Current Control of Common Rail Injectors

An Evaluation of Programmable Logic Solutions

Magnus Pellijeff

Examiner: Associate Prof. Ingo Sander, KTH

Industrial Supervisors: Jens Svensson, Mats Hägg, Scania CV

Royal Institute of Technology, KTH, Stockholm

August 30th, 2006

Abstract

Scania XPI is a Common Rail system that will be standard in new Scania diesel engines. In the XPI engine control unit, an ASIC is used to regulate the current pulses controlling the fuel injectors. An ASIC is good at performing a specific task but is an inflexible solution, in that it allows no reprogrammability. In this Thesis it has been investigated if it is possible to replace the ASIC with a programmable logic device, of some kind, and how such a solution should be designed and implemented. A flash-based FPGA was found that met the specified requirements of this application. By developing current control logic and implementing it in the FPGA, the required size of FPGA was determined. The conclusion is that it is possible to replace the ASIC with a flash-based FPGA, if the FPGA is classified for automotive standard.

Sammanfattning

Scania XPI är ett Common Rail system, som blir standard i Scania's nya dieselmotorer. På motorstyrenheten till XPI sitter en ASIC, som används för att reglera de strömpulser som manövrerar bränsleinsprutarna. En ASIC är bra på att utföra en specifik uppgift, men är en oflexibel lösning, eftersom den inte är omprogrammeringsbar. I detta examensarbete har det undersökts om det är möjligt att ersätta ASIC'en med någon typ av programmerbar logik, samt hur en sådan lösning skulle kunna se ut och implementeras. En flash-baserad FPGA, som stämmer överens med kravspecificationen, hittades. Genom att utveckla och implementera strömstryrningslogik för FPGA'n kunde nödvändig storlek på FPGA bestämmas. Slutsatsen är att det är möjligt att ersätta ASIC'en med en flash-baserad FPGA, om den är klassificerad för automotive standard.

Acknowledgments

This Master Thesis was carried out at Scania, Powertrain Control Systems, during the time period between September 19th, 2005, and March 31st, 2006. I would like to thank my supervisors at Scania, Jens Svensson and Mats Hägg, for their patience and support, and Johan Hansson who was my co-worker during the work on the Thesis. I would also like to thank my examiner Associate Professor Ingo Sander at KTH, and opponent Andreas Gunberg for valuable comments on the report. Special thanks to David Källberg from Acal, for his patience with my many questions, and a big special thank you to Jeannine Labelle who helped me look through the report. Finally I want to thank the rest of the people at Scania, NEP and NEV, for their support and good company during the work on this Thesis.

Table of Contents

Index of Figures

Index of Tables

List of abbreviations

AD	–	Analog to Digital
ASIC	–	Application Specific Integrated Circuit
BSDL	–	Boundary Scan Description Language
BST	–	Boundary Scan Test
CAN	–	Controller Area Network
CBIC	–	Cell Based Integrated Circuit
CPLD	–	Complex Programmable Logic Device
CPU	–	Central Processing Unit
ECU	–	Electronic Control Unit
EDA	–	Electronic Design Automation
EDIF	–	Electronic Design Interchange Format
FPGA	–	Field Programmable Gate Array
FROM	–	Flash Read Only Memory
FS	–	Fuel System
FSM	–	Finite State Machine
GA	–	Gate Array
HDL	–	Hardware Description Language
HPI	–	High Pressure Injection
IC	–	Integrated Circuit
I/O	–	Input / Output
IP	–	Intellectual Property
IDE	–	Integrated Design Environment
ISP	–	In System Programmability
JTAG	–	Joint Test Action Group
LAPU	–	Live At Power Up
LUT	–	Look Up Table
MCU	–	Micro Controller Unit
MGA	–	Masked Gate Array
OTP	–	One Time Programmable
PAL	–	Programmable Array Logic
PCB	–	Printed Circuit Board
PDE	–	Pumpe Düse Einheit
PLA	–	Programmable Logic Array
PLD	–	Programmable Logic Device
PLL	–	Phase Locked Loop
PROM	–	Programmable Read Only Memory
RTL	–	Register Transfer Logic
SPI	–	Serial Peripheral Interface
SPLD	–	Simple Programmable Logic Device
STAPL	–	Standard Test And Programming Language
UIS	–	Unit Injector System
VHDL	–	VHSIC Hardware Description Language
VHSIC	–	Very High Speed Integrated Circuit
WDT	–	Watchdog Timer
XPI	–	eXtreme High Pressure Injection

