| Owner | | Document | |
| --- | --- | --- | --- |
| Name | Dimitris Zervas | Project | HY220 Lab1 |
| E-mail | csd3502@csd.uoc.gr | Filename | Lab1report.docx |

# Abstract

*The scope of this lab is to create a circuit that writes the last 2 integers received from UART to 2 7-segment displays on the Basys-2 development board.*

Table of Contents

[1 Introduction 3](#_Toc434002929)

[1.1 Background Reading 3](#_Toc434002930)

[1.2 Glossary 3](#_Toc434002931)

[1.3 Design Library 3](#_Toc434002932)

[1.4 People Involved 3](#_Toc434002933)

[2 General Description 4](#_Toc434002934)

[3 Pin List 5](#_Toc434002935)

[4 Functional Description 6](#_Toc434002936)

[4.1 Overview 6](#_Toc434002937)

[4.2 Procedural Diagram 6](#_Toc434002938)

[4.3 Detailed Description 6](#_Toc434002939)

[4.3.1 adder8 6](#_Toc434002940)

[4.3.2 reg8 7](#_Toc434002941)

[5 Implementation 8](#_Toc434002942)

[5.1 Adder8 (adder\_0) Subblock 8](#_Toc434002943)

[5.1.1 Pin List 8](#_Toc434002944)

[5.1.2 Interface Diagram 8](#_Toc434002945)

[5.1.3 Description 8](#_Toc434002946)

[5.2 reg8 (reg\_0)Subblock 9](#_Toc434002947)

[5.2.1 Pin List 9](#_Toc434002948)

[5.2.2 Interface Diagram 9](#_Toc434002949)

[5.2.3 Description 9](#_Toc434002950)

[6 Verification 10](#_Toc434002951)

[6.1 List of Tests 10](#_Toc434002952)

[6.2 Detailed Test Description 10](#_Toc434002953)

[6.2.1 Test A 10](#_Toc434002954)

[6.2.2 Test B 10](#_Toc434002955)

[6.2.3 Test C 10](#_Toc434002956)

# Introduction

## Background Reading

[1] [Digilent Basys2 Board Reference Manual.](http://www.csd.uoc.gr/~hy220/http:/www.csd.uoc.gr/~hy220/2015f/Basys2_rm.pdf/Basys2_rm.pdf)

[2] [Basys2 Master Board Constraints File.](http://www.csd.uoc.gr/~http:/www.csd.uoc.gr/~hy220/2015f/Basys2_100_250General.ucf/2015f/Basys2_100_250General.ucf)

## Glossary

## Design Library

lab1

## People Involved

| Name | E-mail address |
| --- | --- |
| Dimitris Zervas | csd3502@csd.uoc.gr |
|  |  |
|  |  |

# General Description

There are 2 blocks: the UART FSM and the 7seg FSM. The UART FSM reads from the given UART interface and manages data\_in\_ready and data\_in\_consume accordingly and saves each character to a register which is replaced every time a character arrives until a newline has arrived, so the last character remains in the register to be processed by the 7seg block.

The 7seg block consists of an FSM which drives the 7 segment displays so that the 2 first display the contents of the register that the UART block controls and the last 2 are empty.

# Pin List

| Signal | | I/O | | Description |
| --- | --- | --- | --- | --- |
| Clock & Reset | | | | |
| system\_clock | in | | System clock. | |
| system\_reset | in | | Global synchronous reset. Active high. | |
| **UART interface** | | | | |
| data\_in[7:0] | in | | Data input from terminal | |
| data\_out[7:0] | out | | Data to be displayed to terminal | |
| data\_in\_ready | in | | Signal to denote that data\_in is valid | |
| data\_in\_consume | out | | Signal to denote that data\_in was processed and ready to accept more data | |
| data\_out\_ready | out | | Signal to denote that data\_out is valid | |
| **7seg interface** | | | | |
| SevenSegAnode[3:0] | out | | This signal drives the seven segment display anodes. | |
| SevenSegData[7:0] | out | | These signals drive which segment is turned on (7 segments + the dot) | |

*Table 3.1:*

# Functional Description

## Overview

The diagram of Figure 4.1 illustrates a single instantiation of blinker block.

