE (Include Security Classification) Applicable) NO. TASK Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution PROJECT TASK 2. FERSONAL AUTHOR(S) ALEWINE, N. J, W. K. Fuchs, and WM. Hwu 14. DATE OF REPORT (Year, Month, Day) 15. FAGE COUNT Technical 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUB	Coordin	ated Scie	nce Lab	(ir applicatie)	Nationa	1 Aeronauti	cs and Spa	ce Administrat
ADDRESS (Chy, State, and ZP Code) Noffitt Field, CA 1101 W. Springfield Avenue Noffitt Field, CA Urbana, IL 61801 Noffitt Field, CA ADDRESS (Chy, State, and ZP Code) 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER CADRESS (Chy, State, and ZP Code) 9. PROCURE OF FUNDING NUMBERS The ADDRESS (Chy, State, and ZP Code) 10. SOURCE OF FUNDING NUMBERS The ADDRESS (Chy, State, and ZP Code) 10. SOURCE OF FUNDING NUMBERS The ADDRESS (Chy, State, and ZP Code) 10. SOURCE OF FUNDING NUMBERS The ADDRESS (Chy, State, and ZP Code) 10. SOURCE OF FUNDING NUMBERS The ADDRESS (Chy, State, and ZP Code) 10. SOURCE OF FUNDING NUMBERS The ADDRESS (Chy, State, and ZP Code) 10. SOURCE OF FUNDING NUMBERS The ADDRESS (Chy, State, and ZP Code) 11. SUBJECT TERMS (Continue on reverse to Speculative Execution 2. PERSONAL AUTHOR(S) ALEWINE, N. J, W. K. Fuchs, and WM. Hwu 3a. TYPE OF REPORT 13. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Tollback recovery, compiler-assisted multiple instruction 11. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI	Univers	ity of 11	linois	N/A	The ADDRESS /	the State and 2	TIP Code)	
1101 W. Springfield Avenue Moffitt Field, CA Urbana, IL 61801 Moffitt Field, CA NAME OF PUNDING/SPONSORING ORGANIZATION 7a 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER 7a 10. SOURCE OF FUNDING NUMBERS 7b PROGRAM ELEMENT NO. PROJECT Task NO. WORK UNIT ACCESSION NC 1. TITLE (Include Security Classification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution PROSECT Task NO. WORK UNIT ACCESSION NC 2. PERSONAL AUTHOR(S) ALEWINE, N. J, W. K. Fuchs, and WM. Hwu 10. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT Technical 15. PAGE COUNT FROM 1993 July 12 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) rollback recovery, compiler-assisted multiple instruction transient processor failures, instructional level paral- lelist 9. ABSTRACT (Continue on reverse if necessary and identify by block number) Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instruction due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. 20. OSTRIBUTION /AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassifiedow 21. ABSTRACT SECURITY CLASSIFICATIO	ADDRESS (C	ity, State, and	ZIP Code)		70. ADDRESS (C	ity, state, and i	ur coder	
Urbana, II 6 61801 a NAME OF FUNDING/SPONSORING ORGANIZATION 7a 8b. OFFICE SYMBOL (If applicable) 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER To 2. ADDRESS (City, state, and ZIP Code) 7b 7b 7b 7c 7b 7c 7b 7c 7c 7b 7c	1101 W.	Springfi	eld Avenue		Moffit	t Field, CA	•	
AMME OF FUNDING/SRONSORING ORGANIZATION Bb. OFFICE SYMBOL (If applicable) 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER 7a 7a 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER 7a 7a 10. SOURCE OF FUNDING NUMBERS 7b PROGRAM PROJECT TASK NO. WORK UNIT ACCESSION NO. 7b 10. SOURCE OF FUNDING NUMBERS WORK UNIT RELEMENT NO. NO. WORK UNIT ACCESSION NO. 7b 10. SOURCE OF FUNDING NUMBERS PROJECT TASK NO. WORK UNIT ACCESSION NO. 7b 7b 10. SOURCE OF FUNDING NUMBERS WORK UNIT ACCESSION NO. WORK UNIT ACCESSION NO. 7 7b 10. SOURCE OF FUNDING NUMBERS TASK NO. WORK UNIT ACCESSION NO. 7 7b 14. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT Technical 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7 COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 1993 7 COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 1993 7 COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 1993	Urbana,	IL 6180)1					
ORGANIZATION 7a (if applicable) 2. ADDRESS (City, State, and ZIP Code) 10. SOURCE OF FUNDING NUMBERS 7b PROGRAM PROJECT TASK WORK UNIT 7b 7b NO. NO. NO. WORK UNIT 7b 7b NO. NO. WORK UNIT 7b 7b NO. NO. WORK UNIT 7b 7b NO. NO. WORK UNIT 7b NO. NO. NO. WORK UNIT 7c COSATICON OR ALEWINE, N. J, W. K. Fuchs, and WM. Hwu 15. PAGE COUNT 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 13. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 19.3 July 12 18 7. COSATI CODES 13. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 19.1 Lettist 7. COSATI CODES 19.3 July 12 18 7. COSATI CODES 19.4 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 19.3 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) <td>. NAME OF</td> <td>UNDING / SPO</td> <td>NSORING</td> <td>8b. OFFICE SYMBOL</td> <td>9. PROCUREME</td> <td>NT INSTRUMENT</td> <td>IDENTIFICATION</td> <td>NUMBER</td>	. NAME OF	UNDING / SPO	NSORING	8b. OFFICE SYMBOL	9. PROCUREME	NT INSTRUMENT	IDENTIFICATION	NUMBER
2. ADDRESS (City, State, and ZIP Code) 10. SOURCE OF FUNDING NUMBERS 7b 7b 7b NO. 1. TITLE (include Security Classification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution 2. PERSONAL AUTHOR(S) 2. PERSONAL AUTHOR(S) 3a. TYPE OF REPORT 13b. TIME COVERED 70 14. DATE OF REPORT 13b. TIME COVERED 70 14. DATE OF REPORT 13b. TIME COVERED 70 14. DATE OF REPORT 13b. TIME COVERED 14. DATE OF REPORT 13b. TIME COVERED 14. DATE OF REPORT 14. DATE OF REPORT 13b. TIME COVERED 14. DATE OF REPORT 15. SUBJECT TERMS (Continue on reverse if mecessary and identify by block number) rollback recovery, compiler-assited multiple instruction 16. SUBJECT TERMS (Continue on reverse if mecessary and identify by block number) 7 Tollback recovery, compiler-assited multiple instruction 7 COSATI CODES 18. SUBJECT TERMS (Continue on reverse if mecessary and identify by block number) 7	ORGANIZAT	TION 7a		(If applicable)				
2. ADDRESS (City, State, and ZiP Code) 10. SOURCE OF "FUNCING NUMBERS 7b 7b 1. TITLE (include Security Classification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution 2. PERSONAL AUTHOR(S) 3a. TYPE OF REPORT 13b. TIME COVERED FROM 14. DATE OF REPORT 13b. TIME COVERED 14. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) rollback recovery, compiler-assisted multiple instruction transient processor failures, instructional level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction relback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative execution architecute are similar to these encountered during multiple instruction rollback. 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION 22. DISTR							2505	
7b NO. NO. ACCESSION NC 1. TITLE (include Security Classification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution 2. PERSONAL AUTHOR(S) ALEWINE, N. J, W. K. Fuchs, and WM. Hwu 3a. TYPE OF REPORT 13b. TIME COVERED 14. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT Technical FROM TO 1493 July 12 18 5. SUPPLEMENTARY NOTATION 15. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) rollback recovery, compiler-assisted multiple instruction transient processor failures, instructional level paral- lelism 9. ABSTRACT (Continue on reverse if necessary and identify by block number) Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution repair. Extensions to the ormpiler-assisted multiple instruction rollback scheme to aid in speculative execution repair. Extensions to the ormpiler-assisted multiple instruction programs. 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified.	ADDRESS (C	ity, State, and	ZIP Code)		10. SOURCE OF	PROJECT	TASK	WORK UNIT