Chapter 1 Introduction

This Master Thesis was carried out at Scania, Powertrain Control Systems, as part of a project aimed at developing a new engine control unit, for Scania diesel engines. Presently, there are different fuel injection systems in use in Scania diesel engines. Each fuel injection system uses a control unit, which is applicable to only that specific fuel injection system. The injection principle considered in the Thesis has been Scania XPI (eXtreme High Pressure Injection), a Common Rail System, which will be standard in new Scania diesel engines. In the XPI engine control units, an ASIC (Application Specific Integrated Circuit) is used to regulate the current pulses which control the fuel injectors. A micro processor, in the control unit, sends a request to the ASIC when an injection event will occur, and the ASIC maintains the current pulse. An ASIC is excellent at performing a specific task, but is an inflexible solution in that it allows no reprogrammability. The design process of an ASIC is also complicated. It is therefore normally made on contract by an external company. This Thesis explores the possibility of finding a more flexible solution, which will enable Scania to easily develop, change and update the functionality, which is now implemented in the ASIC.

1.1 Problem Definition

The aim of this Master Thesis has been to investigate the possibilities of replacing the XPI fuel injection ASIC with a flexible, reprogrammable, solution. The main alternatives originally considered were: replacing the ASIC with either a micro processor, or a programmable logic device; or removing the ASIC and implement its functionality entirely with software, and use only already available hardware in the engine control unit. The Thesis was then divided into two branches. One branch, which is covered by this thesis, evaluates a PLD (Programmable Logic Device) solution. The other branch, which will not be covered in this report, evaluates software and micro processor solutions.

1.2 Method

The evaluation was carried out by implementing a PLD based fuel injection system, which corresponds to the ASIC solution used today. To determine the requirements of a PLD solution, the currently used fuel injection system was studied and analyzed (see Chapter 4). To find a suitable PLD the commercial market was surveyed, and different PLD solutions were considered (see Chapter 3). A fuel injection system design was implemented based on a PLD which matched the stated requirements, and which also was available on the commercial market (see Chapter 5). The conclusions from the evaluation are found in Chapter 6. Figure 1-1 shows the phases in which the thesis was carried out.

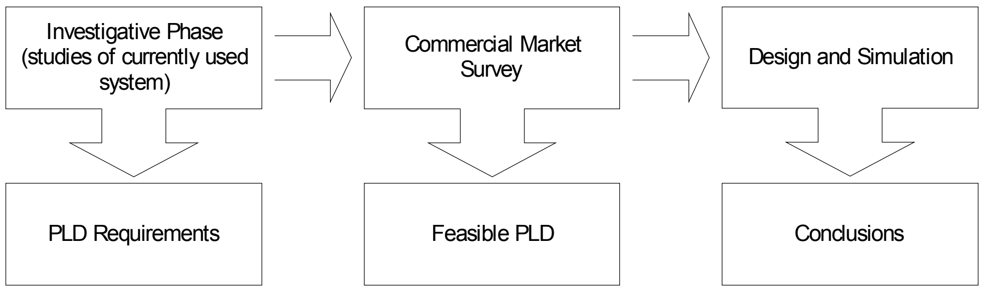


Figure 1-1 Sequence of Work

Chapter 2 Background

In this section, an overview of the fuel injection systems used today is provided. In section 2.1 the different fuel injection principles used in Scania diesel engines are introduced. Section 2.2 explains how the injection event is controlled by the ECU (Electronic Control Unit, or Engine Control Unit) with help from the ASIC.

2.1 Fuel Injection principles

Over the years, different techniques have been used for fuel injection in Scania diesel engines. The most commonly used techniques, at the current date, are PDE and HPI, which both are so called unit injector systems. XPI, which is a common rail system, will be used in the next generation of Scania diesel engines. The evolution of injection techniques has been driven by changing emissions restrictions over the years [1]. This thesis will only deal with XPI. However, PDE and HPI are introduced for background purpose.

