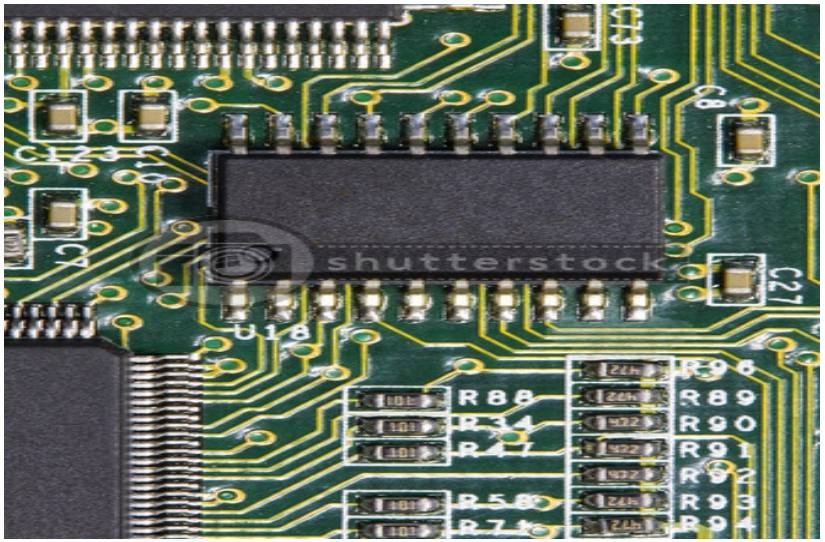
UNC CHARLOTTE

Evaluating the Fault Tolerance Capabilities of Embedded Systems via BDM



M. Rebaudengo, M. Sonza Reorda
Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Italy

- **Fault tolerant system**
  - System which is redundant to faults so that output is not affected
- **Why do we need fault tolerant systems?**
  - Vast increase in role of embedded systems in day to day applications, critical systems
- **How to verify the fault tolernace of embedded systems?**
  - By injecting faults and cross checking the output
- **Fault Injection**
  - Simulation based, Software based, Hardware based

- **Contribution of this paper is to exploit some features in recent microcontrollers for software based fault injection in embedded boards**

Background Debugging Mode [BDM]

- Case study for behavior of target board with MC68332 in presence of faults
- BDM has many plus points over other software based fault injectors
- One of the special mode of operation in Motorola microcontrollers

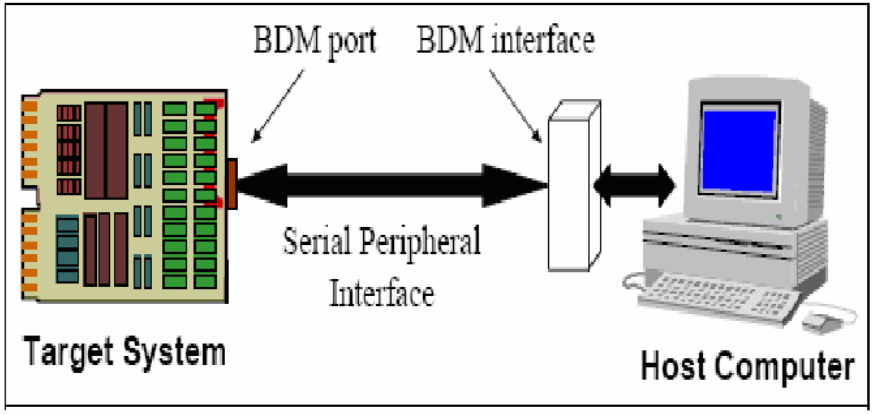


Fig. 1: BDM environment.

- Debugger is implemented in CPU microcode
- Allows host processor to access memory, registers, I/O data
- BDM port shares pins with other development features and when enabled functions as a synchronous serial port

Fault Injection Environment Architecture

- Minimum intrusiveness achieved by deploying target application code on target system while entire FIM on host machine
- **Fault Injection Manager [FIM]:** Takes one fault from the fault list, schedules its injection time and loads environment on target system

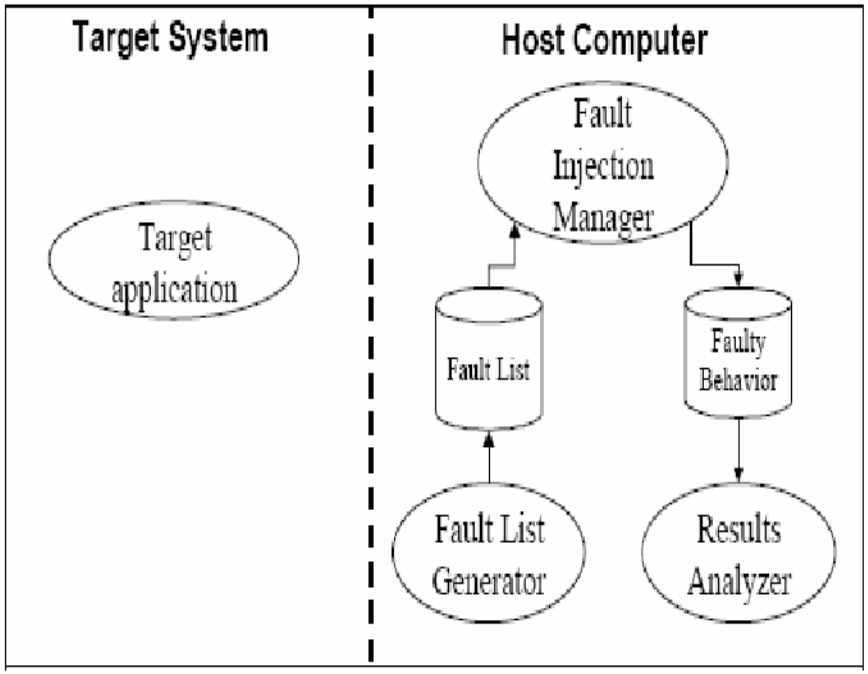


Fig. 2: Fault Injection system architecture.