7b 1. TITLE (include Security Cassification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution 2. PERSONAL AUTHOR(S) ALEWINE, N. J, W. K. Fuchs, and WM. Hwu 2. TYPE OF REPORT 13b. TIME COVERED FROM 14. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT Technical 7. COSATI CODES FROM 10 14. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT Technical 7. COSATI CODES FROM 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) rollback recovery, compiler-assisted multiple instruction transient processor failures, instructional level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution rachitecute are similar to those encountered during multiple instruction rollback. 20. DISTRIBUTION/AVAILABULITY OF ABSTRACT CINC USER 21. ABSTRACT SECURITY CLASSIFICATION Unclassified					ELEMENT NO.	NO.	NO.	ACCESSION NO
1. TITLE (include Security Classification) Application of Compiler-Assisted Multiple Instruction Rollback Recovery to Speculative Execution 2. PERSONAL AUTHOR(S) 3a. TYPE OF REPORT 13b. TIME COVERD 14. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT 13a. TYPE OF REPORT 13b. TIME COVERD 14. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT 13c. TYPE OF REPORT 13b. TIME COVERD 14. DATE OF REPORT (Year, Month, Day) 15. PAGE COUNT 13c. TYPE OF REPORT 13b. TIME COVERD 1993 July 12 18 6. SUPPLEMENTARY NOTATION 1993 July 12 18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) rollback recovery, compiler-assisted multiple instruction transient processor failures, instructional level paral- lelist 9. ABSTRACT (Continue on reverse if necessary and identify by block number) Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution rollback scheme to aid in specu		71	0			and the second		
5. SUPPLEMENTARY NOTATION 7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) FIELD GROUP 9. ABSTRACT (Continue on reverse if necessary and identify by block number) 9. ABSTRACT (Continue on reverse if necessary and identify by block number) Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 220. NAME OF RESPONSIBLE INDIVIDUAL 21. ABSTRACT SECURITY CLASSIFICATION	Applica Exe	AUTHOR(S)	ALEWINE, N.	isted Multiple In J, W. K. Fuchs, a	nstruction R and WM. Hw	oİlback Rec u	covery to S	peculative
7. COSATI CODES 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) FIELD GROUP SUB-GROUP rollback recovery, compiler-assisted multiple instruction transient processor failures, instructional level paral-lelism 9. ABSTRACT (Continue on reverse if necessary and identify by block number) Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 22a. NAME OF RESPONSIBLE INDIVIDUAL 21. ABSTRACT SECURITY CLASSIFICATION	Applica Exe 2. PERSONAL 3a. TYPE OF Technic	AUTHOR(S) REPORT	ALEWINE, N	ISTED Multiple In J, W. K. Fuchs, a COVERED	and WM. Hw 14. DATE OF REF 1993	oʻllback Rec u PORT (Year, Mon July 12	tovery to Spanners to S	AGE COUNT
FIELD GROUP SUB-GROUP rollback recovery, compiler-assisted multiple instruction transient processor failures, instructional level paral-lelism 9. ABSTRACT (Continue on reverse if necessary and identify by block number) Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative execution repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 22a. NAME OF RESPONSIBLE INDIVIDUAL 21. ABSTRACT SECURITY CLASSIFICATION	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME	AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT	ALEWINE, N 13b. TIME FROM	isted Multiple In J, W. K. Fuchs, a COVERED TO	and WM. Hw 14. DATE OF REF 1993	oʻllback Rec u PORT (Year, Mon July 12	overy to Spanners to S	AGE COUNT
transient processor failures, instructional level paral- lelis transient processor failures, instructional level paral- lelis sort (Continue on reverse if necessary and identify by block number) Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative execution repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION /AVAILABILITY OF ABSTRACT DIC USERS 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 22b. TELEPHONE (include Area Code) 22c. OFFICE SYMBOL	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7.	AUTHOR(S) AUTHOR(S) REPORT COSATI	ALEWINE, N 13b. TIME FROM TION	ISTED Multiple In J, W. K. Fuchs, a COVERED TO	and WM. Hw 14. DATE OF REF 1993	oʻllback Rec u PORT (Year, Mon July 12	and identify by	AGE COUNT 18
9. ABSTRACT (Continue on reverse if necessary and identify by block number) 9. ABSTRACT (Continue on reverse if necessary and identify by block number) Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 220. DISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 220. NISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 220. DISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD	AUTHOR(S) AUTHOR(S) REPORT COSATI GROUP	ALEWINE, N 13b. TIME FROM TION CODES SUB-GROUP	Isted Multiple In J, W. K. Fuchs, a COVERED TO	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com	oʻllback Rec u PORT (Year, Mon July 12 erse if mecessary piler-assis	and identify by sted multip	AGE COUNT 18 block number) le instruction
 9. ABSTRACT (Continue on reverse if necessary and identify by block number) Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. DISTRIBUTION/AVAILABILITY OF ABSTRACT AME OF RESPONSIBLE INDIVIDUAL 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 22b. TELEPHONE (Include Area Code) 22c. OFFICE SYMBOL 	Applica Exe 2. PERSONAL 3a. TYPE OF <u>Technic</u> 6. SUPPLEME 7. FIELD	AUTHOR(S) AUTHOR(S) REPORT COSATI GROUP	ALEWINE, N. (13b. TIME FROM_ TION CODES SUB-GROUP	Isted Multiple In J, W. K. Fuchs, a COVERED 	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f	oʻllback Rec u PORT (Year, Mon July 12 rse if necessary piler-assis ailures, in	and identify by sted multip nstructiona	AGE COUNT 18 block number) le instruction l level paral-
Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs.	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD	AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT	ALEWINE, N. ALEWIN	Isted Multiple In J, W. K. Fuchs, a COVERED TO	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f	oʻllback Rec u PORT (Year, Mon July 12 rse if mecessary piler-assis ailures, in	and identify by sted multipustructiona	AGE COUNT 18 block number) le instruction l level paral- lelist
Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs.	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT	COSATI GROUP	ALEWINE, N 13b. TIME FROM TION CODES SUB-GROUP reverse if necessa	Isted Multiple In J, W. K. Fuchs, a COVERED TO	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f number)	oʻllback Rec u PORT (Year, Mon July 12 rse if mecessary piler-assis ailures, in	and identify by asted multip astructiona	AGE COUNT 18 block number) le instruction l level paral- lelisn
 super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. DISTRIBUTION/AVAILABILITY OF ABSTRACT ABSTRACT SECURITY CLASSIFICATION Unclassified ABME AS RPT. DTIC USERS ABSTRACT SECURITY CLASSIFICATION Unclassified ADME OF RESPONSIBLE INDIVIDUAL 	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT	AUTHOR(S) AUTHOR(S) REPORT COSATI GROUP	ALEWINE, N. ALEWINE, SCIERCOM TON CODES SUB-GROUP Freverse if necessa	Isted Multiple In J, W. K. Fuchs, a COVERED 	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f number)	oʻllback Rec u PORT (Year, Mon July 12 erse if mecessary piler-assis ailures, in	and identify by attack multip	AGE COUNT 18 block number) le instruction l level paral- lelisn
 mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 22b. TELEPHONE (Include Area Code) 22c. OFFICE SYMBOL 	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT	AUTHOR(S) AUTHOR(S) REPORT COSATI GROUP (Continue on	ALEWINE, N ALEWINE, N 13b. TIME FROM TION CODES SUB-GROUP reverse if necessa xecution is a me	Isted Multiple In J, W. K. Fuchs, a COVERED TO	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f number) action level para	ollback Rec u PORT (Year, Mon July 12 mailer-assis ailures, in	and identify by sted multip nstructiona	AGE COUNT 18 block number) le instruction l level paral- lelisn ed by both
