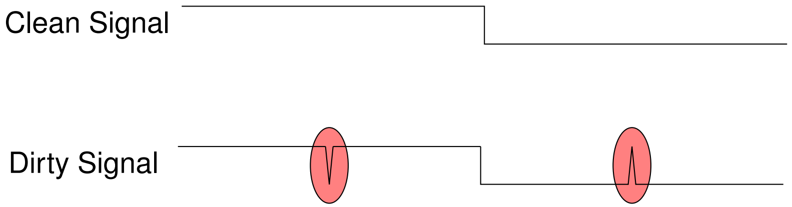
Run-time Soft Error Injection and Testing of a Microprocessor using FPGAs

A. Spilla[1], I. Polian[2], J. Müller[1], **M. Lewis**[1], V. Tomashevich[2],
B. Becker[1], and W. Burgard[1]

[1]Albert-Ludwigs-University
Georges-Köhler-Allee 51
79110 Freiburg i. Br., Germany

[2]University of Passau
Innstraße 43
94032 Passau, Germany

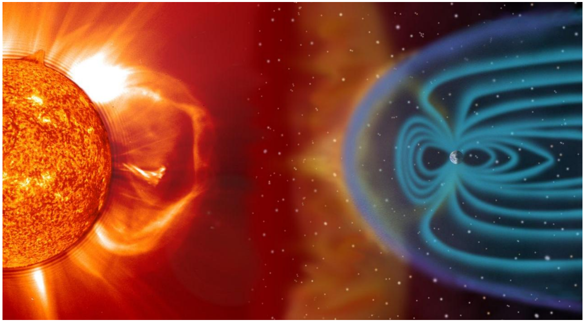
Motivation



What are Soft Errors (Transient Faults)?

- One time error in circuit
  - i.e. not a permanent structural or functional failure
- Caused by external radiation
  - i.e.cosmic rays, α-particles, ...
  - Solar flares, EM interference, radiation from packaging or other system components (e.g. Intel's 16 KBit DRAMs from 1978)
- Important when incorrect value is saved in memory (latch)

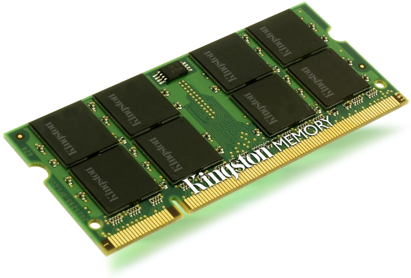
Motivation



What are Soft Errors (Transient Faults)?

- One time error in circuit
  - i.e. not a permanent structural or functional failure
- Caused by external radiation
  - i.e.cosmic rays, α-particles, ...
  - Solar flares, EM interference, radiation from packaging or other system components (e.g. Intel's 16 KBit DRAMs from 1978)
- Important when incorrect value is saved in memory (latch)

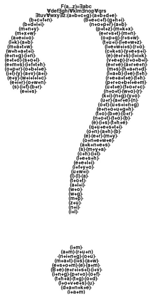
Motivation



What are Soft Errors (Transient Faults)?

- One time error in circuit
  - i.e. not a permanent structural or functional failure
- Caused by external radiation
  - i.e.cosmic rays, α-particles, ...
  - Solar flares, EM interference, radiation from packaging or other system components (e.g. Intel's 16 KBit DRAMs from 1978)
- Important when incorrect value is saved in memory (latch)

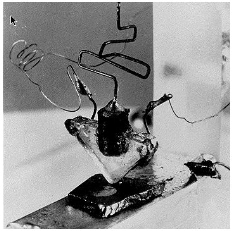
Motivation

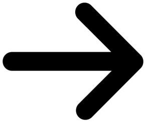


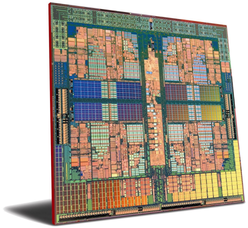
Why is it important to study them?