2.1.1 Scania PDE

Scania PDE (Pumpe Düse Einheit) was developed for the Euro II emission standard in 1996. In PDE injectors (Figure 2-1) the injection pressure is built up by the plunger which is powered by the engine camshaft. The ECU controls the injection time and the amount of fuel to be injected by applying voltage to the fuel inlet valve. Pressure is generated when the inlet valve is closed and the camshaft moves the plunger downwards. The injector nozzle is normally closed by the needle which is held down by a spring. However, when the generated pressure is larger than the spring pressure the needle is pushed open and the injection starts. The injection pressure is a function of engine speed and the injected fuel amount [2].

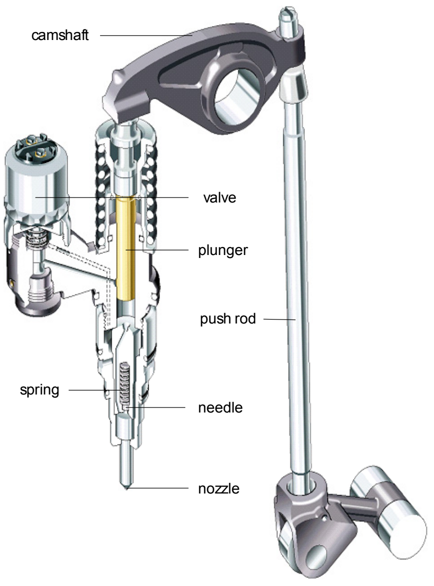


Figure 2-1 PDE Unit Injector

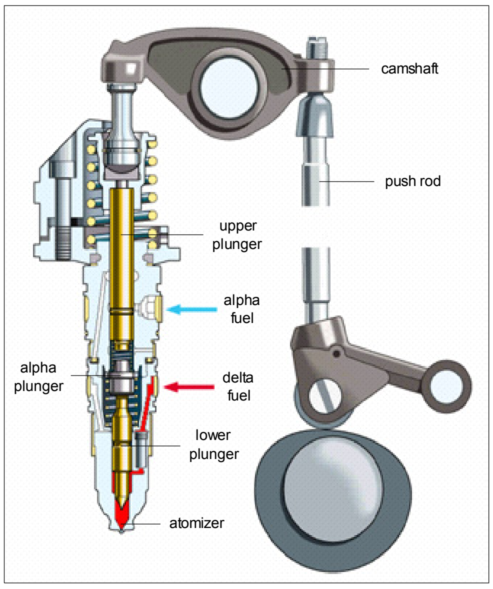


Figure 2-2 HPI Unit injector

2.1.2 Scania HPI

Scania HPI (High Pressure Injection) was taken into use for the Euro III emission standard in 2000. In HPI injectors (Figure 2-2), three plungers are used; the upper, the lower and the alpha plunger. Between the upper and alpha plunger, “alpha” fuel is led in. It will function as a hydraulic link to regulate the start of the injection. Delta fuel is led to the atomizer underneath the lower plunger. The ECU controls the time and amount of fuel to inject by applying voltage to the fuel inlet valve actuators. When the camshaft moves the plungers downward, pressure is generated in the atomizer. The HPI injector has an open nozzle. Delta fuel will be injected to the cylinder when the pressure in the atomizer overcomes the cylinder pressure. The injection pressure is a function of engine speed and the injected fuel amount [3].

2.1.3 Scania XPI

Scania XPI (eXtreme high Pressure Injection) will be used for the Euro V emission standard, which will be the forthcoming standard in 2008. Unlike PDE and HPI, unit injectors, Scania's XPI is a common rail system, which means that the injection pressure is generated separately from the fuel injection (see Figure 2-3). The injection pressure is generated in an accumulator, called the rail, by a high-pressure pump, which may be asynchronous with the injection timing. The camshaft is not needed by the injectors, but provides power to the high-pressure pump. The injectors are regulated by solenoids which are controlled electrically by the ECU. The time each injector is open, and the system pressure defines the how much fuel is injected [4].