Multiple instruction rollback is a technique developed for recovery from transient processor failure. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT SUNCLASSIFIED/UNLIMITED SAME AS RPT. DTIC USERS 22a. NAME OF RESPONSIBLE INDIVIDUAL 22b. TELEPHONE (Include Area Code) 22c. OFFICE SYMBOL	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT S s	AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT COSATI GROUP (Continue on Speculative exuper-scalar a	ALEWINE, N. ALEWIN	Isted Multiple In J, W. K. Fuchs, a COVERED 	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f number) uction level para o a successful g	ollback Rec u PORT (Year, Mon July 12 prose if necessary piler-assis ailures, in allelism which of general specula	and identify by sted multip astructiona.	AGE COUNT 18 block number) le instruction l level paral- lelisn ed by both is a repair
 of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exception in a speculative execution architecute are similar to those encountered during multiple instruction rollback. This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT Same AS RPT. DTIC USERS 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 22a. NAME OF RESPONSIBLE INDIVIDUAL 21. ALL SCIEPHONE (Include Area Code) 22c. OFFICE SYMBOL 22c. OFFICE SYMBOL 22c. OFFICE SYMBOL 22 and the difficulties and the second statements across is a specific to the second statement of the second	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT S s n	AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT COSATI GROUP (Continue on Speculative et uper-scalar a nechanism to	ALEWINE, N. ALEWIN	Isted Multiple In J, W. K. Fuchs, a COVERED 	And WM. Hw 14. DATE OF REF 1993 (Continue on reve covery, com processor f number) uction level para o a successful g urate reporting of	ollback Rec u PORT (Year, Mon July 12 rse if necessary piler-assis ailures, in allelism which of general specula of exceptions fo	and identify by sted multip nstructiona can be exploit tion strategy or speculated in	AGE COUNT 18 block number) le instruction l level paral- lelisn ed by both is a repair istrucitons.
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 222. NAME OF RESPONSIBLE INDIVIDUAL 22b. TELEPHONE (Include Area Code) 22c. OFFICE SYMBOL	Applica Exe 2. PERSONAL 3a. TYPE OF <u>Technic</u> 6. SUPPLEME 7. FIELD 9. ABSTRACT S S M M	AUTHOR(S) AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT GROUP (Continue on Cosati GROUP (Continue on Cosati GROUP	ALEWINE, N. ALEWINE, N. 13b. TIME FROM TION CODES SUB-GROUP reverse if necessa xecution is a me and VLIW archi b handle mispredi ruction rollback	Isted Multiple In J, W. K. Fuchs, a COVERED TO 18. SUBJECT TERMS rollback re transient ry and identify by block thod to increase instru- itectures. The key to cted branches and accu- is a technique develop	(Continue on reve ecovery, com processor f number) uction level para o a successful g urate reporting o	ollback Rec u PORT (Year, Mon July 12 and if necessary piler-assis ailures, in allelism which of general specular of exceptions for from transient	and identify by sted multipustructiona can be exploited tion strategy or speculated in processor fail	AGE COUNT 18 block number) le instruction l level paral- lelist ed by both is a repair strucitons. ure. Many
Instruction relation This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified Image: Strand Content of the security of the secure of the security of the security of the security of the security	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT S S S M M O O	AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT COSATI GROUP (Continue on Speculative et uper-scalar a nechanism to Aultiple instr f the difficul	ALEWINE, N. 13b. TIME FROM_ TION CODES SUB-GROUP reverse if necessa xecution is a me and VLIW archi b handle mispredi ruction rollback tties encountered	Isted Multiple In J, W. K. Fuchs, a COVERED TO	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f number) uction level para o a successful g urate reporting of branch mispred	ollback Rec u PORT (Year, Mon July 12 rse if necessary piler-assis ailures, in allelism which of general specular of exceptions fo from transient diction or from	can be exploited in struction as can be exploited in struction re- processor fail	AGE COUNT 18 block number) le instruction l level paral- lelist ed by both is a repair strucitons. ure. Many execution a multiple
 rollback scheme to aid in speculative exectuion repair. Extensions to the ocmpiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Unclassified 22a. NAME OF RESPONSIBLE INDIVIDUAL 22b. TELEPHONE (Include Area Code) 22c. OFFICE SYMBOL 	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 5. SUPPLEME 7. FIELD 9. ABSTRACT S S S M M O O d	AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT COSATI GROUP (Continue on Epeculative ex uper-scalar a nechanism to Aultiple instrict of the difficul lue to except netruction ro	ALEWINE, N. ALEWINE, N. 13b. TIME FROM_ TION CODES SUB-GROUP reverse if necessa xecution is a me and VLIW archi b handle mispredir ruction rollback ties encountered tion in a specula	Isted Multiple In J, W. K. Fuchs, a COVERED TO 18. SUBJECT TERMS rollback re transient ry and identify by block thod to increase instru- itectures. The key to cted branches and accr- is a technique develop during recovery from ative execution archite	And WM. Hw 14. DATE OF REF 1993 (Continue on reve covery, com processor f number) uction level para o a successful g urate reporting of branch mispred ecute are simila	DORT (Year, Mon July 12 PORT (Year, Mon July 12 Prse if necessary piler-assis ailures, in allelism which a general specula of exceptions fo from transient diction or from r to those enco	and identify by sted multip astructiona can be exploit tion strategy or speculated in processor fail instruction re- pountered durin	AGE COUNT 18 block number) le instruction l level paral- lelisn ed by both is a repair istrucitons. ure. Many e-execution ag multiple
branch and exception repair are presented along with performance measurements across ten application programs. 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED SAME AS RPT. DTIC USERS 22a. NAME OF RESPONSIBLE INDIVIDUAL 22b. TELEPHONE (Include Area Code) 22c. OFFICE SYMBOL	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT S S S M N O O d in	AUTHOR(S) AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT COSATI GROUP (Continue on Epeculative exuper-scalar inechanism to Aultiple instrict of the difficul lue to except instruction ro This pape	ALEWINE, N. ALEWINE, N. 13b. TIME FROM TION CODES SUB-GROUP reverse if necessa xecution is a me and VLIW archi b handle mispredi ruction rollback ties encountered tion in a specula bilback. er investigates th	Isted Multiple In J, W. K. Fuchs, a COVERED TO 18. SUBJECT TERMS rollback re transient ry and identify by block thod to increase instru- itectures. The key to cted branches and accu- is a technique develop during recovery from ative execution archite- te applicability of a re	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f number) uction level para b a successful g urate reporting of bed for recovery a branch mispred ecute are simila	ollback Rec u PORT (Year, Mon July 12 rse if necessary piler-assis ailures, in allelism which of general specular of exceptions fo from transient diction or from r to those enco d compiler-assi	and identify by sted multipus structiona. can be exploited tion strategy or speculated in processor fail in instruction re- pountered during isted multiple	AGE COUNT 18 block number) le instruction l level paral- lelist ed by both is a repair strucitons. ure. Many -execution ng multiple instruciton