- Circuits are getting smaller, faster, more dense (more susceptible)
- Aviation/Space applications (safety critical, more radiation)
- On larger systems they become quite frequent

Motivation



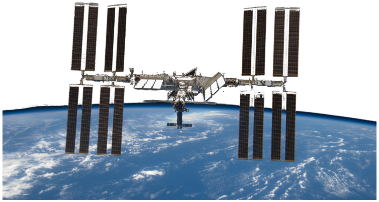




Why is it important to study them?

- Circuits are getting smaller, faster, more dense (more susceptible)
- Aviation/Space applications (safety critical, more radiation)
- On larger systems they become quite frequent

Motivation



Why is it important to study them?

- Circuits are getting smaller, faster, more dense (more susceptible)
- Aviation/Space applications (safety critical, more radiation)
- On larger systems they become quite frequent

Motivation

Why is it important to study them?

- Circuits are getting smaller, faster, more dense (more susceptible)
- Aviation/Space applications (safety critical, more radiation)
- On larger systems they become quite frequent

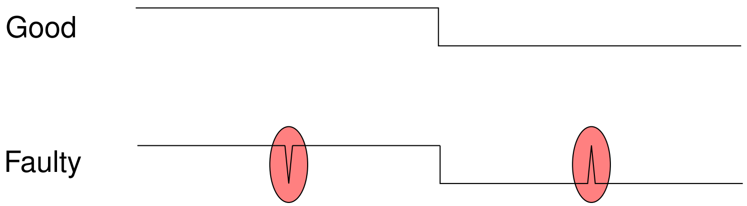
Outline

Possible Soft Error Testing Methods

Three main approaches:

- Physical testing
  - Need finished chip and radiation source (for accelerated testing)
- Software injection/emulation
  - Can do it during development, but either slow or restricted
- **Hardware accelerated simulation**
  - Can be done before production, its fast, but can be limited

Possible Soft Error Testing Methods



Three main approaches:

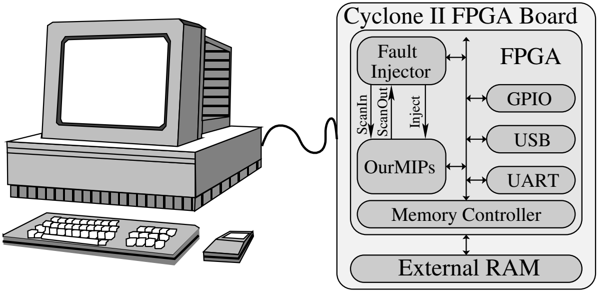
- Physical testing
  - Need finished chip and radiation source (for accelerated testing)
- Software injection/emulation
  - Can do it during development, but either slow or restricted
- **Hardware accelerated simulation**
  - Can be done before production, its fast, but can be limited

Possible Soft Error Testing Methods

Three main approaches:

- Physical testing
  - Need finished chip and radiation source (for accelerated testing)
- Software injection/emulation
  - Can do it during development, but either slow or restricted
- **Hardware accelerated simulation**
  - Can be done before production, its fast, but can be limited

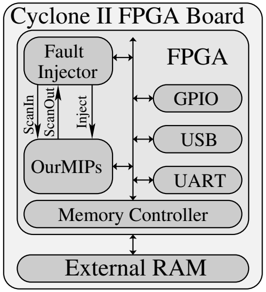
Our Approach - Overview



Goals of the current design:

- Implement and test an entire SoC
  - Current implemention uses a MIPS based SoC
- Run real applications in real time
  - Need GPIO, lots of Memory, RTOS support, ...
- Fast simulation, flexible, and cheap

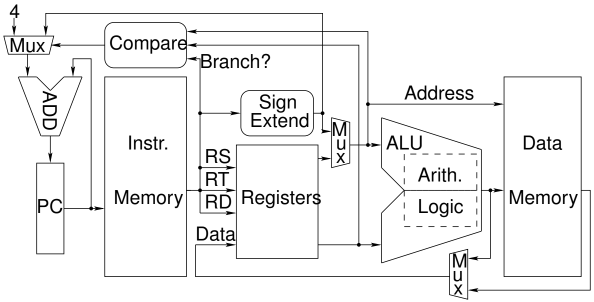
Our Approach - Architecture



The SoPC contains:

