A STUDY OF FAULT COVERAGE OF STANDARD AND WINDOWED WATCHDOG TIMERS

Ashraf M. El-Attar and Gamal Fahmy

German University in Cairo

ABSTRACT

Both standard and windowed watchdog timers were designed to detect flow faults and ensure the safe operation of the systems they supervise. This paper studies the effect of transient failures on microprocessors, and utilizes two methods to compare the fault coverage of both watchdog timers. The first method is injecting a fault while a processor is reading an image from RAM and sending it to the VGA RAM for display. This method is implemented on FPGA, and visually demonstrates the existence of fast watchdog resets which can not be detected by standard watchdog timers, and faulty resets which occur undetected within the safe window of the windowed watchdog timers. The second method is a simulation where the fault coverage for each watchdog timer system is calculated. This simulation tries to take into consideration many factors which could affect the outcome of this comparison.

Index Terms—Microprocessors, Watchdog timer, Fault coverage, Flow faults

1. INTRODUCTION

Microprocessors have become an integral part of everyday systems. The safe operation of these systems can not be taken for granted due to the fact that a malfunction may occur. Flow faults occur when a processor fetches an instruction and executes an incorrect next instruction [1, 2]. Both standard and windowed watchdog timers were designed to detect errors caused by flow faults. These faults may result from temporary hardware failures or software bugs [1, 2].

A watchdog timer is a simple timer circuit. The microprocessor is required to periodically reset the timer. If the timer is not reset within a preset time then a decision is made to recover the system and stop it from inflicting any damage [1-4]. This concurrent error detection system should satisfy the following requirements: 1) should not be complex, 2) provide good error coverage, 3) should not require major changes to the system, and 4) it should not result in high overhead to the monitored system [3].

Due to the unpredicted effect of transient failures, the standard watchdog may be reset too fast affecting its fault coverage. A watchdog with a time window helps overcome this problem by allowing the system to reset the timer only within a preset time window [1]. Yet, windowed watchdog timers are unable to detect resets which occur with their safe window. It is thus important to compare the fault coverage for both watchdog systems to identify their effectiveness and reliability. Section 2 explains the operation of the standard watchdog and the types of errors it detects. Section 3 explains the windowed watchdog timer. Section 4 demonstrates the effect of transient failures on image display, and the effect of both watchdog timers. Section 5 describes the design used to simulate and measure the fault coverage. Section 6 presents the achieved results. Finally, Section 7 concludes this paper.

2. STANDARD WATCHDOG TIMER

A standard watchdog timer in its simplest form is a monostable timer. When the timer reaches its maximum value it changes its logical state. The system must reset the timer before it reaches maturity. If the system fails to reset the timer an action is taken whether to change the state of an output or to immediately restart the system.

If the watchdog timeout is $T_1$ and the system resets the timer after a period $T_2$ then $T_2 < T_1$ [1]. The microprocessor resets the timer by fetching and executing an instruction known as ClearWDT. When an error occurs in the processor’s Program counter points to the wrong next instruction. Thus the time required for the system to reset the timer may increase or decrease depending on how far this faulty location is from the next ClearWDT, and the time remaining for the timer to reach $T_1$ as follows: 1) If the Faulty location is far from the ClearWDT instruction, then this instruction will not be executed on time, and the watchdog timer will reach $T_1$ and detect the fault. 2) If the faulty location is close to the ClearWDT instruction, where the time required for the program counter to reach the ClearWDT instruction is less than the time remaining to reach $T_1$, then the system will reset the watchdog and the fault will not be detected [1]. Case 1 is known as Slow watchdog resets and Case 2 is know as fast watchdog resets.
3. WINDOWED WATCHDOG TIMER

In order to solve the problem of fast watchdog resets, the windowed watchdog timer was introduced. This supervisory system is based on two timers instead of one. The first timer has a timeout of $T_1$ and the second timer has a timeout of $T_3$. The ClearWDT instruction must be executed in a time window of $(T_3 - T_1)$ to reset both timers, where $T_3 > T_1$ [1].

In the case of a fast reset then the ClearWDT will be executed in a time less than $T_1$. In the case of a slow reset then the ClearWDT will be executed after $T_3$. In both cases the windowed watchdog will detect the fault [1].

The only situation where the windowed watchdog may fail is if a flow fault occurs and a faulty ClearWDT is executed within the safe time window $(T_3 - T_1)$. The closer $T_1$ and $T_3$ are the less the probability that the ClearWDT is executed within the safe time window.