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED SAME AS RPT. DTIC USERS 222. NAME OF RESPONSIBLE INDIVIDUAL 225. TELEPHONE (Include Area Code) 22c. OFFICE SYMBOL	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT S S S M M O O d in N	AUTHOR(S) AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT COSATI GROUP (Continue on Opeculative et uper-scalar a nechanism to Aultiple instruction ro This pape ollback scher	ALEWINE, N. ALEWINE, N. 13b. TIME FROM_ TION CODES SUB-GROUP reverse if necessa xecution is a me and VLIW archi b handle mispredi ruction rollback ties encountered tion in a specula blback. er investigates th ne to aid in specu	Isted Multiple In J, W. K. Fuchs, a COVERED TO	And WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f number) uction level para to a successful g urate reporting of branch mispred ecute are simila cently developed r. Extensions to	DORT (Year, Mon July 12 PORT (Year, Mon July 12 Prose if necessary piler-assis ailures, in allelism which of general specular of exceptions fo from transient diction or from r to those enco d compiler-assis the ocmpiler-assis	and identify by and identify by sted multip astructiona can be exploit tion strategy or speculated in processor fail instruction re- pountered durin isted multiple ssisted scheme	AGE COUNT 18 block number) le instruction l level paral- lelist ed by both is a repair strucitons. ure. Many execution ng multiple instruciton to support
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT 21. ABSTRACT SECURITY CLASSIFICATION Image: Strain of the st	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT S S S N M O d d in b	AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT COSATI GROUP (Continue on Epeculative et uper-scalar a nechanism to Aultiple instruction ro This pape ollback scher oranch and e	ALEWINE, N. ALEWINE, N. 13b. TIME FROM_ TION CODES SUB-GROUP reverse if necessa xecution is a me and VLIW archi b handle mispredi ruction rollback ties encountered tion in a specula blback. er investigates th ne to aid in specula exception repair	Isted Multiple In J, W. K. Fuchs, a COVERED TO 18. SUBJECT TERMS rollback re transient ry and identify by block thod to increase instru- itectures. The key to cted branches and accu- is a technique develop during recovery from ative execution archite- te applicability of a re- plative exectuion repain are presented along v	And WM. Hw 14. DATE OF REF 1993 (Continue on reve accovery, com processor f number) uction level para to a successful g urate reporting of bed for recovery a branch mispred becute are similar cently developed r. Extensions to with performance	DORT (Year, Mon July 12 PORT (Year, Mon July 12 Pose if necessary piler-assis ailures, in allelism which a general specula of exceptions fo from transient diction or from r to those enco d compiler-assi the ocmpiler-assi the ocmpiler-assi ce measuremen	and identify by and identify by sted multip astructiona can be exploit tion strategy or speculated in processor fail a instruction re- pountered durin asted multiple ssisted scheme ts across ten a	AGE COUNT 18 block number) le instruction l level paral- lelisn ed by both is a repair strucitons. ure. Many e-execution to support application
Image: Strate of the strate	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT S S S M M O O d in FIELD P	AUTHOR(S) AUTHOR(S) REPORT COSATI GROUP COSATI GROUP COSATI GROUP Continue on Continue on Continue on Cosati GROUP	ALEWINE, N. ALEWINE, N. 13b. TIME FROM TION CODES SUB-GROUP reverse if necessa xecution is a me and VLIW archi b handle mispredi ruction rollback ties encountered tion in a specula ollback. er investigates th ne to aid in specu	Isted Multiple In J, W. K. Fuchs, a COVERED TO 18. SUBJECT TERMS rollback re transient ry and identify by block thod to increase instru- itectures. The key to cted branches and accu- is a technique develop during recovery from ative execution archite- te applicability of a re- plative exectuion repair are presented along v	(Continue on reve and WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f number) uction level para o a successful g urate reporting of bed for recovery branch mispred ecute are simila cently developed r. Extensions to with performance	allelism which a general specular of from transient diction or from r to those enco d compiler-assis the ocmpiler-assis the ocmpiler-assis	and identify by and identify by sted multipus structiona. can be exploited tion strategy or speculated in processor fail instruction re- pountered during isted multiple ssisted scheme ts across ten a	AGE COUNT AGE COUNT 18 block number) le instruction l level paral- lelist ed by both is a repair strucitons. ure. Many e-execution ag multiple instruciton to support application
22a. NAME OF RESPONSIBLE INDIVIDUAL 22b. TELEPHONE (Include Area Code) 22c. OFFICE SYMBOL	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT 9. ABSTRACT S S S S M M O d d in P	AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT COSATI GROUP (Continue on Opeculative et uper-scalar a nechanism to Aultiple instruction ro This pape ollback scher pranch and e programs.	ALEWINE, N. ALEWINE, N. 13b. TIME FROM_ TION CODES SUB-GROUP reverse if necessa xecution is a me and VLIW archi b handle mispredi ruction rollback ties encountered tion in a specula blback. er investigates th ne to aid in specula exception repair	Isted Multiple In J, W. K. Fuchs, a COVERED TO TO TO TO TO TO TO TO TO TO	And WM. Hw 14. DATE OF REF 1993 (Continue on reve covery, com processor f number) action level para b a successful g urate reporting of bed for recovery a branch mispred bed for recovery branch mispred branch mis	DORT (Year, Mon July 12 PORT (Year, Mon July 12 Prose if necessary piler-assis ailures, in allelism which of general specular of exceptions fo from transient diction or from r to those enco d compiler-assis the ocmpiler-assis the ocmpiler-assis	and identify by and identify by sted multip astructiona can be exploit tion strategy or speculated in processor fail instruction re- pountered durin isted multiple ssisted scheme ts across ten a	AGE COUNT 18 block number) le instruction l level paral- lelisn ed by both is a repair strucitons. ure. Many execution to support application
	Applica Exe 2. PERSONAL 3a. TYPE OF Technic 6. SUPPLEME 7. FIELD 9. ABSTRACT 9. ABSTRACT 9. ABSTRACT 9. ABSTRACT 9. ABSTRACT 9. ABSTRACT 9. ABSTRACT 9. ABSTRACT	AUTHOR(S) AUTHOR(S) REPORT Cal NTARY NOTAT COSATI GROUP (Continue on peculative et uper-scalar a nechanism to Aultiple instruction ro This pape ollback scher pranch and e programs.	ALEWINE, N. ALEWIN	Isted Multiple In J, W. K. Fuchs, a COVERED TO 18. SUBJECT TERMS rollback re transient ry and identify by block thod to increase instru- itectures. The key to cted branches and accu- is a technique develop during recovery from ative execution archite the applicability of a re thative execution repair are presented along v	(Continue on reve and WM. Hw 14. DATE OF REF 1993 (Continue on reve ecovery, com processor f number) uction level para to a successful g urate reporting of bed for recovery branch mispred ecute are simila cently developed to Extensions to with performance	PORT (Year, Mon July 12 PORT (Year, Mon July 12 Procession of the second allelism which a general specular of exceptions fo from transient diction or from r to those enco d compiler-assis the ocmpiler-assis the ocmpiler-assis	and identify by and identify by sted multip astructiona can be exploit tion strategy or speculated in processor fail a instruction re- pountered durin isted multiple ssisted scheme ts across ten a SIFICATION	AGE COUNT 18 block number) le instruction l level paral- lelisn ed by both is a repair strucitons. ure. Many execution to support application