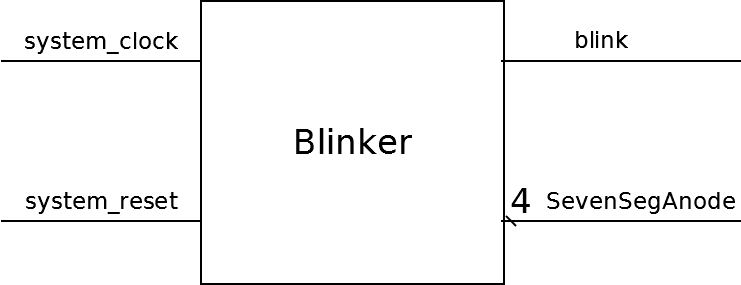


Figure 4.1:Blinker Instance

## Procedural Diagram

Figure 4.2 depicts the internal subblocks of the blinker block.

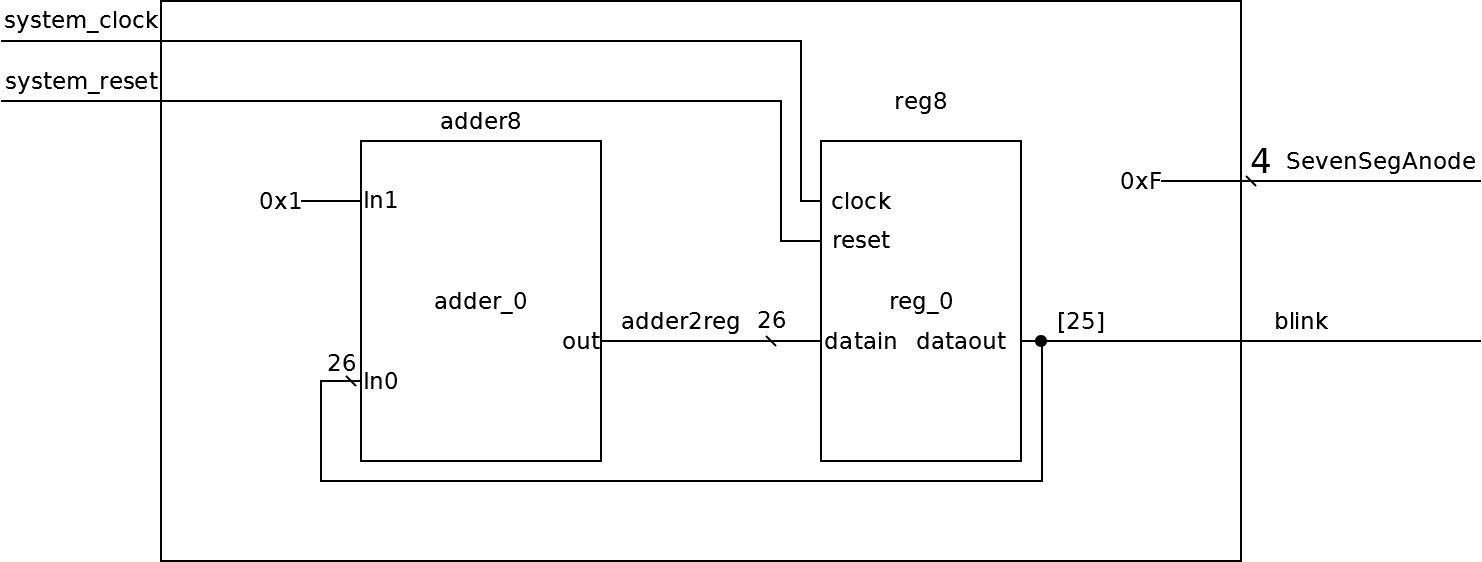


Figure 4.2:Blinker internal subblocks.

## Detailed Description

### adder8

This subblock is a simple adder that adds two 26 bit inputs. A timing diagram follows.

