Chapter 1

Basics

This chapter contains information about basic concepts and devices used in this thesis.

1.1 Field Programmable Gate Array

A Field-Programmable Gate Array (FPGA) is a high density Integrated Circuit (IC) that is designed to be completely programmable by the customer after manufacturing (i.e. when the chip is shipped and ”in the field”). The chips are shipped completely ”blank”, meaning that there are no pre-programmed logic[1]. An FPGA can either be re-programmed using SRAM technology[2], or one-time programmed by burning antifuses. The latter method makes it less prone to soft errors when exposed to radiation.

FPGAs are composed of arrays of Configurable Logic Blocks (CLBs) surrounded by programmable routing resources and I/O pads. A CLB consists of a LookUp Table (LUT) together with a clock and simple write logic, and uses the address bus of the LUT as the function input pins and the value at the selected address as the function output. LUTs are considered fast logic, since computing a complex function only requires a single memory lookup. The LUT can be made out of an SRAM memory-block [**?**].

The FPGA (section **??**) used in this thesis is a re-programmable type. It stores the user hardware setup in an SRAM memory to configure routing and logic functions (for a permanent program storage, the FPGA can be configures to store the hardware setup in the on-board flash memory[3], which then programs the SRAM when powered on).

[1]With the exception of FPGA evaluation boards, which comes shipped with a pre-programmed hardware setup for demo purposes.

[2]Short for Static Random Access Memory, SRAM is a type of volatile memory that uses internal feedback to keep on its data.[**?**] Volatile meaning that the memory loses all the data once the power is switched off.

[3]A nonvolatile memory type known by its name because of its ability to erase memory blocks all at once ”in a flash”.[**?**] Nonvolatile meaning that the memory will keep all its data even if the power is switched off.

1.1.1 Hardware Description Language

The most common way to program an FPGA is by Hardware Description Language (HDL), with the major ones being SystemVerilog and VHDL. HDL consists of text-based expressions used to describe digital hardware and use this to further simulate and synthesize the hardware described. A hardware module is simulated by applying information at the inputs and then do a check on the corresponding outputs and verify that they behave as intended. Synthesis of hardware means transforming the HDL code into a netlist of logic and wire connections describing the hardware. This can then be compiled into an FPGA which re-wires the available internal CLBs according to the given instructions. HDLs have become more and more useful as system complexity have increased [?]. The hardware described in this thesis is done using VHDL, a strongly typed HDL that is known for its capability of describing parallel processes.

Below is a small example describing the logic of a D Flip-Flop using VHDL:

```
-- Example of a D Flip-Flop triggered on the rising edge of the clock.
library IEEE;
use IEEE.std_logic_1164.all;

entity DFF_high is
  port(
    D : in STD_LOGIC;
    Q : out STD_LOGIC;
    Qbar : out STD_LOGIC;
    RESET : in STD_LOGIC;
    CLK : in STD_LOGIC
  );
end DFF_high;

architecture rtl of DFF_high is
begin

process(CLK,RESET)
begin
  if (RESET='1') then
    Q <= '0';
    Qbar <= '1';
  elsif (CLK'event and CLK='1') then -- Can also use rising_edge(clk)
    Q <= D;
    Qbar <= not D;
  end if;
end process;

end rtl;
```

1.2 RS-232

RS-232 standard is a transmission protocol for full duplex, serial transmitting and receiving of data. The standard defines electrical characteristics and timing of signals, the ”meaning” of the signal, and also physical size and pin-out for connectors. For communications to work properly, the Data Terminal Equipment (DTE), in this case the PC; and theData Communication Equipment (DCE), in this case the FPGA, needs to agree on the same data-packet settings, i.e the baud rate, the number of data bits, any additional parity bit and the number of stop bits.

A typical RS-232 data-packet consists of a start bit, followed by 5, 7 or 8 data bits; 1 parity bit, for error checking; and 1 or 2 stop bits, indicating that the transmission is done. The start bit is typically a logical low and the stop bit(s) high (this is usually the case, but can be system dependent). The transmission line remains high until the start bit pulls it down low, and the data transfer begins until the stop bit is reached. The line is then kept high until a new start bit pulls it low again for a new byte to be transfered. Data-packets are often described in the form: $19200 - 8 - N - 1$, which simply means 19200 bit/s, 8 *data bits*, *No parity* and 1 *stop bit*. Figure 1.1 demonstrates a typical RS-232 signal.

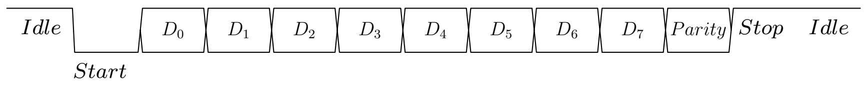

Figure 1.1: Example of an RS-232 signal with 8 data bits, 1 parity bit (Odd, Even or No parity) and 1 stop bit.