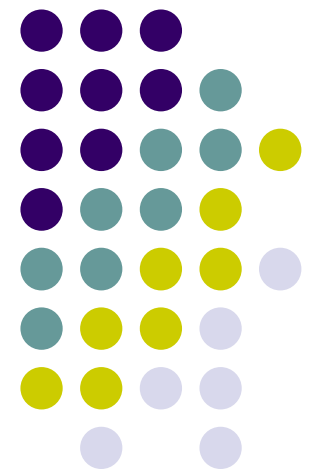
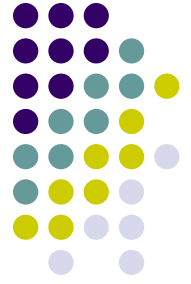


Using Reconfigurable Logic to Simulate Computer Systems

Derek Chiou
University of Texas at Austin
Electrical and Computer Engineering



Supported in part by DOE, NSF, SRC,
Bluespec, Intel, Xilinx, IBM, and Freescale

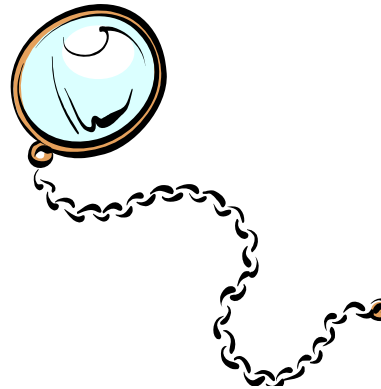


Rand's Talk

- Cycle-poor simulators



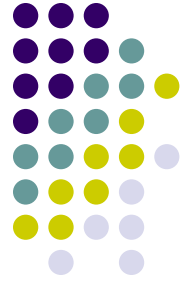
- Cycle-rich hardware





Fast, Accurate Simulator?

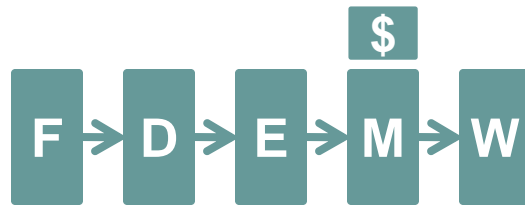
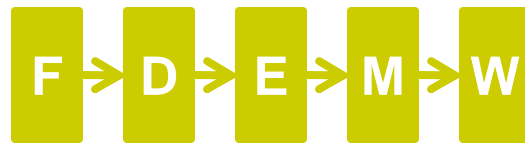
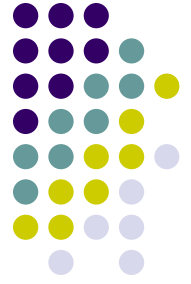
- Fast simulator is easy
 - Several that are within a factor of 10 of reality
 - No performance/power prediction
- Accurate inherently slow, lots of details
 - Intel/AMD arch simulators 100K-1M slower than real
 - RTL simulators 1B slower than real
- Only way to have fast, accurate simulator is ***aggressive (10K+)*** parallelization
 - Multicore not sufficient
 - FPGAs?



How to Apply FPGAs?

- Emulation/Prototyping
 - Port RTL to FPGAs
 - Issues
 - Late
 - RTL not designed for FPGA
 - Not that fast (10K slower than hardware)
 - Lots of FPGA resources
- Port software simulators to FPGAs
- New simulator architectures for FPGAs
- C-to-gates doesn't work well for simulators