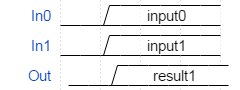


Figure 4.3: Adder8 Timing relationships

### reg8

This subblock is a 28 bit wide register with synchronous reset. A simple timing diagram follows.



Figure 4.4: Reg8 Timing relationships

# Implementation

The following paragraphs describe the implementation of the Blinker block. Figure 4.2 shows the interconnections of the subblocks included in the Blinker block.

## Adder8 (adder\_0) Subblock

### Pin List

| Signal | I/O | Description |
| --- | --- | --- |
| **Input interface** | | |
| In0[25:0] | in | Input signal. |
| In1[25:0] | in | Input signal. |
| **Output interface** | | |
| Out[25:0] | out | Output signal |

*Table 5.1: Adder8 Subblock Pin List*

### Interface Diagram

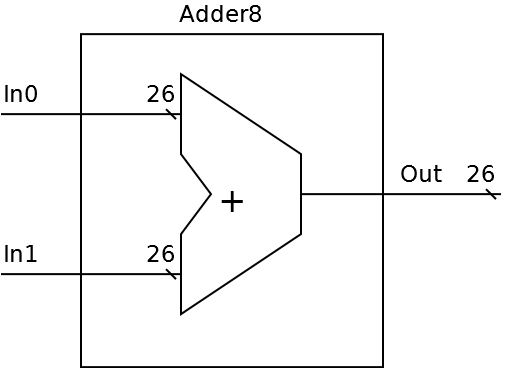


Figure 5.2: Adder8 Subblock Interface Diagram

### Description

This sublock implements an adder that adds two 26 bit values. It does the computation In0+In1. It produces the result on Out.

## reg8 (reg\_0)Subblock

### Pin List

| Signal | I/O | Description |
| --- | --- | --- |
| **Input interface** | | |
| datain[25:0] | in | Data in. |
| clock | in | Clock. |
| reset | in | Reset. Active high |
| **Output interface** | | |
| dataout[25:0] | out | Data out |

*Table 5.1: Adder8 Subblock Pin List*

### Interface Diagram

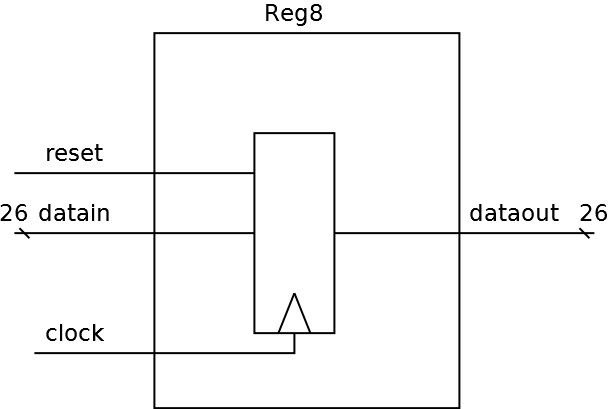


Figure 5.2: Reg8 Subblock Interface Diagram

### Description

This subblock implements a 26 bit positive edge triggered register with synchronous reset. It samples input data from datain and reset at the positive edge of the clock and if reset is high data out is zero. Else data out is data in.

# Verification

## List of Tests

* Test A – Verification of Blinker function.

| Test | Description |
| --- | --- |
| Test A -<tb.v> | Tests Blinker operation. |

*Table 6.1: List of Tests*

## Detailed Test Description

### Test A

**The goal of this test is:** Describe To simulate the Blinker and see that it blinks the led.

* Unit under test: Blinker
* input vectors: system\_clock, system reset.
* output vectors:blink, SevenSegAnode.

The purpose of this test is to check if the blinkers functions properly. We generate a clock equal of that of the FPGA (f=50Mhz T=20ns). First we reset the UUT (Unit under test) for 100 ns and then we wait 2 seconds (2 bilion nanoseconds) and check the blink output vector to see if it changes from low to high and reverse. Since we use a 26bit counter blijnk should go high at 671088740 ns (33554437 cycles) and then low at 1.342.177.380 ns (67.108.869 cycles) .