Hardware Implementation of an Invariant Observer

Marko Stanisic

Department of Informatics

Technical University of Vienna

This dissertation is submitted for the degree of

Bachelor of Science

2014

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words including appendices, bibliography, footnotes, tables and equations and has less than 150 figures.

Marko Stanisic
2014

Abstract

The Invariant Observer is an approach to implement an alternative hardware realization of the invariant operation published in the paper **Runtime Verification of Embedded Real Time Systems.** The Invariant Observer is a Runtime Verification Unit monitoring a signal ϕ from a System under Test in real time and determining whether the signal is currently in an active state and has been active during the past τ clock cycles.
If this is the case then signal ϕ is observed as being invariant within the time interval $[0,\tau]$.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of the Bachelor Thesis

In embedded real time systems it is necessary to make efforts to verify a system design. A system design can be formalized by a mathematical specification for a dynamic system model. One approach to system design verification is the deduction, that shows that the design implies the requirements.

In critical Real Time Systems (RTS) timing constraints have to be considered in the requirement engineering. Such Real Time Systems are modelled by states changing over time. Time constraints can be formulated as constraints on the duration of critical states. A real time logic should be able to specify that real time constraints. Generally it seems that two main classes of real time logic are present, explicit or implicit temporal logic.[2]

Explicit temporal logic is an expression of a time variable. The time variable can be the representation of a time interval or a variable in temporal logic. Implicit temporal logic (for example MTL - Metric Temporal Logic) is using temporal operators that constrain the extend of a state. It is based on interval temporal logic and the duration concept. Implicit temporal logic can be very useful to express before/after relations between concurrent actions. For further details [2] can be a good source of information. In runtime verification a monitor evaluates executions of a **System under Test (SUT)** [4]. The evaluation is formalised from a formal specification described in temporal logic.

For ultra critical systems it is important to meet four major requirements:

1. Functionality : cannot change target's behaviour
2. Certifiability: must avoid re-certification
3. Timing : must not interfere with the target's timing
4. Swap : must not exhaust size, weight and power tolerance

A **Runtime Verification Unit (RVU)** is a verification monitor that meets that four major requirements. As part of this requirements, the RVU must be separated from SUT. In fact it is a synthesized hardware that monitors the execution of a SUT.
The topic of my thesis "Hardware implementation of an Invariant Observer" can also be considered as a RVU, it evaluates the execution of a SUT and checks it for invariance conditions. My observer is an alternative implementation of the invariant observer INVARIANT-SYMBOL published in [4], that bypass the problem of resource limitation and make use of the significant advantages of a high parallel **Field Programmable Gate Array(FPGA)** hardware implementation. The most important difference is that my observer is not bounded to a specific τ, but the observer in [4] are bounded.This feature will be explained in the next section.

In the publication "**Real-Time Runtime Verification on Chip** " [4] the concept of a RVU and the principles of that Verification Framework are described in great detail.

A survey about the functionality of the invariant observer in the following sections.

1.2 The Invariant Observer

This section is a survey about the invariant observer and how it works. More details about the observers algorithm are presented in the next chapter.
The Invariant Observer acts like the temporal (invariant previously) operator $\boxdot_\tau \phi$ of the Metric Temporal Logic (MTL) and is of course restricted on the past time (ptMTL). The temporal operator takes an input ϕ , the calculation of a propositional formula, and evaluates if ϕ holds for the past τ execution times , including the current execution time in a discrete time setting. For example the logical consequence $e^n \models \boxdot_3 \phi$ expresses that the current execution e^n (with n as the discrete execution time, n$\in \mathbb{N}$) is true iff (if and only if) the evaluation of ϕ is true now and was also true the last $\tau = 3$ execution times. In fact the $\boxdot_\tau \phi$ is a specialization of the $\boxdot_{[0,\tau]} \phi$ ptMTL operator which inidcates the range of the invariance qualification.
Figure 1.1 and Figure 1.2 show an example for such a temporal operator.