- MIP32 compatible processor (OurMIPS)
- A programmable fault injector (time, location, number, ...)
- Multiple I/O components (GPIO, USB, UART, ...)
- External memory (**512KB SRAM**, 8MB SDRAM)

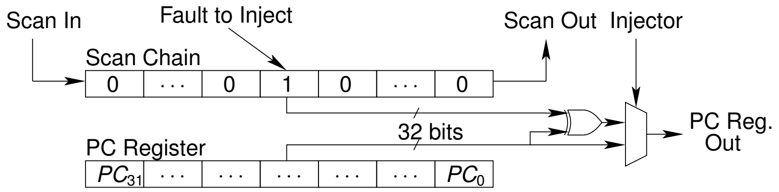
Our Approach - Architecture



Our current version of OurMips supports:

- 32 bit MIPSv1 instruction set
- Hardware interrupt support
- Hardware multiply/divide units
- Co-processors (e.g. FP unit) and GPIO support

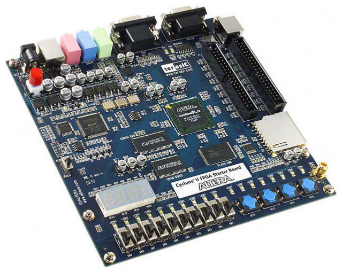
Our Approach - Architecture



To inject soft errors we currently use shadow registers:

- OurMIPs contains two copies of every register
- Fault location(s) can be shifted in without pausing the processor
- Fault location(s) can be random or specifically selected
- Can easily change from input based to output based fault injection

Our Approach - FPGA Based

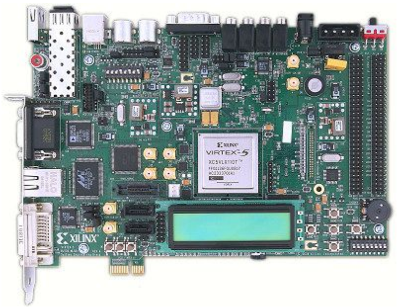


We currently use an Altera FPGA starter board:

- Connected to multiple external memory and I/O signals
- Excellent debugging capabilities (Signal Tap, Logic Analyzers, ...)
- Can handle our full SoC (≈40% Utilization, 10's MHz)
- Cheap (≈100€), and getting more capable every year

Scalable: From embedded processors to Sun's OpenSPARC T2

Our Approach - FPGA Based



We currently use an Altera FPGA starter board:

- Connected to multiple external memory and I/O signals
- Excellent debugging capabilities (Signal Tap, Logic Analyzers, ...)
- Can handle our full SoC ($\approx$40% Utilization, 10's MHz)
- Cheap ($\approx$100€), and getting more capable every year

Scalable: From embedded processors to Sun's OpenSPARC T2

Our Approach - Flow

Our total flow is as follows:

1. Program the FPGA with SoPC
2. Send the fault injection parameters to the system
3. Send the software application to the system
4. Initialize and start the fault injector
5. Execute the software application
6. Monitor the application and compare to error-free run
7. Repeat X times

Test Application

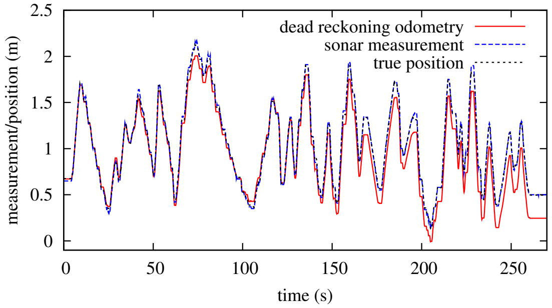
We used a Pioneer P3-DX mobile robot:

- Scalable and allows us to use multiple sensors

For the results:

- Compared dead reckoning odometry and sonar measurements to ground-truth data (laser range finder) using a Kalman filter