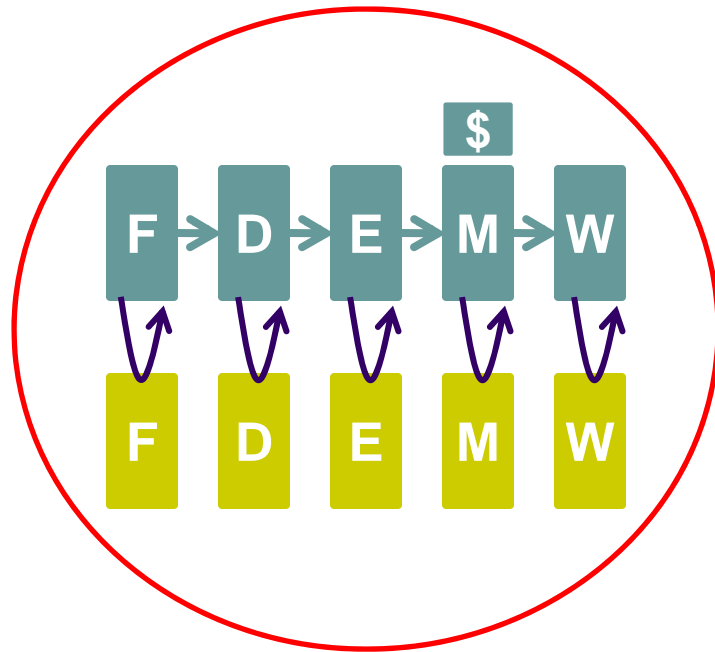
Functional/Timing Partitioned Simulators



- Simulator partitioned into two
 - Partitions change roughly independently, reducing cost of change
- Functional model
 - Executes functionality of target system
 - E.g., ISA, peripheral functionality
 - Implement x86 once, reuse many times
- Timing model
 - Models time of target system
 - E.g., caches, pipelining



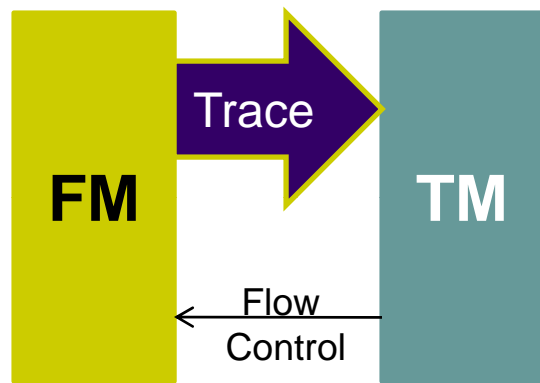
Timing-Directed



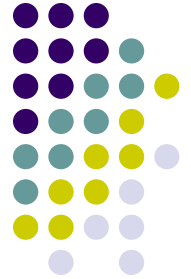
- Timing model calls functional model at appropriate target time
- Ensures functionality performed ***in the correct order***
- Requires very frequent communication

- For FPGA implementation, both functional and timing need to be implemented on FPGA for performance
- Intel/MIT HAsim, Berkeley RAMP-Gold

Another Way?

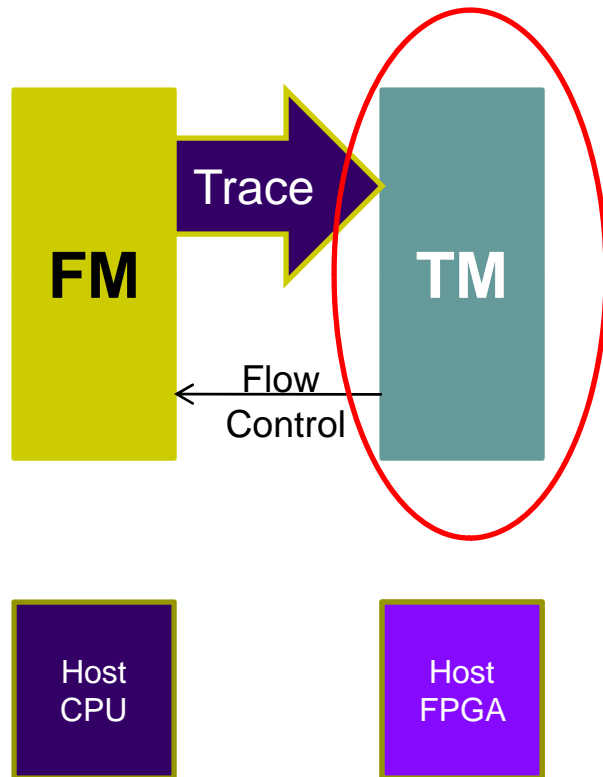


- Difficult to implement full ISA on FPGA
 - Intel has implemented x86 roughly 3 times on FPGA with full RTL
- Software functional models very fast, very complete
 - Boot full operating systems, run unmodified code
- Functional first: Functional model (FM) executes, feeds trace to timing model (TM)
 - All information that TM needs (opcode, register names, addresses, etc.) can be computed by FM