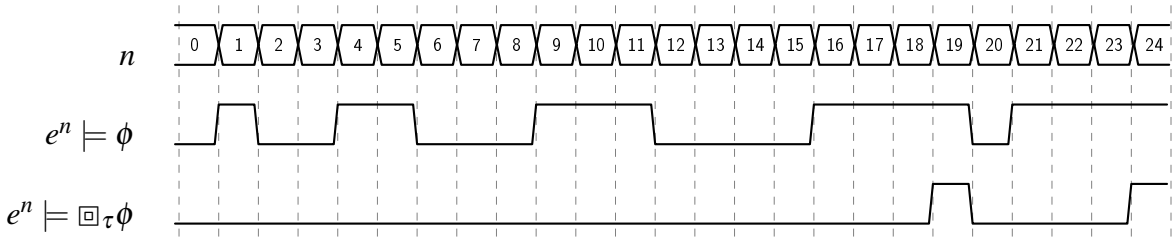


Fig. 1.1 Example for Invariant Operator $\boxdot_\tau \phi$ with τ=3

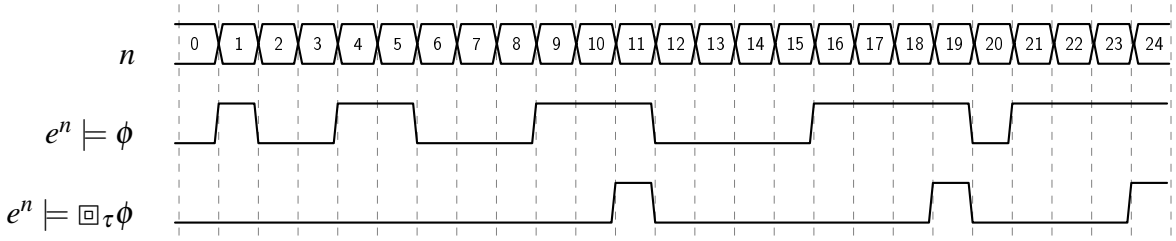


Fig. 1.2 Example for Invariant Operator $\boxdot_\tau \phi$ with τ=2

My approach of the invariant observer is based on some certain requirements. The following subsection will discuss these requirements.

1.2.1 First Requirement

To introduce the first requirement we begin with the discussion of the problem that the calculation of a atomic propositional formula ϕ could take several clock cycles (execution times). This means that a observer has to wait until the calculation of the proposition is finished. In [4] the observer needs to guarantee that it finishes evaluation of atomic propositions within a tight time bound. In our case ,if we start calculation of a propositional formula ϕ at every clock cycle and the calculation itself needs y clock cycles , than we need y observer stages to cover at every clock cycle a finished calculation. These observer stages are part of the whole observer. After y clock cycles , at every following clock cycle a calculation of ϕ will be available. At least one observer stage is ready to evaluate a calculation ϕ at any time. In other words, we are implementing temporal pipeline stages that represents components of the invariant operator and these components together are evaluating the invariance qualification of the proposition ϕ. We don't have one observer, in fact the observer is composed from several observer stages. As a matter of fact, different many clock cycles for the calculation of ϕ are possible. Propositional formulas can be composed from several other complex propositional subformulas. In some cases a subformula is waiting for the resolution of an another subformula. In [4] this balance is achieved with the restriction of the atomic proposition class by the abstract domain of logahedron.

1.3 Where does it come from?

Chapter 2

My Second Chapter

2.1 Reasonably Long Section Title

I'm going to randomly include a picture Figure 2.1.

If you have trouble viewing this document contact Krishna kks32@cam.ac.uk.

Enumeration

1. The first topic is dull
2. The second topic is duller
   (a) The first subtopic is silly
   (b) The second subtopic is stupid
3. The third topic is dullest

itemize

- The first topic is dull
- The second topic is duller
  - The first subtopic is silly
  - The second subtopic is stupid
- The third topic is dullest