Test Application



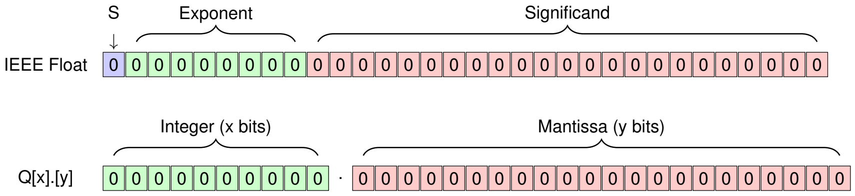
We used a Pioneer P3-DX mobile robot:

- Scalable and allows us to use multiple sensors

For the results:

- Compared dead reckoning odometry and sonar measurements to ground-truth data (laser range finder) using a Kalman filter

Test Application - Software Hardening



IEEE Floating point numbers:

- 1 bit sign, 8 bit exponent, 23(24) bit significand

Our floating point representations:

- 16 and 32 bit (fx16, fx32)
- Variable size integer and mantissa parts
- Range is more than adequate for our application

Results

		10 Transient Faults/s			100 Transient Faults/s		
Q-Format	Type	Time (s)	RMSE (m)	# TSF	Time (s)	RMSE (m)	# TSF
IEEE	float	1.731	$9.83E30$	8	1.658	2.07E36	22
Q[4].[28]	fx32	0.441	0.021	3	0.441	0.022	14
Q[7].[25]	fx32	0.487	0.020	2	0.488	0.024	20
Q[10].[22]	fx32	0.558	0.019	2	0.580	0.121	21
Q[13].[19]	fx32	0.595	0.049	2	0.593	1.794	14
Q[4].[12]	fx16	0.349	0.022	1	0.343	0.022	19
Q[7].[9]	fx16	0.300	0.026	0	0.297	0.026	14
Q[10].[6]	fx16	0.257	0.036	1	0.258	0.106	17
Q[13].[3]	fx16	0.230	0.140	0	0.300	0.140	18

Results:

- Total of 1,800 simulation runs performed (100/configuration)
  - Simulation are fast (< 2 seconds per run)
- IEEE FP calculations are very susceptible to transient faults
- Custom FP representations reduce calculation errors drastically
- Total system failures are rare (unless transient fault count $\ggg 0$)

Results

		10 Transient Faults/s			100 Transient Faults/s		
Q-Format	Type	Time (s)	RMSE (m)	# TSF	Time (s)	RMSE (m)	# TSF
IEEE	float	1.731	$9.83E30$	8	1.658	2.07E36	22
Q[4].[28]	fx32	0.441	0.021	3	0.441	0.022	14
Q[7].[25]	fx32	0.487	0.020	2	0.488	0.024	20
Q[10].[22]	fx32	0.558	0.019	2	0.580	0.121	21
Q[13].[19]	fx32	0.595	0.049	2	0.593	1.794	14
Q[4].[12]	fx16	0.349	0.022	1	0.343	0.022	19
Q[7].[9]	fx16	0.300	0.026	0	0.297	0.026	14
Q[10].[6]	fx16	0.257	0.036	1	0.258	0.106	17
Q[13].[3]	fx16	0.230	0.140	0	0.300	0.140	18

Results:

- Total of 1,800 simulation runs performed (100/configuration)
  - Simulation are fast (< 2 seconds per run)
- IEEE FP calculations are very susceptible to transient faults
- Custom FP representations reduce calculation errors drastically
- Total system failures are rare (unless transient fault count $\ggg 0$)

Results

		10 Transient Faults/s			100 Transient Faults/s		
Q-Format	Type	Time (s)	RMSE (m)	# TSF	Time (s)	RMSE (m)	# TSF
IEEE	float	1.731	$9.83E30$	8	1.658	2.07E36	22
Q[4].[28]	fx32	0.441	0.021	3	0.441	0.022	14
Q[7].[25]	fx32	0.487	0.020	2	0.488	0.024	20
Q[10].[22]	fx32	0.558	0.019	2	0.580	0.121	21
Q[13].[19]	fx32	0.595	0.049	2	0.593	1.794	14
Q[4].[12]	fx16	0.349	0.022	1	0.343	0.022	19
Q[7].[9]	fx16	0.300	0.026	0	0.297	0.026	14
Q[10].[6]	fx16	0.257	0.036	1	0.258	0.106	17
Q[13].[3]	fx16	0.230	0.140	0	0.300	0.140	18

