DD2452 – Lab 2
Mert Alp Taytak – taytak@kth.se

Report

1) What has actually been verified?

My implementation of certain restrictions and characteristics of the design has been verified on an abstract model of the design.

2) What has not been verified?

Whether the designed system would work on a given physical implementation has not been verified yet. Moreover, viability of the design is still in question. Because, my work on the subject matter has not been reviewed.

3) What is the accuracy of the verification?

It is highly accurate. Sheer computing power alone would be enough to brute force possibilities (after some reductions) especially since the design was purposefully restricted to use 2 bit memory spaces.

4) How could the verification be made more accurate?

We could increase the granularity of the model to allow for better simulation of real life.

5) How reliable is the verification?

Considering it is part guesswork that is unreviewed so far, it is probably not very reliable in the literal sense of the word. Otherwise, ignoring the user aspect, there can be bugs and errors with the verification software. Some piece of code could be working as intended but be plain wrong. Verifying a certain model on a certain Linux system using NuSMV 2.6 only means that your model passes the tests under those circumstances.

6) What relevant aspects/properties might be desirable to verify or to take into account in the verification (abstraction level of the model) that are not taken into account in the verification (abstracted away)?

First issue that comes into mind is that there is a likely possibility that the CPU and the I/O device would have different working frequencies. That would increase the complexity of the task. Since a lot of idling states would have to be added into the model to balance the frequency differential and increasing the number of states requires more attention to cover the illegal or unwanted transitions.

7) What is of practical importance in construction of models?

Building a model on a computer is comparatively cheap. Possibly hundreds of designs can be tested as a model before a physical implementation can be tested in matter. That is the practical benefit of construction of models.

If we are talking about practical as in ease of development, then that would be using modules as much as possible and keeping them simple. On that point, using formal definitions in a way that makes separate entities very connected makes it harder to construct clean and modular models.

8) What was most difficult?

Rewriting the state transitions to get rid of recursive definitions. In fact, I could not figure it out.

9) Did you find any bugs in your model or device driver design? In such a case, what was the bug(s)?

It appears that there exists paths for buffer overflows. Unless I made a mistake with my implementation this is a natural result of buffer itself having an address space of 2 bits and buffer lengths having a size of 2 bits as well. Hence, anything on address 0b2_11 causes an overflow if it is of length 2.