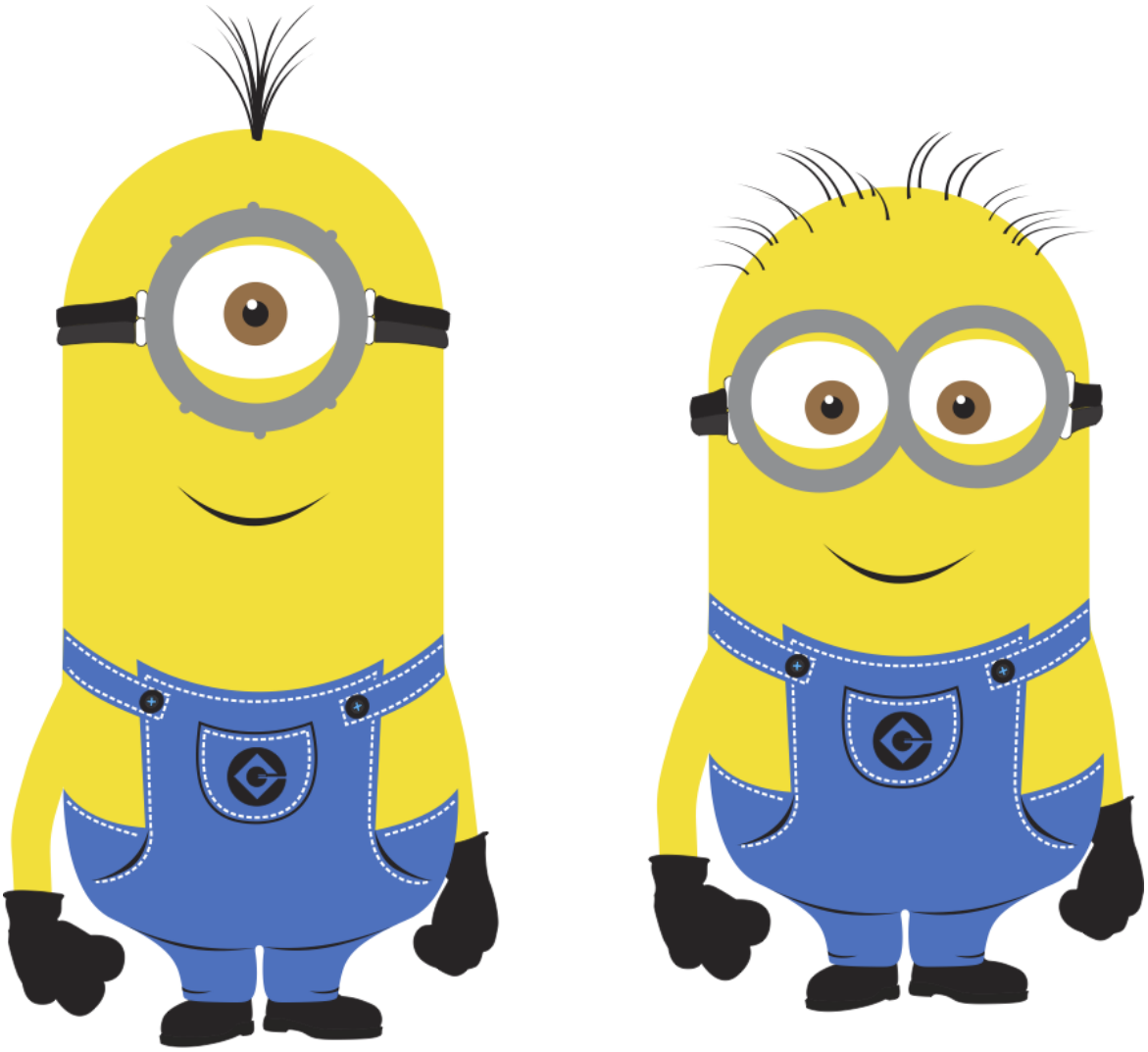


Fig. 2.1 This is just a long figure caption for the minion in Despicable Me from Pixar