4. EFFECT OF FAULTS ON DISPLAYING AN IMAGE

This method uses a simple 8 bit Princeton architecture microprocessor attached to it a VGA display circuit. This system is implemented on an FPGA. An 8 color BMP format file is processed as follows: 1) the file header is removed 2) the matrix is transposed into a vector 3) the 4 bit vector is then copied to the FPGA RAM at the desired location. The 3 least significant bits represent the RGB of each pixel. The image used is 256 x 96 pixels. The microprocessor then executes the following:

```c
Ram_address = [location of first pixel]
While i < 24576
    Load ACC [Ram_address]
    Store ACC [V_RAM_address]
    Increment [Ram_address]
    Increment [V_Ram_address]
    Increment i;
    ClearWDT;
```

The rest of the program memory is filled with NOP. ClearWDT instruction is inserted at equidistant locations within the entire span of the program RAM. The VGA circuit reads the VGA RAM and displays its RGB values for each pixel. The display mode is 640x480 60 Hz. Both watchdog timers are then added to the system. A fault is injected inflicting a fast reset, and the windowed watchdog timer only was able to detect it. This is indicated as the watchdog generates a vertical line at $x = 300$ pixels. A second fault is injected inflicting a slow reset, and both watchdog systems are able to detect it. The standard watchdog generates a horizontal line at $y = 120$ pixels indicating that a fault has been detected. A third fault is generated where the faulty ClearWDT is executed within the safe window of the windowed watchdog timer and the fault is not detected.

5. SIMULATION DESIGN

The system tested is an 8 bit Princeton architecture microprocessor. The tool used for simulation is LogicWorx™ 5.0.2. The design was made starting from gate level in order to achieve a near realistic timing as each gate delay was taken into consideration. Figure 1 shows a block diagram of the system including the fault generator and the watchdog timer.
5.1. Random Fault Generator

The target of this module is to generate a random number and to inject this into the system’s Program counter. The Fault injection is done at random time in order to achieve a realistic fault pattern.

The random number generator is an 8 bit PN sequence generator. Each four bits are timed with a different clock, and both clocks are not synchronized. This gives an enhancement to the randomness. Figure 2 shows the design of the PN Sequence Generator.

5.2. Fault Injection

The random numbers generated are injected into the program counter at random periods of time. A random pulse is driven from a second PN sequence generator. This pulse controls a multiplexer whose output is connected to the Program counter. The inputs are either an incrementer or the random generator.

At all times the incrementer is selected as this is the normal operation of the system. The random number Generator is selected only when a random pulse occurs so as to create a transient failure causing a flow fault in the system. Figure 3 shows a block diagram of the fault injection module.

5.3. System operation

The microprocessor is started and is executing the following instructions.

- Load ACC 04
- Add 04
- Sub 08

This simple program is repeated to fill the entire addressing space. The ClearWDT is inserted within the program at set intervals for each test run.

While the microprocessor is in its normal operation a fault is injected at a random time causing the Program counter to point to an out of sequence next instruction. At this point a flow fault is created and a counter counts the number of injected faults. Simultaneously the watchdog timer is running and a counter records the number of times the watchdog is able to detect the injected faults.

This simulation is tested while using once the standard watchdog timer and then for the windowed watchdog timer, and the results are tabulated as shown in the next section.

6. SIMULATION RESULTS

All results of the time measured are in terms of system clock cycles. The fault probability was measured to be 0.0133. The fault coverage is calculated by means of the percentage of the faults detected from the total number of faults injected.

The following are the results for different T1 and T2 for the same program running on the same system using the standard watchdog timer. Show is the watchdog time timeout period, the ClearDWT instruction frequency of occurrence in the program, the number of injected faults, the number of detected faults, and the fault coverage.
Table 1. Standard Watchdog Timer Results

<table>
<thead>
<tr>
<th>Timeout</th>
<th>ClearWDT Frequency</th>
<th>Injected Faults</th>
<th>Detected Faults</th>
<th>Fault Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>11</td>
<td>4095</td>
<td>1595</td>
<td>38.9 %</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>4095</td>
<td>2359</td>
<td>57.6 %</td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>4095</td>
<td>1965</td>
<td>48 %</td>
</tr>
<tr>
<td>26</td>
<td>22</td>
<td>4095</td>
<td>205</td>
<td>51 %</td>
</tr>
</tbody>
</table>

The following results are for different window sizes for the same program running on the same system using the windowed watchdog timer. It shows the window size, the number of faults generated, the number of detected faults, and the fault coverage.

Table 2. Windowed Watchdog Timer Results

<table>
<thead>
<tr>
<th>Window Size</th>
<th>Injected Faults</th>
<th>Detected Faults</th>
<th>Fault Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4095</td>
<td>1595</td>
<td>99.97 %</td>
</tr>
<tr>
<td>13</td>
<td>4095</td>
<td>3151</td>
<td>76.94 %</td>
</tr>
<tr>
<td>22</td>
<td>4095</td>
<td>2985</td>
<td>72.9 %</td>
</tr>
<tr>
<td>41</td>
<td>4095</td>
<td>2323</td>
<td>56.72 %</td>
</tr>
</tbody>
</table>

7. CONCLUSION

The comparison of the watchdog timers introduced in this paper utilized 2 methods which aimed at demonstrating the real effect of transient failures on the flow of the execution of software. This is clear in the first method as the produced image is corrupted. The results of both methods show that fast resets do affect the standard watchdog, while the windowed watchdog is capable of solving the problem. Moreover, it shows that the windowed watchdog is not able to detect faults if the faulty ClearWDT is executed within the safe window. This is clear when the window size is increased as the fault coverage decreased. The results from the simulation show that it is essential to carefully choose the watchdog timer parameters in order to achieve a good fault coverage. It is also clear that the choice of watchdog for a given system must be carefully studied and referenced to the proper metrics when designing microprocessor based systems.

8. REFERENCES