Since injection pressure, timing and duration is not dependent of the camshaft position it is possible to have multiple injection pulses. For example: pilot injections that occur before the main pulse can be used to improve noise and emissions control; post injections, appearing after the main pulse, can be used for NO_X-control. The average injection pressure for XPI is higher than that utilized for PDE and HPI [1]. As a result the injected fuel atomizes easily and burns cleanly with reduced exhaust emissions and increased efficiency. XPI is described briefly, since it still is under development, and most available information about XPI is confidential.

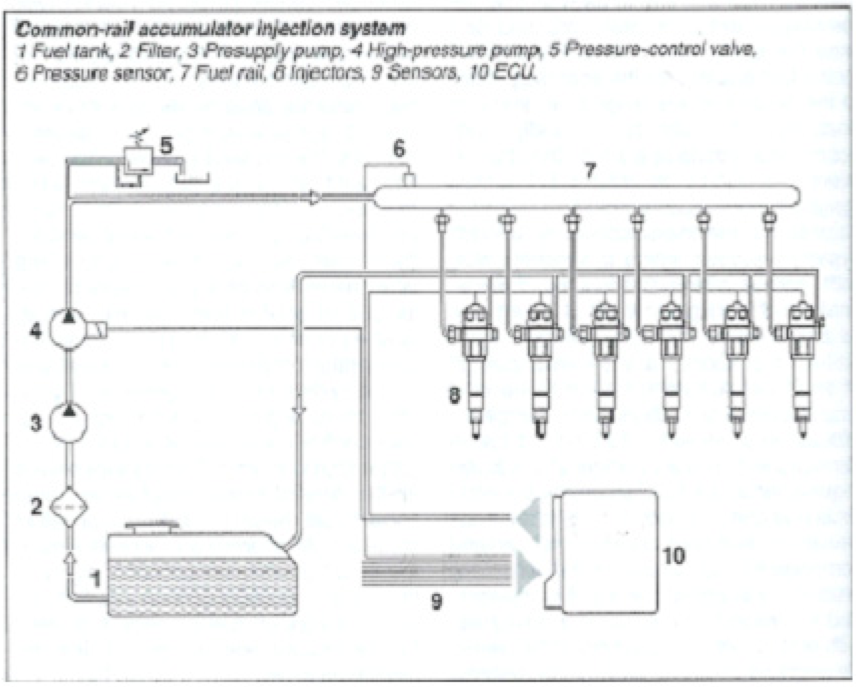

Common-rail accumulator injection system
1 Fuel tank, 2 Filter, 3 Presupply pump, 4 High-pressure pump, 5 Pressure-control valve,
6 Pressure sensor, 7 Fuel rail, 8 Injectors, 9 Sensors, 10 ECU.
1
2
3
4
5
6
7
8
9
10

Figure 2-3 Common Rail System [Bosch Automotive Handbook, 5th edition]

2.2 Electronic Control Unit

In Scania trucks, the ECU controls the engine and the various actuators used to operate different units belonging to the engine. The ECU is mounted on the engine, inside the engine compartment. It is activated by the ignition key, and communicates with the rest of the truck through the CAN[1]-buses. The tasks of the ECU includes such activities as controlling the cooler fan, the exhaust brake, the starter motor, the engine speed, the fuel actuators, as well as other tasks [5]. The architecture of the ECU depends on the engine type and the injection principle used in the engine. The injection event described in this Thesis belongs to an XPI fuel injection system. The ECU is described briefly, since it still is under development, and most available information about the ECU is confidential.

2.2.1 Current Pulse

The injectors are regulated with solenoid based actuators which are controlled with current pulses from the ECU. In Figure 2-4 a typical current pulse is shown. The higher current level is called the "pull-in level", and the lower current level is called the "hold level". To compensate for inertia in the solenoids [6] the current level is higher in the injector opening phase than it is when the injector is fully opened. The task of the ASIC is to produce this current pulse when the CPU signals an injection event to the ASIC [7].