UNCLASSIFIED

TO APPEAR: WORKSHOP ON HARDWARE AND SOFTWARE ARCHITECTURES FOR FAULT TOLERANCE: PERSPECTIVES AND TOWARDS A SYNTHESIS JUNE 14-16, 1993 LE MONT SAINT-MICHEL, FRANCE

APPLICATION OF COMPILER-ASSISTED MULTIPLE INSTRUCTION ROLLBACK RECOVERY TO SPECULATIVE EXECUTION

N. J. Alewine, W. K. Fuchs, W.-M. Hwu

Center for Reliable and High-Performance Computing Coordinated Science Laboratory University of Illinois at Urbana-Champaign

Abstract

Speculative execution is a method to increase instruction level parallelism which can be exploited by both super-scalar and VLIW architectures. The key to a successful general speculation strategy is a repair mechanism to handle mispredicted branches and accurate reporting of exceptions for speculated instructions. Multiple instruction rollback is a technique developed for recovery from transient processor failures. Many of the difficulties encountered during recovery from branch misprediction or from instruction re-execution due to exceptions in a speculative execution architecture are similar to those encountered during multiple instruction rollback.

This paper investigates the applicability of a recently developed compiler-assisted multiple instruction rollback scheme to aid in speculative execution repair. Extensions to the compiler-assisted scheme to support branch and exception repair are presented along with performance measurements across ten application programs.

1 Introduction

Super-scalar and VLIW architectures have been shown effective in exploiting instruction level parallelism (ILP) present in a given application [1-3]. Creating additional ILP in applications has been the subject of study in recent years [4-6]. Code motion within a basic block is insufficient to unlock the full potential of super-scalar and VLIW processors with issue rates greater than two [3]. Given a trace of the most frequently executed basic blocks, limited code movement across block boundaries can create additional ILP at the expense of requiring complex compensation code to ensure program correctness [7]. Combining multiple basic blocks into *superblocks* permits code movement within the superblock without the compensation code required in standard trace scheduling [3].

General upward and downward code movement across trace entry points (joins) and general downward code motion across trace exit points (branches, or forks) is permitted without the need for special hardware support [7]. Sophisticated hardware support is required, however, for unrestricted upward code motion across a branch boundary. Such code motion is referred to as speculative execution and has been shown to substantially enhance performance over nonspeculated architectures [8-10]. This paper focuses on the support hardware for speculative execution, which ensures correct operation in the presence of excepting speculated instructions (referred to as exception repair) and of mispredicted branches (referred to as branch repair). It is shown that data hazards which result from exception and branch repair are very similar to data hazards that result from multiple instruction rollback, and that techniques used to resolve rollback data hazards are applicable to exception and branch repair.

The remainder of the paper is organized as follows. Section 2 gives a brief overview of a compiler-assisted multiple instruction rollback (MIR) scheme to be used as a base for application to speculative execution repair (SER). Section 3 describes speculative execution and the requirements for exception repair and branch repair. Section 4 introduces a schedule reconstruction scheme and extends the compiler-assisted rollback scheme. Section 5 describes read buffer flush costs and Section 6 presents performance impacts which result

^{*}International Business Machines Corporation, Boca Raton, Fl.

¹This research was supported in part by the National Aeronautics and Space Administration (NASA) under grant NASA NAG 1-613, in cooperation with the Illinois Computer Laboratory for Aerospace Systems and Software (ICLASS), and in part by the Department of the Navy and managed by the Office of the Chief of Naval Research under Contract N00014-91-J-1283.

from read buffer flushes.

2 Compiler-Assisted Multiple Instruction Rollback Recovery

2.1 Hazard Classification

Within a general error model, data hazards resulting from instruction retry are of two types [11-13]. On-path hazards are those encountered when the instruction path after rollback is the same as the initial path and branch hazards are those encountered when the instruction path after rollback is different than the initial path. As shown in Figure 1, r_x represents an on-path hazard where during the initial instruction se-



Figure 1: On-path data hazard.

quence r_x is written and after rollback is read prior to being re-written. As shown in Figure 2, r_y represents a branch hazard where the initial instruction sequence writes r_y and after rollback r_y is read prior to being rewritten however this time not along the original path.

2.2 On-path Hazard Resolution Using a Read Buffer

Hardware support consisting of a read buffer of size 2N, as shown in Figure 3, has been shown to be effective in resolving on-path hazards [11-13]. The read buffer maintains a window of register read history. If an on-path hazard is present, then prior to writing over the old value of the hazard register, a read of that value must have taken place within the last N instructions (else after rollback of $\leq N$, a read of the hazard register would not occur before a redefinition). Key to this scenario is the fact that the original path is repeated. Branch hazard resolution is left to the



Figure 2: Branch data hazard.



Figure 3: Read buffer.

compiler. At rollback, the read buffer is flushed back to the general purpose register file (GPRF), restoring the register file to a restartable state. The primary advantage of the read buffer is that it does not require an additional read port as with a history buffer, replication of the GPRF as with the future file, or bypass logic as with the reorder buffer or delayed write buffer [14, 15].

2.3 Branch Hazard Removal Compiler Transformations

Compiler transformations have been shown to be effective in resolving branch hazards [11, 12]. Branch hazard resolution occurs at three levels; 1) pseudo code, 2) machine code, and 3) post-pass. Resolution at the pseudo code level would be accomplished by renaming the pseudo register r_y of instruction I_i (Figure 2) to r_z . Node splitting, loop expansion and loop protection transformations aid in breaking pseudo register equivalence relationships so that renaming can be performed. After the pseudo registers are mapped to physical registers, some branch hazards could reappear. This is prevented at the machine code level by adding hazard constraints to live range constraints prior to register allocation. Branch hazards that remain after the first two levels can be resolved by either creating a "covering" on-path hazard or by inserting nop (no operation) instructions ahead of the hazard instruction until the rollback is guaranteed to be under the branch. Given the branch hazard of Figure 2, a covering on-path hazard is created by inserting an MOV r_y, r_y instruction immediately before the instruction in which r_y is defined. This guarantees that the old value of r_y is loaded into the read buffer and is available to restore the register file during rollback.

3 Speculative Execution

Figures 4 and 5 illustrate the two basic problems which are encountered when attempting upward code motion across a branch. As shown in Figure 4, if the



Figure 4: r1 in live_out of taken path.

speculated instruction (i.e., an instruction moved upward past one or more branches) modifies the system state, and due to the branch outcome the speculated instruction should not have been executed, program correctness could be affected. Figure 5 illustrates that if the speculated instruction causes an exception, and again due to the branch outcome, the excepting instruction should not have been executed, program performance or even program correctness could be affected.



Figure 5: Speculated instruction traps.

3.1 Branch Repair

Figure 6 shows an original instruction schedule and a new schedule after speculation. Instructions d, i, and f have been speculated above branches c and q from their respective fall-through paths.² Speculated instructions are marked "(s)." The motivation for such a schedule might be to hide the load delay of the speculated instructions or to allow more time for the operands of the branch instructions to become available. If c commits to the taken path (i.e., it is mispredicted by the static scheduler), some changes to the system state that have resulted from the execution of d, i, and f, may have to be undone. No update is required for the PC; execution simply begins at j. If instead, c commits to the fall-through path but gcommits to the taken path, then only i's changes to the system state may have to be undone.

Not all changes to the system state are equally important. If for example, d writes to register r_x and $r_x \notin live_in(j)$ (i.e., along the path starting at j, a redefinition of r_x will be encountered prior to a use of r_x [16]), then the original value of r_x does not have to be restored. Inconsistencies to the system state as a result of mispredicted branches exhibit similarities to branch hazards in multiple instruction rollback [11, 12]. Given this similarity between branch hazards due to speculative execution, compiler-driven dataflow manipulations, similar to those developed to eliminate branch hazards for MIR [11, 12], can be used to resolve branch hazards that result from speculation. Such compiler transformations have been proposed for

²For this example it is assumed that the fall-through paths are the most likely outcome of the branch decisions at c and g.



Figure 6: Branch repair.

branch misprediction handling [9]. Since re-execution of speculated instructions is not required for branch misprediction, compiler resolution of branch hazards becomes a sufficient branch repair technique.

3.2 Exception Repair

Figure 6 also demonstrates the handling of speculated trapping instructions. If d is a trapping instruction and an exception occurred during its execution, handling of the exception must be delayed until c commits so that changes to the system state are minimized, and in some cases to ensure that repair is possible in the event that c is mispredicted. If c commits to the taken path, the exception is ignored and d is handled like any other speculated instruction given a branch mispredict. If c was correctly predicted, three exception repair strategies are possible. The first is to undo the effects of only those instructions speculated above c (i.e., d, i, and f) and then branch to a recovery block RB_c [10] as shown in Figure 6. The address of the recovery block can be obtained by using the PC value of the excepting instruction as an index into a hash table. This strategy ensures precise interrupts [14, 17] relative to the nonspeculated schedule but not relative to the original schedule. Recovery blocks can cause significant code growth [10]. The second strategy undoes the effects of all instructions subsequent to d (i.e., i, b, and f), handles the exception, and resumes execution at instruction i [9]. This latter strategy provides restartable states and does not require recovery blocks. A third exception repair strategy undoes the effects of only those subsequent instructions that are speculated above c (i.e., only i and f), handles the exception, and resumes execution at instruction i, however, this time only executing speculated instructions until c is reached. The improved efficiency of strategy 3 over that of strategy 2 comes at the cost of slightly more complex exception repair hardware.