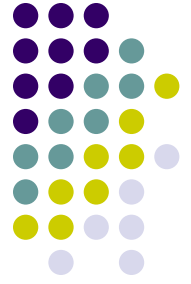


Parallelize Functional First



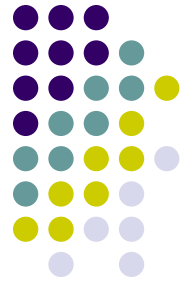
- Parallelize between FM/TM
 - Minimized round-trip communication between FM & TM (just flow control) maximizes parallel performance
- Parallelize TM by implementing in FPGA
 - TM bottleneck, small, lots of fine grain communication
 - FPGA is excellent at fine-grained communication needed by timing model
 - FM on CPU runs very fast
- Result is a fast simulator
 - 10MIPS-100MIPS to simulate single core target

But, Functional First Is Inaccurate



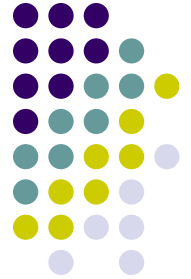
- FM executes first without timing information
- Functional accuracy dependent on timing
 - Shared memory accesses highly dependent on timing of loads
 - FM executes load/store in different order than TM
 - Branch mispredictions and resolution highly dependent on timing
 - Wrong path instructions pollute pipeline, caches
- Timing dependent on accurate functionality
- ***Inaccurate even for uncore target***

Example: Dekker's Algorithm



Core0	Core1
10: $M[0] = 1$	20: $M[1] = 1$
11: $R0 = M[1]$	21: $R0 = M[0]$
12: $BR_{R0 \neq 0} 15$	22: $BR_{R0 \neq 0} 25$
13: $M[CS] = 0$	23: $M[CS] = 1$
14: BR END	24: BR END
15: $M[0] = 0$	25: $M[1] = 0$

Functional First: Core 0 Gets Lock

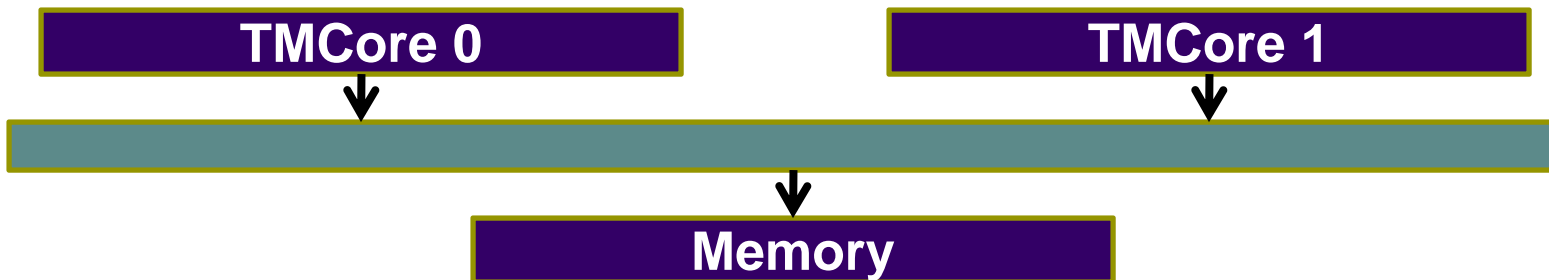


10	M[0]=1	←
11	R0=M[1]	
12	BR _{I=0} 15	
13	M[CS]=10	
14	JMP END	
15	M[0]=0	

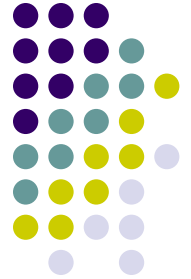
20	M[1]=1	←
21	R0=M[0]	
22	BR _{I=0} 25	
23	M[CS]=20	
24	JMP END	
25	M[1]=0	