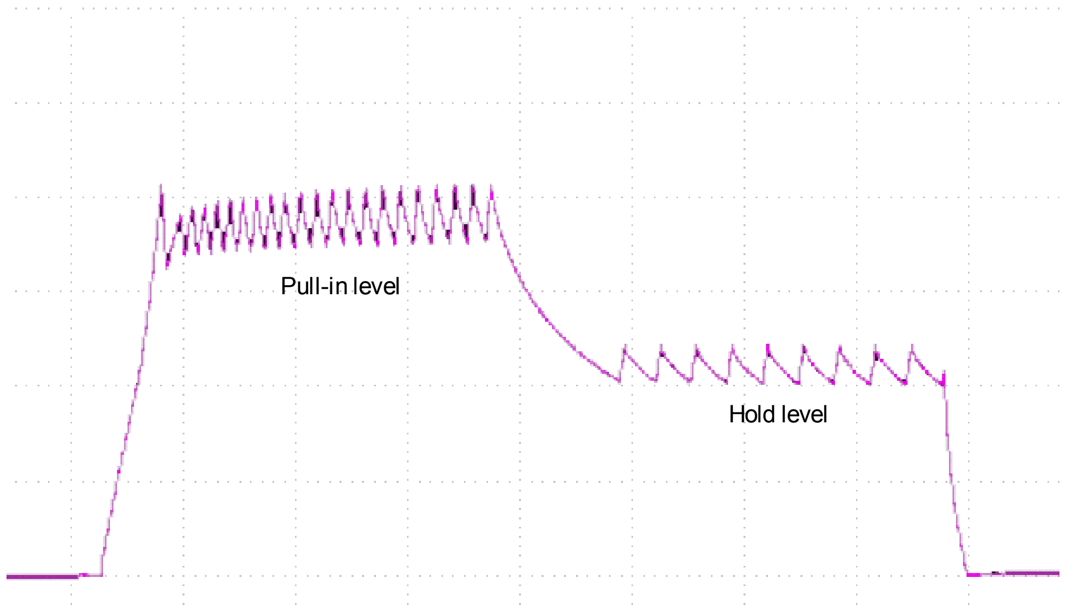


Figure 2-4 Typical Current Pulse

[1] The CAN (Controller Area Network) is an ISO standard for serial data communication. The protocol was developed for automotive applications. In Scania trucks all electronic control units are connected in a network of CAN-buses.

2.2.2 CPU

The CPU (Central Processing Unit) is the coordinator of the tasks that the ECU has to execute. There are several peripheral units belonging to the CPU. These units are used for analog to digital conversions, communication on the CAN-bus, communication on the SPI-bus (SPI is explained in Appendix B), as well as for other uses. One of the peripheral units, in this Thesis called *PU* (*Peripheral Unit*), calculates the motor speed and determines the position in the injection event. This unit is in charge of the injection event and sends the control signals to the ASIC. When an injection event is about to begin the *PU* signals the ASIC by setting the “injection” signal to logic high. Another signal controls which cylinder is active, and a third signal controls whether the current level is “pull-in” or “hold”. The “injection” signal is set logic low when the *PU* determines that the injection event is over [8].

2.2.3 ASIC

The ASIC controls the actuators through the source and the sink drivers. Figure 2-5 shows how the source and sink drivers are connected to the ASIC, and to a solenoid actuator. The source driver, as the name implies, is connected to the current source while the sink driver is connected to the ground. Each actuator is connected to a source driver and a sink driver. To let current flow through the actuator both drivers need to be activated by the ASIC [9].