When a branch commits and is mispredicted, the exception repair hardware must perform three functions: 1) determine whether an exception has occurred during the execution of a speculated instruction, 2) if an exception has occurred, determine the PC value of the excepting instruction, and 3) determine which changes to the system state must be undone. Functions 1 and 2 are similar to error detection and location in multiple instruction rollback. Function 3 is similar to on-path hazard resolution in multiple instruction rollback [11, 12, 18]. On-path hazards assume that after rollback the initial instruction sequence from the faulty instruction to the instruction where the error was detected is repeated.

Figure 7 illustrates the speculation of a group of



Figure 7: Exception repair.

instructions and re-execution strategy 3. The load instruction traps, but the exception is not handled until the branch instruction commits to the fall-through path. Control is then returned to the trapping instruction. This scenario is identical to multiple instruction rollback where an error occurs during the load instruction and is detected during the branch instruction. For this example, only r_1 must be restored during rollback since r_4 and r_5 will be rewritten prior to use during re-execution. Figure 7 shows that exception repair hazards in speculative execution are the same as onpath hazards in multiple instruction rollback, and a read buffer as described in Section 2 can be used to resolve these hazards. The depth of the read buffer is the maximum distance from I_b to I_n along any backwards walk³, where I_n is a trapping instruction that was speculated above branch instruction I_b .

3.3 Schedule Reconstruction

Assumed in Figures 6 and 7 are mechanisms to identify speculative instructions, determine the PC value of excepting speculated instructions, and determine how many branches a given instruction has been speculated above. An example of the latter case is shown in Figure 6 where instructions d, i, and f, are undone if c is mispredicted; however, only i must be undone if g is mispredicted.

If the hardware had access to the original code schedule, the design of these mechanisms would be straightforward. Unfortunately, static scheduling reorders instructions at compile-time and information as to the original code schedule is lost. To enable recovery from mispredicted branches and proper handling of speculated exceptions, some information relative to the original instruction order must be present in the compiler-emitted instructions. This will be referred to as schedule reconstruction.

By limiting the flexibility of the scheduler, less information about the original schedule is required. For example, if speculation is limited to one level only (i.e., above a single branch), a single bit in the opcode field is sufficient to indicate that the instruction has been moved above the next branch [8]. The hardware would then know exactly which instruction effects to undo (i.e., the ones with this bit set). Also, removing branch hazards directly with the compiler permits general speculation with no schedule reconstruction for branch repair [9].

4 Implicit Index Schedule Reconstruction

Implicit index scheduling supports general speculation of regular and trapping instructions. The scheme was inspired by the handling of stores in the sentinel scheduling scheme [9] and was designed to exploit the unique properties of the read buffer hardware design described in Section 2. Schedule reconstruction is accomplished by marking each instruction speculated or nonspeculated by including a bit in the opcode field, and using this encoding to maintain an operand history of speculated instructions in a FIFO queue called a speculation read buffer (SRB). The SRB operates similar to a read buffer with additional provisions for exception handling.

4.1 Exception Repair Using a Speculation Read Buffer

Figure 8 shows an original code schedule and two speculative schedules, along with the contents of the SRB at the time branches I_c and I_g commit. Instructions I_d and I_f have been speculated above branch instruction I_c , and I_i has been speculated above both I_g and I_c . The encoding of speculated instructions informs the hardware that the source operands are to be saved in the SRB, along with the source operand values, corresponding register addresses, and the PC of the speculated instruction.

Speculated instructions execute normally unless they trap. If a speculated instruction traps, the exception bit in the SRB which corresponds to the trapping instruction is set and program execution continues. Subsequent instructions that use the result of the trapping instruction are allowed to execute normally.

A $chk_except(k)$ instruction is placed in the home block of each speculated instruction. Only one $chk_except(k)$ instruction is required for a home block. As the name implies, $chk_except(k)$ checks for pending exceptions. The command can simultaneously interrogate each location in the SRB by utilizing the bit field k. As shown in schedule 1 of Figure 8, $chk_except(001111)$ in I'_c checks exceptions for instructions I_d and I_d . If a checked exception bit is set, the SRB is flushed in reverse order, restoring the appropriate register and PC values. Execution can then begin with the excepting instruction.

Figure 8 illustrates several on-path hazards which are resolved by the SRB. In schedule 1, if I_i traps and the branch I_c commits to the taken path, I_i has corrupted r_2 and I_f has corrupted r_7 . Flushing the SRB up through I_i restores both registers to their values prior to the initial execution of I_i . Note that register r_6 is also corrupted but not restored by the SRB, since after rollback r_6 will be rewritten with a correct value before the corrupted value is used.

As an alternative to checking for exceptions in each home block, the exception could be handled when the exception bit reaches the bottom of the SRB. This is similar to the reorder buffer used in dynamic scheduling [14] and eliminates the cost of the $chk_except(k)$ command, however, increases the exception handling

³A walk is a sequence of edge traversals in a graph where the edges visited can be repeated [19].



Figure 8: Exception repair using a speculation read buffer (SRB).

latency which can impact performance depending on the frequency of exceptions.

Implicit index scheduling derives its name from the ability of the compiler to locate a particular register value within the SRB. This is possible only if the dynamically occurring history of speculated instructions is deterministic at branch boundaries. Superblocks guarantee this by ensuring that the sole entry into the superblock is at the header and by limiting speculation to within the superblock. For standard blocks, bookkeeping code [7] can be used to ensure this deterministic behavior.

4.2 Branch Repair Using a Speculation Read Buffer

As described in Section 2, branch repair can be handled by resolving branch hazards with the compiler. Branch hazard resolution in multiple instruction rollback can be assisted by the read buffer when covering on-path hazards are present, reducing the performance cost of variable renaming [11, 12]. In a similar fashion, the SRB can assist in branch repair. Figure 9 shows the original code schedule and the two speculative schedules of Figure 8. For this example, it is assumed that r_2 , r_3 , r_6 , and r_7 are elements in both $live_in(I_j)$ and $live_in(I_k)$.

As shown in schedule 1, if branch instruction I_e commits to the taken path, r_2 , r_6 , and r_7 , which were modified in I_i , I_d , and I_f , respectively, must be restored. If instead, I_e commits to the fall-through path and I_g commits to the taken path, only r_2 must be restored. Registers r_2 and r_7 are rollback hazards that result from exception repair; therefore, the SRB contains their unmodified values. By including a flush(k)command at the target of I_e and I_g , the SRB can be used to restore r_2 and/or r_7 given a misprediction of I_e or I_g .

The flush(k) command selectively flushes the appropriate register values given a branch misprediction. For example, in schedule 2 of Figure 9, if I_e is predicted correctly and I_g is mispredicted, the SRB is flushed in reverse order up through I_i , restoring $value(r_2)$ from I_i but not restoring $value(r_7)$ from I_f . Since speculation is always from the most probable branch path, the flush(k) command is always placed on the most improbable branch path, minimizing the performance penalty. Not all branch hazards are resolved by the presence of on-path hazards. These remaining hazards can be resolved with compiler transformations.