UNC CHARLOTTE

Fault Injection Environment Architecture

```
void fault_injection_manager()
{                    /* Experiment Control Loop */
        for(every fault fi in the fault list)
        {
                    target_system_initialization(fi);
                    spawn(target_application);
                    inject_and_observe(fi);
         }
         return();
}
```

– All operations are with respect to BDM commands

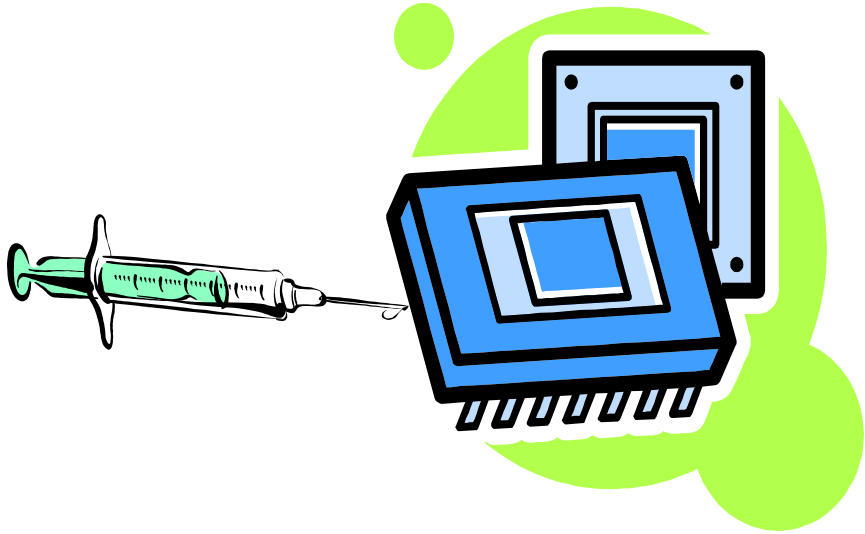
- **Target System Initialization**
  - Prepare data area and download the program
  - Fault Injection Data set Up: Fault is picked up from list and breakpoint is set in code where fault is to be injected
- **Inject and observe**
  - Monitors the system, injects faults and observes the system behavior

Fault Detection and Recovery

- Recovery from fault effects is achieved by modifying Error Detection Exception [EDE] procedures
- Return address from exception stack frame is modified so as to return to host processor and give the error message
- FIM reads error message, classifies the fault and compares it with fault free output
- Categories of Faulty system:- Fail Silent, Detected by some error detection mechanism, Fail Silent Violation and Time Out Violation
- Time Out check is done using watchdog set to twice the time requied for normal operation

Fault Model and List Generation

- Fault is injected between any two instructions, so its effect is easily reproduced
- Fault is identified by: Fault location, Injection Time
- Injection Time: Given in terms of instruction address or number of instruction repetition
- Inject and observe routine injects fault at the ‘n’th activation of an instruction by means of BDM command which modfies the memory location or user register as determined by fault location



Fault Model and List Generation

- Fault list generated before injection
  - Fault free execution time and instruction set is traced
  - Random fault list is created in terms of fault location and injection time
  - Fault list is converted in format required by BDM
- Fault collapsing
  - Can be used to optimize the fault list
- Two classes of Error Detection Mechanism [EDM]
  - Hardware
  - Software

Fault Analysis and Fault Coverage

- Benchmark evaluation for Bubble Sort, Parser and Dhrystone
  - Set of 1000 randomly generated faults were injected in data memory, code memory and microcontroller registers

	a	*b*	*c*
	Fault-free Execution [ms]	3,000 faults injection [s]	Ratio c=b/(a*3,000)
Bubble Sort	2.6	753	96.53
Parser	9.4	1,995	70.75
Dhrystone	20.6	4,950	80.09

Table 1: time requirements.

Program	*Fault Location*	*Fault Effects*				*Total*
		FS	*D*	*FSV*	*TO*	
Bubble sort	data	257	2	741	0	1,000
	code	703	102	118	77	1,000
	registers	846	86	50	18	1,000
Parser	data	810	187	2	1	1,000
	code	406	222	359	33	1,000
	registers	836	77	83	4	1,000
Dhrystone	data	991	1	2	6	1,000
	code	664	71	233	32	1,000
	registers	807	64	111	18	1,000

Table 2: result summary

Limitations of Fault Injection via BDM

- Microcontroller speed decreases when BDM mode is entered upon
- Serial communication between host and target introduces significant delay
- Little difficult to apply for time related characteristics of real time systems

Paper Summary

- The papers gives a good insight to understand a new perspective of debugging environment
- The fault analysis and coverage statistics helps understand the need for deployment of such methods to make fault tolerant product development process more efficient