P0.3		13: M[CS]=
P0.2		12: BR _{I=0} 15
P0.1		11: R0=M[1]
P0.0		10: M[0]=

P1.2		22: BR _{I=0} 25
P1.1		21: R1=M[0]
P1.0		20: M[1]=



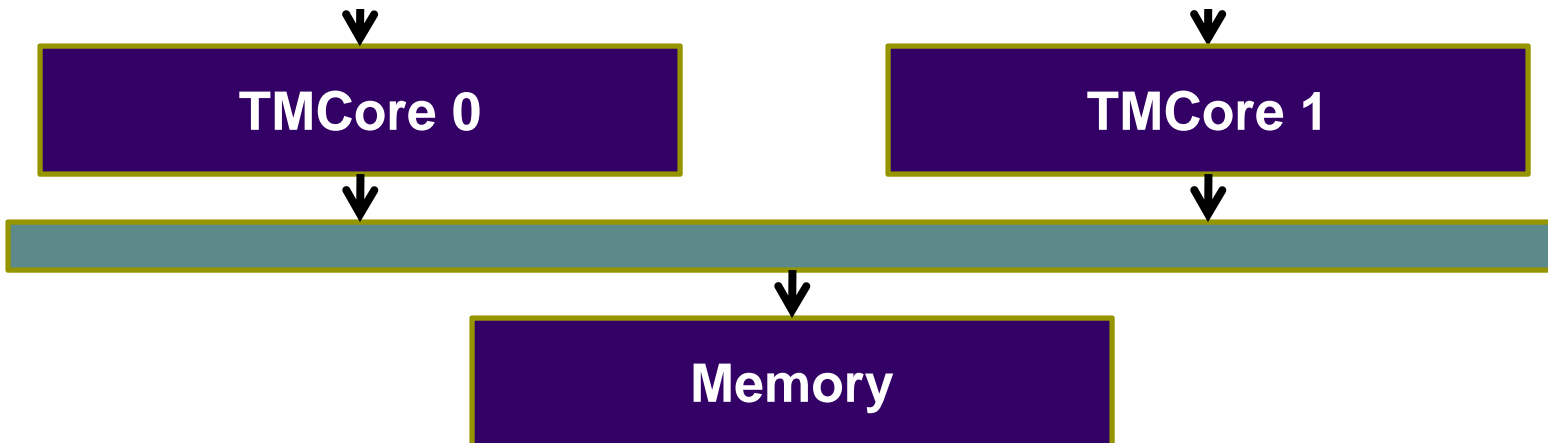
What if on Target, Core1 Gets Lock?

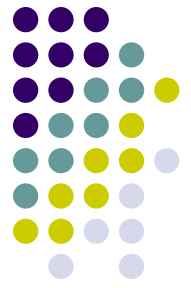


Functional Trace Is Target Incorrect!

- How to detect?
- How to correct?

Traditional solution is to avoid functional-first when accuracy important
- timing-directed, execute-in-execute