5 SRB Flush Penalty

The examples of Section 4 demonstrate that compiler-assisted multiple instruction rollback can be applied to both branch repair and exception repair in a speculative execution architecture. The flush penalty of the read buffer is not a key concern in multiple instruction rollback applications since instruction faults are typically very rare. In application to exception repair in speculative execution, the SRB flush penalty is also not a major concern due to the infrequency of exceptions involving speculated instructions. However, in application to branch repair, the SRB flush penalty could produce significant performance impacts. Studies of branch behavior show a conditional branch frequency of 11% to 17% [20]. Static branch prediction methods result in branch mispredictions in the range of 5% to 15%. This results in a branch repair frequency as high as 2.5%. Assuming a CPI (clock cycles per instruction) rate of one and an average SRB flush penalty of ten cycles, the performance overhead of the flush mechanism would reach 22.5%. This indicates the importance of minimizing the amount of redundant data stored in the SRB so that the flush penalty is reduced.

Recently, a technique was proposed to reduce the amount of redundant data in a read buffer so that the read buffer size could be reduced [12, 13]. A similar technique can be used to assure that only the data required for branch and exception repair is stored in the SRB. In the implicit index scheme of Section 4, a bit indicating whether an instruction is speculated is added to the opcode field. By expanded this field to two bits, operand storage requirements can be specified. Figure 10 shows the reduced contents of the SRB given schedule 1 of Figure 9. In the modified scheme, only the first read of r_7 must be maintained. Register rs is not required since it was not modified. The improved scheme also eliminates blank spaces in the SRB. For this example, the misprediction of I_c in schedule 1 of Figure 9 results in four less variables to flush.

The coding of the two speculation bits would be as follows: 00) no save required, 01) save operand 1, 10) save operand 2, and 11) save both operands. If neither operand of a speculated instruction has be saved in the SRB, the instruction is not marked as speculated. This is not a problem for branch repair: however, if such an instruction traps, the hardware would have no way of knowing not to handle the exception immediately. There would also be no entry in the SRB for the exception bit or for the corresponding PC value. One solution to the problem would be to add another bit to



Figure 9: Branch repair using a speculation read buffer (SRB).



SRB Contents





Figure 11: Instrumentation code placement.

the opcode field which marks speculated trapping instructions. A better solution is to code all speculated trapping instructions which have no operands to save as 01. This will indicate that exception handling is to be delayed and cause a reservation of an entry in the SRB, and also will slightly increase the flush penalty during branch repairs.

6 Performance Evaluation

6.1 Evaluation Methodology

In this section, results of a read buffer flush penalty evaluation are presented. The instrumentation code segments of Figure 11 call a branch error procedure which performs the following functions:

1. Update the read buffer model.

- 2. Force actual branch errors during program execution, allowing execution to proceed along an incorrect path for a controlled number of instructions.
- 3. Terminate execution along the incorrect path and restore the required system state from the simulated read buffer.
- 4. Measure the resulting flush cycles during the branch repair.
- 5. Begin execution along the correct path until the next branch is encountered.

An example instrumentation code segment is shown in Figure 12. Parameters, such as operand saving information, current PC, branch fall-though PC, and branch target PC values, are passed by the instrumentation code to the branch error procedure. An additional miscellaneous parameter contains instruction type and information used for debugging.

Figure 13 gives a high level flow of operation for the branch error procedure. When a branch instruction in the original application program is encountered, an *arm_branch* flag is set. Prior to the execution of the next application instruction, the *arm_branch* flag is checked, and if set, the branch decision made by the application program is set aside. The branch is then predicted by the branch prediction model. Four models are used in the evaluation: 1) predict taken, 2) predict not taken, 3) dynamic prediction, and 4) static prediction from profiling information. The dynamic prediction model is derived from a two bit counter branch target buffer (BTB) design [21] and is the only model that requires updating with each prediction outcome.

After the branch is predicted, the prediction is checked against the actual branch path taken by the application program. If the prediction was correct, execution proceeds normally. If the prediction was incorrect, the correct branch path is loaded into the recovery queue along with a branch error detection (BED) latency, and the predicted path is loaded into the PC. The BED latency indicates how long the execution of instructions is to continue along the incorrect path. The branch error time_out flag is set when the BED latency is reached. When a branch error is detected, the register file state is repaired using the read buffer contents. The PC value of the correct branch path is obtained from the recovery queue. During branch error rollback recovery, the number of cycles required to flush the read buffer during branch repair is recorded.



Figure 12: Instrumentation code sequences.



- PC program counter
- GPRF general purpose register file
- RB read buffer
- BPM branch prediction model

Figure 13: Branch error procedure operation.

..

Table 1. Application programs	Table	1: App	lication	programs
-------------------------------	-------	--------	----------	----------

Program	Static Size	Description
QUEEN	148	eight-queen program
WC	181	UNIX utility
QSORT	252	quick sort algorithm
CMP	262	UNIX utility
GREP	907	UNIX utility
PUZZLE	932	simple game
COMPRESS	1826	UNIX utility
LEX	6856	lexical analyzer
YACC	8099	parser-generator
CCCP	8775	preprocessor for gnu C compiler

It is assumed for this evaluation that two read buffer entries can be flushed in a single cycle. This corresponds to a split-cycle-save assumption of the general purpose register file [12]. Performance overhead due to read buffer flushes (% increase) is computed as

 $Flush_OH = 100 * \frac{flush_cycles}{total_cycles}$

All instructions are assumed to require one cycle for execution. This assumption is conservative since the MIPS processor used for the evaluation requires two cycles for a load. The additional cycles would increase the *total_cycles* and thereby reduce the observed performance overhead. In addition to accurately measuring flush costs, the evaluation verifies the operation of the read buffer and its ability to restore the appropriate system state over a wide range of applications.

The instrumentation insertion transformation operates on the s-code emitted by the MIPS code generator of the IMPACT C compiler [3]. The transformation determines which operands require saving in the read buffer and inserts calls to the *initialization*, branch error, and summary procedures. The resulting s-code modules are then compiled and run on a DECstation 3100. For the evaluation, BED latencies from 1 to 10 were used. Table 1 lists the ten application programs evaluated. Static Size is the number of assembly instructions emitted by the code generator, not including the library routines and other fixed overhead.

6.2 Evaluation Results

Experimental measurements of read buffer flush overhead (*Flush OH*) for various BED latencies are shown in Figures 14 through 23. The four branch







Figure 15: Flush penalty: WC.

prediction strategies used for the evaluation are: 1) predict taken (P_Taken) , 2) predict not taken (P_N_Taken) , 3) dynamic prediction based on a branch target buffer (Dyn_Pred) , and 4) static branch prediction using profiling data $(Prof_Pred)$.

Flush costs were closely related to branch prediction accuracies, i.e., the more often a branch was mispredicted, the more often flush costs were incurred. In a speculative execution architecture, branch prediction inaccuracies result in performance impacts in addition to the impacts from the branch repair scheme. Branch misprediction increases the base run time of an application by permitting speculative execution of unproductive instructions. Increased levels of speculation increase the performance impacts associated with branch prediction inaccuracies. Only the performance impacts associated read buffer flushes are shown in Figures 14 through 23.



Figure 16: Flush penalty: COMPRESS.



Figure 17: Flush penalty: CMP.

For nine of the ten applications, P_N -Taken was significantly more accurate or marginally more accurate in predicting branch outcomes than P_T -Taken. For QSORT, P_T -Taken was significantly more accurate than P_N -Taken. This result demonstrates that in a speculative execution architecture, it is difficult to guarantee optimal performance across a range of applications given a choice between predict-taken and predict-not-taken branch prediction strategies.

For all but one application, *Prof_Pred* was more accurate than either *P_Taken* or *P_N_Taken*. For CMP, *Prof_Pred*, *P_N_Taken*, and *Dyn_Pred* were nearly perfect in their prediction of branch outcomes. *Prof_Pred* marginally outperformed *Dyn_Pred* in all applications except LEX.