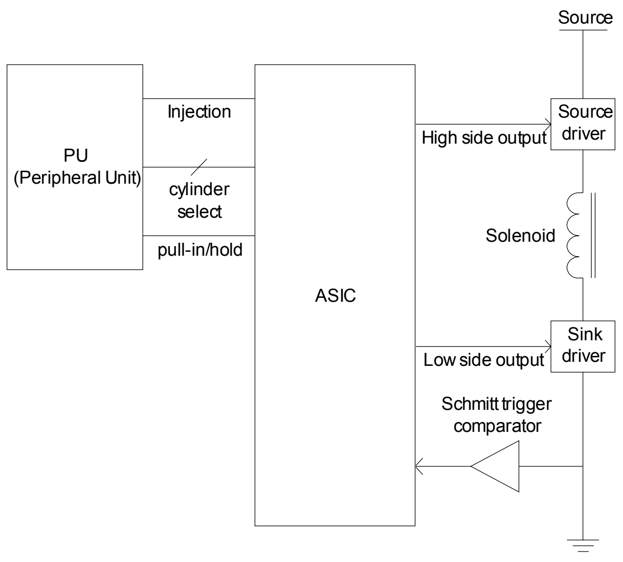


Figure 2-5 Source and Sink Drivers

When a cylinder is selected by the *PU*, the sink driver for the corresponding actuator is activated by the ASIC. When the “injection” signal switches to logic high, the ASIC activates the high side driver, which causes current to flow through the solenoid. The current level is measured, and constantly monitored by Schmitt trigger comparators[2], which are connected in a feedback loop to the ASIC. When a Schmitt trigger indicates to the ASIC that the current has reached the appropriate level, the ASIC maintains this level by switching the source driver on and off, as a reaction to the feedback from the Schmitt trigger [8].

[2] The Schmitt trigger comparator has hysteresis, which means it has two different threshold levels, for a rising and a falling current. In this application the threshold level for the rising current is higher than for the falling current. This prevents an excessive, on and off, switching as the current level drifts around the set point.

The set point of the Schmitt trigger defines the current level to be held. The highest current level allowed, deviating from the set point, is called the peak, and the lowest is called the valley [7] (see Figure 2-6). A rising edge on the Schmitt trigger output indicates that the peak current level is reached. When the ASIC detects the rising edge it turns the source driver off to let the current level drop. A falling edge on the Schmitt trigger indicates to the ASIC that the valley is reached. The ASIC then enables the source driver again. The current level is kept until either the "pull-in/hold" signal from the *PU* changes or the "injection" signal turns to logic low [8].

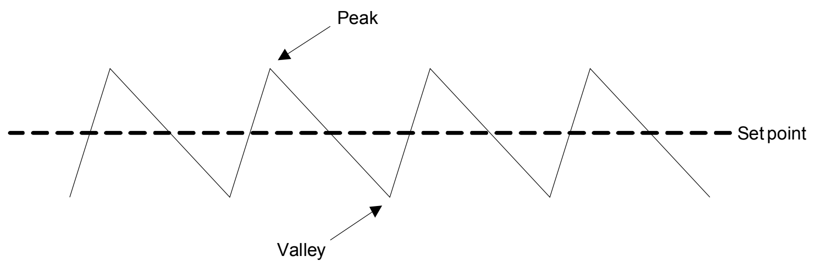


Figure 2-6 Current Peak and Valley

The ASIC will also diagnose if, and when, "certain" faults occur during and in between injection events. Whenever a fault occurs it is indicated in a register called the fault register. The fault register can be read by the CPU via the SPI-interface. Depending on what the settings are for an occurrence of a certain fault, the ASIC may also execute a predefined action as a safety measure. It might for example halt the injection event, or move on to the next state [10].

Because information about the ASIC's functionality is proprietary of Motorola, no details can be revealed in this report. ASIC technologies in general will be explained in 3.1.

Chapter 3 ASIC and PLD Overview

This section gives a short introduction to ASIC and PLD technologies. The JTAG interface, which is a common programming and debugging interface used for these technologies, is also explained.

3.1 ASIC

An ASIC (Application Specific Integrated Circuit) can be generalized as an integrated circuit specified for one particular task. An IC (Integrated Circuit) is a collection of one or more gates, which are fabricated on a silicon die[3] [11]. Not all specialized ICs are however generalized as ASICs. Examples of typical non-ASICs are standard parts, such as memories and microprocessors. But there is no absolute definition of exactly what an ASIC is. Basically there are three types of ASICs; full-custom, standard-cell-based and gate-array-based ASICs. The last two types are regarded semi-custom ASICs [12].

3.1.1 Full-Custom ASICs

In a full custom ASIC, logic cells, circuits and layout are specifically designed for the ASIC. For different reasons no standardized logic cell libraries are used, typically because they do not meet the requirements in a new or specialized ASIC design. Full-custom ASICs are the most expensive ASICs to manufacture and design, and there are often problems involved with specially designed parts. For these reasons full-custom ASICs are the least common type of ASICs [12].