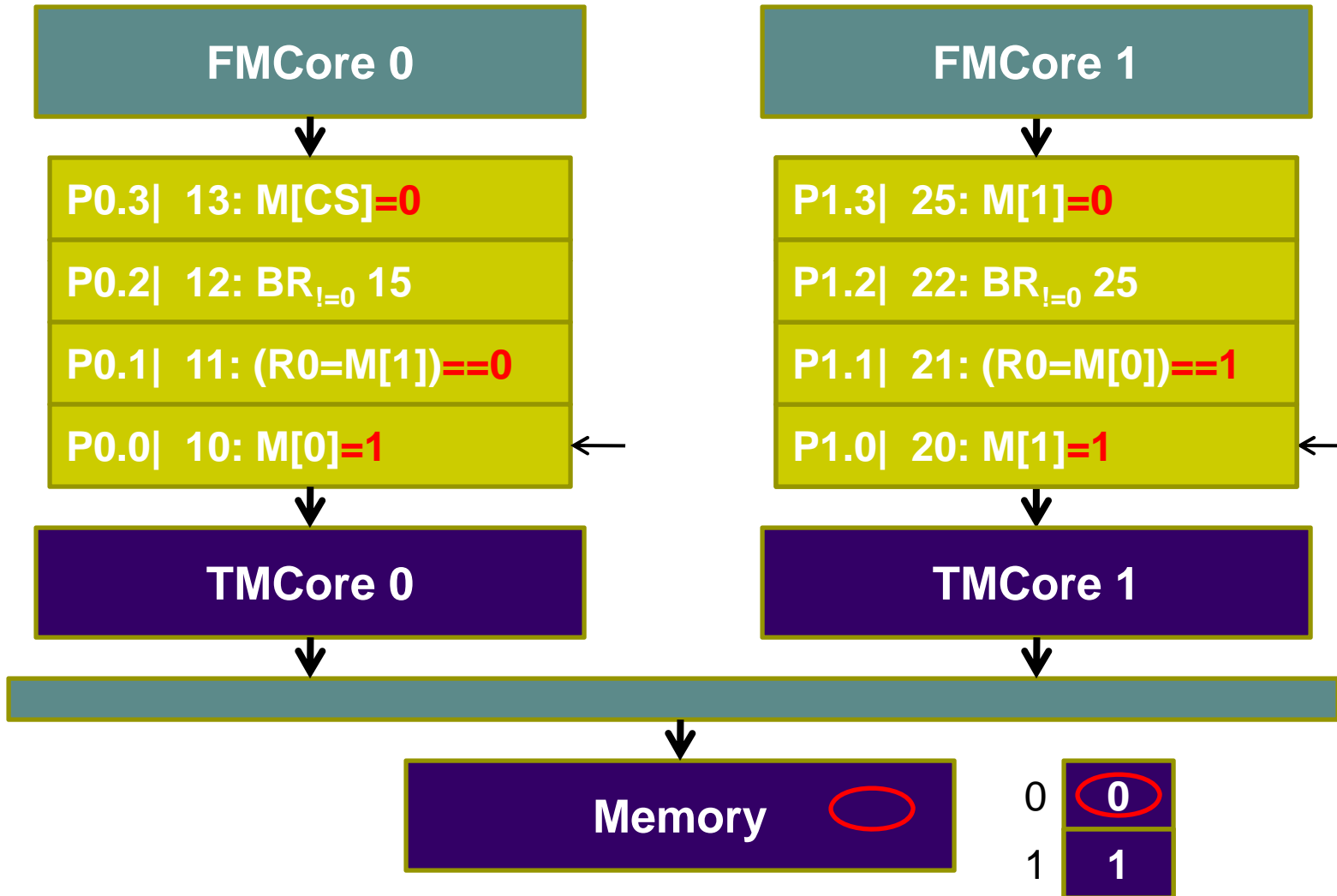


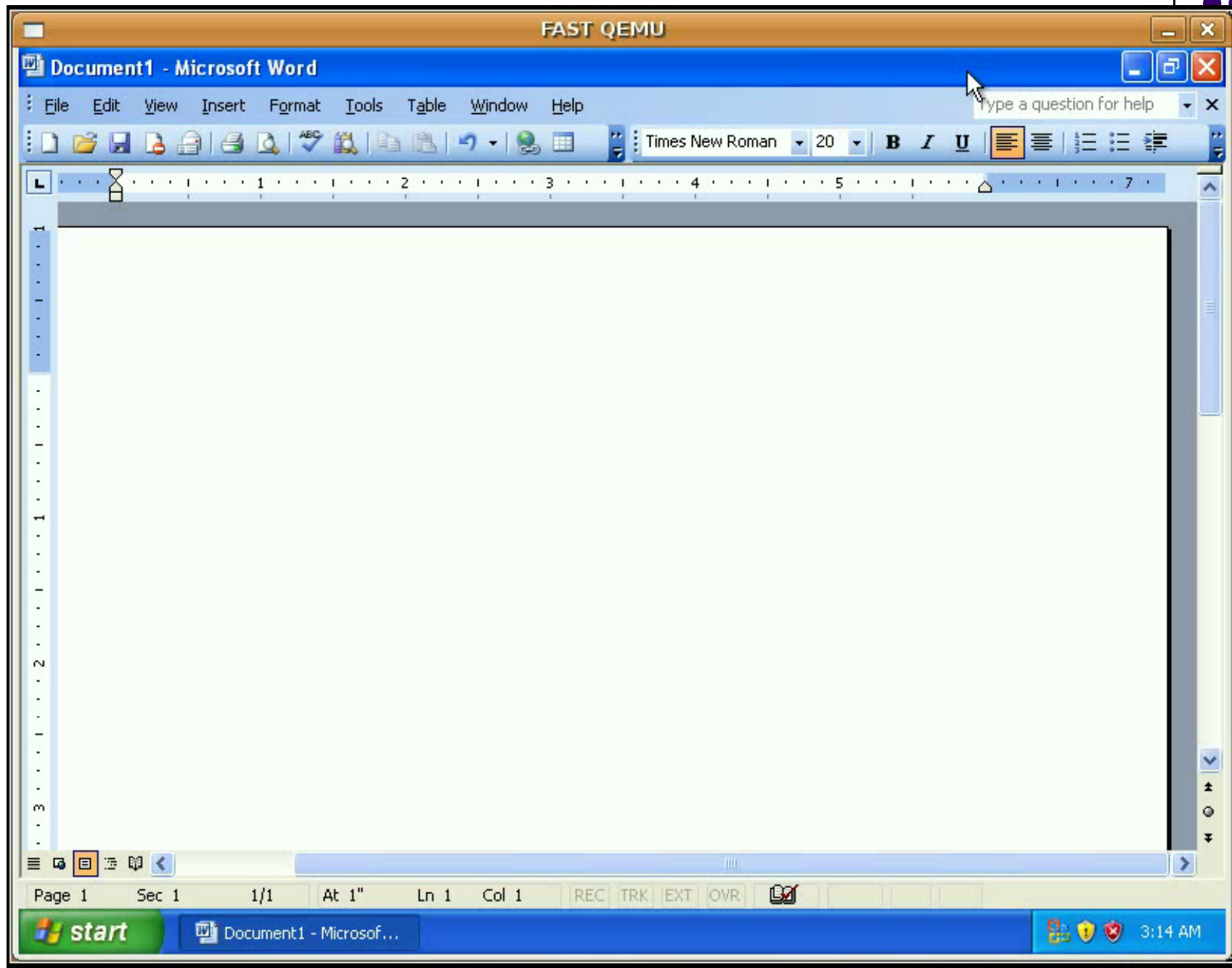


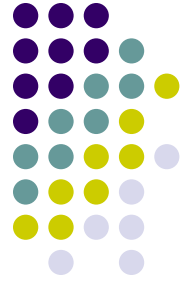
Solution: Speculative Functional First

- ***Assume we have target-correct values***
 - Easy to get functional load/store values (hard to get exec order)
 - **Load values (and store values) provided in functional trace**
 - Compare target load value with functional load value to detect
 - Have target correct value to correct when necessary
 - Rollback functional model, change value, replay, **regenerating trace including addresses, stored values**
 - Differs from traditional parallelization techniques (e.g., PDES) that use order
- How do we get target-correct values?
 - Target Memory Oracle (TMO) models target-correct memory values
 - TMO read at target time with target-correct address
 - TMO written at target correct time with target-correct address, data
 - Won't execute in timing model until address/data values correct
- ***Speculatively*** execute functionally, produce functional values, correct when wrong

Speculative Functional First and Oracles







Conclusions

- Fast computer system simulators would be really useful for architecture, verification, debug
- FPGA-based simulators can help achieve speed
 - Several ways to attack the problem
- Could be used for hardware/software codesign, performance/power tuning
- Current work:
 - Accurate power models at same speed
 - 5% cycle-by-cycle RMS for ARM A8, Freescale superscalar core (FPL 2010)
 - Automatically transforming simulator description to implementation (DAC 2011)
- Biggest Issue
 - FPGA design still hard, need to simplify for faster development



Acknowledgements

- Students
 - Hari Angepat (FM-MP)
 - Ram Chakravarthy (parallelizing FM-MP)
 - Dam Sunwoo (FM-UP, Power), now at ARM Research
 - Nikhil Patil (TM, tools, FAST2Imp)
 - Gene Wu (FM, Power)
 - Yi Yuan (TM, Reliability)
 - Dan Zhang (TMO)
 - Xiaoyu Ma (MP TM)
 - Maysam Lavasani (Magilla)
- Funding, Equipment
 - DOE, NSF, SRC
 - Intel, Xilinx, IBM, Freescale
- Software, tools
 - Bluespec, Xilinx
- Open-source full system simulators
 - QEMU, Bochs