The purpose of measuring read buffer flush costs given the recovery from injected branch errors is to establish the viability of using a read buffer design







Figure 19: Flush penalty: QSORT.

for branch repair for speculative execution. Although in such a speculative schedule only static prediction strategies would be applicable, the *Dyn_Pred* model was included to better assess how varying branch prediction strategies impact flush costs. Overall, the accuracy of *Dyn_Pred* fell between *P_Taken/P_N_Taken* and *Prof_Pred*.

Over the ten applications studied, read buffer flush overhead ranged from 49.91% for the P_Taken strategy in CCCP to .01% for the P_N_Taken strategy for CMP given a BED of ten. It can be seen from Figures 14 through 23 that a good branch prediction strategy is key to a low read buffer flush cost. The results show that given a static branch prediction strategy using profiling data, an average BED of ten produces flush costs no greater than 14.8% and an average flush cost of 8.1% across the ten applications studied. This performance overhead is comparable to the overhead



Figure 20: Flush penalty: GREP.



Figure 21: Flush penalty: LEX.

expected from a delayed write buffer scheme with a maximum allowable BED of ten [15]. Given a maximum BED of ten and an average BED of less than ten, the flush costs of the read buffer would be less than that of a delayed write buffer, since a delayed write buffer is designed for a worst-case BED and the flush penalty of a read buffer is based on the average BED. The observed flush costs are small in comparison to the substantial performance gain of speculated architectures (8-10].

The BED for a given branch in this evaluation corresponds to the number of instructions moved above a branch in a speculative schedule. The results of the evaluation indicate that if the average number of instructions speculated above a given branch is ≤ 10 , then the read buffer becomes a viable approach to handling branch repair.



Figure 22: Flush penalty: YACC.



Figure 23: Flush penalty: CCCP.

7 Summary

Speculative execution has been shown to be an effective method to create additional instruction level parallelism in general applications. Speculating instructions above branches requires schemes to handle mispredicted branches and speculated instructions that trap.

This paper showed that branch hazards resulting from branch mispredictions in speculative execution are similar to branch hazards in multiple instruction rollback developed for processor error recovery. It was shown that compiler techniques previously developed for error recovery can be used as an effective branch repair scheme in a speculative execution architecture. It was also shown that data hazards that result in rollback due to exception repair are similar to on-path hazards suggesting a read buffer approach to exception repair.

Implicit index scheduling was introduced to exploit the unique characteristics of rollback recovery using a read buffer approach. The read buffer design was extended to include PC values to aid in rollback from excepting speculated instructions.

Read buffer flush penalties were measured by injecting branch errors into ten target applications and measuring the flush cycles required to recover from the branch errors using a simulated read buffer. It was shown that with a static branch prediction strategy using profiling data, flush costs under 15% are achievable. The results of these evaluations indicate that compiler-assisted multiple instruction rollback is viable for branch and exception repair in a speculative execution architecture.

8 Acknowledgements

The authors wish to thank Shyh-Kwei Chen and C.-C. Jim Li for their help with the compiler aspects of this paper. We would like to thank Scott Mahlke, William Chen, and John Christopher Gyllenhaal for their excellent technical suggestions and assistance with the IMPACT C compiler. Finally, we express our thanks to Janak Patel for his contributions to this research.

References

- R. P. Colwell, R. P. Nix, J. O'Donnell, D. B. Papworth, and P. K. Rodman, "A VLIW Architecture for a Trace Scheduling Compiler," in Proc. 2nd Int. Conf. Architecture Support Programming Languages and Operating Syst., pp. 105-111, Oct. 1987.
- [2] J. C. Dehnert, P. Y. Hsu, and J. P. Bratt, "Overlapped Loop Support in the Cydra 5," in Proc. Srd Int. Conf. Architecture Support Programming Languages and Operating Syst., pp. 26-38, April 1989.
- [3] P. Chang, W. Chen, N. Warter, and W.-M. W. Hwu, "IMPACT: An Architecture Framework for Multiple-Instruction-Issue Processors," in Proc. 18th Annu. Symp. Comput. Architecture, pp. 266-275, May 1991.
- [4] B. R. Rau and C. D. Glaeser, "Some Scheduling Techniques and an Easily Schedulable Horizontal Architecture for High Performance Scientific

Computing," in Proc. 20th Annu. Workshop Microprogramming Microarchitecture, pp. 183-198, Oct. 1981.

- [5] M. S. Lam, "Software Pipelining: An Effective Scheduling Technique for VLIW Machines," in Proc. ACM SIGPLAN 1988 Conf. Programming Language Design Implementation, pp. 318-328, June 1988.
- [6] A. Aiken and A. Nicolau, "Optimal Loop Parallelization," in Proc. ACM SIGPLAN 1988 Conf. Programming Language Design Implementation, pp. 308-317, June 1988.
- [7] J. A. Fisher, "Trace Scheduling: A Technique for Global Microcode Compaction," *IEEE Trans. Comput.*, vol. c-30, no. 7, pp. 478-490, July 1981.
- [8] M. D. Smith, M. S. Lam, and M. Horowitz, "Boosting Beyond Scalar Scheduling in a Superscalar Processor," in Proc. 17th Annu. Symp. Comput. Architecture, pp. 344-354, May 1990.
- [9] S. A. Mahlke, W. Y. Chen, W.-M. W. Hwu, B. R. Rao, and M. S. Schlansker, "Sentinel Scheduling for VLIW and Superscalar Processors," in Proc. 5th Int. Conf. Architecture Support Programming Languages and Operating Syst., pp. 238-247, Oct. 1992.
- [10] M. D. Smith, M. A. Horowitz, and M. S. Lam, "Efficient Superscalar Performance Through Boosting," in Proc. 5th Int. Conf. Architecture Support Programming Languages and Operating Syst., pp. 248-259, Oct. 1992.
- [11] N. J. Alewine, S.-K. Chen, C.-C. J. Li, W. K. Fuchs, and W.-M. W. Hwu, "Branch Recovery with Compiler-Assisted Multiple Instruction Retry," in Proc. 22th. Int. Symp. Fault-Tolerant Comput., pp. 66-73, July 1992.
- [12] N. J. Alewine, Compiler-assisted Multiple Instruction Rollback Recovery using a Read Buffer. PhD thesis, Tech. Rep. CRHC-93-06, University of Illinois at Urbana-Champaign, 1993.
- [13] N. J. Alewine, S.-K. Chen, W. K. Fuchs, and W.-M. W. Hwu, "Compiler-assisted Multiple Instruction Rollback Recovery using a Read Buffer," Tech. Rep. CRHC-93-11, Coordinated Science Laboratory, University of Illinois, May 1993.
- [14] J. E. Smith and A. R. Pleszkun, "Implementing Precise Interrupts in Pipelined Processors," *IEEE Trans. Comput.*, vol. 37, pp. 562-573, May 1988.

- [15] Y. Tamir and M. Tremblay, "High-Performance Fault-Tolerant VLSI Systems Using Micro Rollback," *IEEE Trans. Comput.*, vol. 39, pp. 548-554, Apr. 1990.
- [16] A. V. Aho, R. Sethi, and J. D. Ullman, Compilers: Principles, Techniques, and Tools. Reading, MA: Addison-Wesley, 1986.
- [17] M. Johnson, Superscalar Microprocessor Design. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1991.
- [18] C.-C. J. Li, S.-K. Chen, W. K. Fuchs, and W.-M. W. Hwu, "Compiler-Assisted Multiple Instruction Retry," Tech. Rep. CRHC-91-31, Coordinated Science Laboratory, University of Illinois, May 1991.
- [19] J. A. Bondy and U. Murty, Graph Theory with Applications. London, England: Macmillan Press Ltd., 1979.
- [20] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach. San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1990.
- [21] J. K. Lee and A. J. Smith, "Branch Prediction Strategies and Branch Target Buffer Design," *Computer*, vol. 17, no. 1, pp. 6-22, Jan. 1984.