Results:

- Total of 1,800 simulation runs performed (100/configuration)
    - Simulation are fast (< 2 seconds per run)
- IEEE FP calculations are very susceptible to transient faults
- Custom FP representations reduce calculation errors drastically
- Total system failures are rare (unless transient fault count $\ggg$ 0)

Results

		10 Transient Faults/s			100 Transient Faults/s		
Q-Format	Type	Time (s)	RMSE (m)	# TSF	Time (s)	RMSE (m)	# TSF
IEEE	float	1.731	$9.83E30$	8	1.658	2.07E36	22
Q[4].[28]	fx32	0.441	0.021	3	0.441	0.022	14
Q[7].[25]	fx32	0.487	0.020	2	0.488	0.024	20
Q[10].[22]	fx32	0.558	0.019	2	0.580	0.121	21
Q[13].[19]	fx32	0.595	0.049	2	0.593	1.794	14
Q[4].[12]	fx16	0.349	0.022	1	0.343	0.022	19
Q[7].[9]	fx16	0.300	0.026	0	0.297	0.026	14
Q[10].[6]	fx16	0.257	0.036	1	0.258	0.106	17
Q[13].[3]	fx16	0.230	0.140	0	0.300	0.140	18

Results:

- Total of 1,800 simulation runs performed (100/configuration)
  - Simulation are fast (< 2 seconds per run)
- IEEE FP calculations are very susceptible to transient faults
- Custom FP representations reduce calculation errors drastically
- Total system failures are rare (unless transient fault count $\ggg$ 0)

Continuing work

Generate data from use in a real application

- A ground controlled or autonomous blimp (an UAV)

Filter the IMU (inertial measurement unit) data in real time

- i.e. the data of 3 gyro., 3 accel. and 3 magneto.

Measure/monitor the effects on the RTOS

- Is real time always real time?

Continuing work

Generate data from use in a real application

- A ground controlled or autonomous blimp (an UAV)

Filter the IMU (inertial measurement unit) data in real time

- i.e. the data of 3 gyro., 3 accel. and 3 magneto.

Measure/monitor the effects on the RTOS

- Is real time always real time?

Continuing work

Generate data from use in a real application

- A ground controlled or autonomous blimp (an UAV)

Filter the IMU (inertial measurement unit) data in real time

- i.e. the data of 3 gyro., 3 accel. and 3 magneto.

Measure/monitor the effects on the RTOS

- Is real time always real time?

Conclusion

- Introduced an new architecture to simulate soft errors
    - Fast, flexible, accurate, and cheap
- Architecture allows for real time fault injection
    - Can be used as a simulator, or directly in live test applications
- Provides easy access and insight with debugging tools
    - Every latch can be monitored with embedded logic analyzer
- Overall goal is to increase soft error performance and chip quality
    - Designers can compare software/hardware hardening approaches

Conclusion

- Introduced an new architecture to simulate soft errors
  - Fast, flexible, accurate, and cheap
- Architecture allows for real time fault injection
  - Can be used as a simulator, or directly in live test applications
- Provides easy access and insight with debugging tools
  - Every latch can be monitored with embedded logic analyzer
- Overall goal is to increase soft error performance and chip quality
  - Designers can compare software/hardware hardening approaches

Conclusion

- Introduced an new architecture to simulate soft errors
  - Fast, flexible, accurate, and cheap
- Architecture allows for real time fault injection
  - Can be used as a simulator, or directly in live test applications
- Provides easy access and insight with debugging tools
  - Every latch can be monitored with embedded logic analyzer
- Overall goal is to increase soft error performance and chip quality
  - Designers can compare software/hardware hardening approaches

Conclusion

- Introduced an new architecture to simulate soft errors
  - Fast, flexible, accurate, and cheap
- Architecture allows for real time fault injection
  - Can be used as a simulator, or directly in live test applications
- Provides easy access and insight with debugging tools
  - Every latch can be monitored with embedded logic analyzer
- Overall goal is to increase soft error performance and chip quality
  - Designers can compare software/hardware hardening approaches