description

The first topic is dull

The second topic is duller

- **The first subtopic** is silly
- **The second subtopic** is stupid

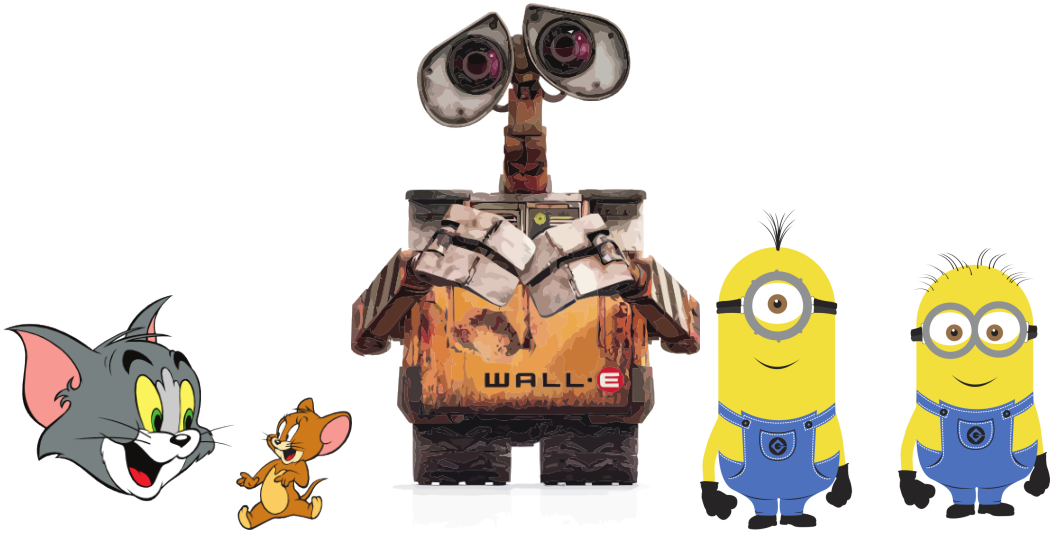
The third topic is dullest

2.2 Hidden Section

Lorem ipsum dolor sit amet, *consectetur adipiscing elit*. In magna nisi, aliquam id blandit id, congue ac est. Fusce porta consequat leo. Proin feugiat at felis vel consectetur. Ut tempus ipsum sit amet congue posuere. Nulla varius rutrum quam. Donec sed purus luctus, faucibus velit id, ultrices sapien. Cras diam purus, tincidunt eget tristique ut, egestas quis nulla. Curabitur vel iaculis lectus. Nunc nulla urna, ultrices et eleifend in, accumsan ut erat. In ut ante leo. Aenean a lacinia nisl, sit amet ullamcorper dolor. Maecenas blandit, tortor ut scelerisque congue, velit diam volutpat metus, sed vestibulum eros justo ut nulla. Etiam nec ipsum non enim luctus porta in in massa. Cras arcu urna, malesuada ut tellus ut, pellentesque mollis risus.Morbi vel tortor imperdiet arcu auctor mattis sit amet eu nisi. Nulla gravida urna vel nisl egestas varius. Aliquam posuere ante quis malesuada dignissim. Mauris ultrices tristique eros, a dignissim nisl iaculis nec. Praesent dapibus tincidunt mauris nec tempor. Curabitur et consequat nisi. Quisque viverra egestas risus, ut sodales enim blandit at. Mauris quis odio nulla. Cras euismod turpis magna, in facilisis diam congue non. Mauris faucibus nisl a orci dictum, et tempus mi cursus.

Etiam elementum tristique lacus, sit amet eleifend nibh eleifend sed [1]. Maecenas dapibu augue ut urna malesuada, non tempor nibh mollis. Donec sed sem sollicitudin, convallis velit aliquam, tincidunt diam. In eu venenatis lorem. Aliquam non augue porttitor tellus faucibus porta et nec ante. Proin sodales, libero vitae commodo sodales, dolor nisi cursus magna, non tincidunt ipsum nibh eget purus. Nam rutrum tincidunt arcu, tincidunt vulputate mi sagittis id. Proin et nisi nec orci tincidunt auctor et porta elit. Praesent eu dolor ac magna cursus euismod. Integer non dictum nunc.

[1] My footnote goes blah blah blah! ...



(a) A Tom and Jerry (b) A Wall-E (c) A Minion

Fig. 2.2 Best Animations

Subplots

I can cite Wall-E (see Fig. 2.2b) and Minions in despicable me (Fig. 2.2c) or I can cite the whole figure as Fig. 2.2

Chapter 3

My Third Chapter

3.1 First Section of the Third Chapter

And now I begin my third chapter here . . .

And now to cite some more people Ancey et al. [1], Read [3]

3.1.1 First Subsection in the First Section

. . . and some more

3.1.2 Second Subsection in the First Section

. . . and some more . . .

First subsub section in the second subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it . . .

3.1.3 Third Subsection in the First Section

. . . and some more . . .

First subsub section in the third subsection

. . . and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it and some more and some more and some more and some more

and some more and some more and some more ...

Second subsub section in the third subsection

... and some more in the first subsub section otherwise it all looks the same doesn't it? well we can add some text to it ...

3.2 Second Section of the Third Chapter

and here I write more ...

Now we can refer to the table using Table. 3.1.

Table 3.1 Table with Borders

1	2	3
4	5	6
7	8	9

References

[1] Christophe Ancey, Philippe Coussot, and Pierre Evesque. Examination of the possibility of a fluid-mechanics treatment of dense granular flows. *Mechanics of Cohesive-frictional Materials*, 1(4):385–403, 1996. URL `http://doi.wiley.com/10.1002/(SICI)1099-1484(199610)1:4<385::AID-CFM20>3.0.CO;2-0`.

[2] A.P. Ravn, H. Rischel, and K.M. Hansen. Specifying and verifying requirements of real-time systems. *Software Engineering, IEEE Transactions on*, 19(1):41–55, Jan 1993. ISSN 0098-5589. doi: 10.1109/32.210306.

[3] C. J. Read. A solution to the invariant subspace problem on the space l_1. *Bull. London Math. Soc.*, 17:305–317, 1985.

[4] Thomas Reinbacher, Matthias Függer, and Jörg Brauer. Real-time runtime verification on chip. In Shaz Qadeer and Serdar Tasiran, editors, *Runtime Verification*, volume 7687 of *Lecture Notes in Computer Science*, pages 110–125. Springer Berlin Heidelberg, 2013.

Appendix A

How to install LaTeX

Windows OS

TeXLive package - full version

1. Download the TeXLive ISO (2.2GB) from
   https://www.tug.org/texlive/

2. Download WinCDEmu (if you don't have a virtual drive) from
   http://wincdemu.sysprogs.org/download/

3. To install Windows CD Emulator follow the instructions at
   http://wincdemu.sysprogs.org/tutorials/install/

4. Right click the iso and mount it using the WinCDEmu as shown in
   http://wincdemu.sysprogs.org/tutorials/mount/

5. Open your virtual drive and run setup.pl

or

Basic MikTeX - TeX distribution

1. Download Basic-MiKTeX(32bit or 64bit) from
   http://miktex.org/download

2. Run the installer

3. To add a new package go to Start » All Programs » MikTex » Maintenance (Admin) and choose Package Manager

4. Select or search for packages to install

TexStudio - Tex Editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Run the installer

Mac OS X

MacTeX - TeX distribution

1. Download the file from
   https://www.tug.org/mactex/

2. Extract and double click to run the installer. It does the entire configuration, sit back and relax.

TexStudio - Tex Editor

1. Download TexStudio from
   http://texstudio.sourceforge.net/#downloads

2. Extract and Start

Unix/Linux

TeXLive - TeX distribution

Getting the distribution:

1. TexLive can be downloaded from
   http://www.tug.org/texlive/acquire-netinstall.html.

2. TexLive is provided by most operating system you can use (rpm,apt-get or yum) to get TexLive distributions

Installation

1. Mount the ISO file in the mnt directory

```
mount -t iso9660 -o ro,loop,noauto /your/texlive####.iso /mnt
```

2. Install wget on your OS (use rpm, apt-get or yum install)

3. Run the installer script install-tl.

```
cd /your/download/directory
./install-tl
```

4. Enter command 'i' for installation

5. Post-Installation configuration:
   http://www.tug.org/texlive/doc/texlive-en/texlive-en.html#x1-320003.4.1

6. Set the path for the directory of TexLive binaries in your .bashrc file

For 32Bit OS

For Bourne-compatible shells such as bash, and using Intel x86 GNU/Linux and a default directory setup as an example, the file to edit might be

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/i386-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

For 64Bit

```
edit $~/.bashrc file and add following lines
PATH=/usr/local/texlive/2011/bin/x86_64-linux:$PATH;
export PATH
MANPATH=/usr/local/texlive/2011/texmf/doc/man:$MANPATH;
export MANPATH
INFOPATH=/usr/local/texlive/2011/texmf/doc/info:$INFOPATH;
export INFOPATH
```

Fedora/RedHat/CENTOS:

```
sudo yum install texlive
sudo yum install psutils
```

SUSE:

```
sudo zypper install texlive
```

Debian/Ubuntu:

```
sudo apt-get install texlive texlive-latex-extra
sudo apt-get install psutils
```

Appendix B

Installing the CUED Class file

LaTeX.cls files can be accessed system-wide when they are placed in the <texmf>/tex/latex directory, where <texmf> is the root directory of the user's TeXinstallation. On systems that have a local texmf tree (<texmflocal>), which may be named "texmf-local" or "localtexmf", it may be advisable to install packages in <texmflocal>, rather than <texmf> as the contents of the former, unlike that of the latter, are preserved after the LaTeXsystem is reinstalled and/or upgraded.

It is recommended that the user create a subdirectory <texmf>/tex/latex/CUED for all CUED related LaTeXclass and package files. On some LaTeXsystems, the directory look-up tables will need to be refreshed after making additions or deletions to the system files. For TeXLive systems this is accomplished via executing "texhash" as root. MIKTeXusers can run "initexmf -u" to accomplish the same thing.

Users not willing or able to install the files system-wide can install them in their personal directories, but will then have to provide the path (full or relative) in addition to the filename when referring to them in LaTeX.