3.1.2 Standard-Cell-Based ASICs

A standard-cell-based ASIC, also called CBIC (Cell-Based Integrated Circuit), is a semi-custom type of ASIC. The IC manufacturer has developed libraries of predesigned logic cells, such as AND, OR gates, MUXes and flip-flops etc, for use on the CBIC. These standard cells can also be connected to larger predesigned blocks such as microprocessors or microcontrollers. The designer defines only placement of the standard cells and the interconnections between blocks on the CBIC. Custom-cells are created only if absolutely necessary [12].

3.1.3 Gate-Array-Based ASICs

The gate-array ASIC is a semi-custom type of ASIC. The structure of the gate-array, or GA, is an array of gates in which the interconnections initially are unspecified. The designer specifies which gates and interconnections to use by applying custom masks on the top metal layer. For this reason the GA is also called MGA (Masked Gate Array). The placement of the gates is fixed and cannot be changed. There is no possibility to create custom-cells, which makes the chip layout of a GA less optimized than for a CBIC. This typically results in a larger die size and therefore a higher cost for the die. The GA design time however is normally much shorter than for a CBIC which gives a lower nonrecurring engineering cost for the GA [11] [12].

Basically three kinds of GA exist, channeled, channelless and Structured GA. In the channeled gate array the transistors are arranged in rows on the silicon die, and spaces between the rows are left for wiring. The channelless gate array has no space in between the rows of gates. Instead routing is customized on top of unused gates, which gives a more compact layout and allows more logic to be implemented. One of the drawbacks with

[3] A die is an IC chip.

the GA is that the array of gates is fixed. This makes implementation of certain logic, such as memory cells or microcontrollers, inefficient. The structured gate array has an area, like the CBIC, dedicated for such logic circuits. The Structured GA can be either channeled or channelless [12].

3.2 Programmable Logic Devices

A PLD (Programmable Logic Device) can be considered a kind of programmable ASIC. The PLD die is produced at an IC foundry, and is covered with rows of logical gates. The interconnections between the gates are unspecified and designed to let the PLD designer program the interconnections, outside the foundry [13]. The technology used to configure the interconnections depends on the type of PLD, which is explained in 3.2.1, 3.2.2 and 3.2.3.

In the design entry both ASICs and PLDs are modeled with a HDL (Hardware Description Language). In the next step, the design is synthesized to a logic netlist, which describes the actual hardware created from the HDL code. The netlist is then placed on the device and the interconnections are routed. For PLDs the routing can be made entirely by the designer, while the ASIC is manufactured entirely at a foundry [12] [14].

In Table 3-1 a few examples of PLDs are shown. PLAs, PALs, GALs and EPLDs are regarded SPLD technologies. Often SPLDs and CPLDs are grouped as PLDs and a distinction is made between PLDs and FPGAs, because they differ slightly in architecture. This distinction will not be made here since an FPGA is also a type of programmable logic device. Furthermore, the SPLD and CPLD have different architectures from each other.

Table 3-1 Programmable Logic Devices

PLA	Programmable Logic Array
PAL® (AMD)	Programmable Array Logic
GAL® (Lattice Semiconductor)	Generic Array Logic
SPLD (often just called PLD)	Simple Programmable Logic Device
CPLD	Complex Programmable Logic Device
EPLD	Erasable Programmable Logic Device
FPGA	Field Programmable Gate Array

Programming of a PLD is often made through the JTAG interface (explained in 3.3). Depending on the technology, the PLD can be either one-time-programmable or reprogrammable.

3.2.1 SPLD

The PLA was the first type of PLD to be invented. It is based on programmable arrays of AND, and OR gates. The PAL is an enhanced PLA which has one programmable AND plane and a fixed OR plane. This makes the PAL faster and cheaper than the PLA, but less flexible since it only has one programmable plane. Programmable logic devices are often based on the PAL or PLA structure. For this reason the PAL and the PLA are commonly referred to as PLDs. To avoid confusion with other types of PLDs this type of device can be generalized as an SPLD (Simple PLD) [11] [15]. When an SPLD is unprogrammed it has fusible links at all connection points. By blowing the fuses for all unwanted connections the SPLD is configured.

The EPLD was originally developed by Altera and is similar to the PAL. It differs in that it uses EPROM/EEPROM technology (explained below) for configuration which allows it to be reprogrammed. The GAL is a variant of the EPLD which was developed by Lattice Semiconductor. Both the EPLD and GAL are regarded SPLD technologies [15].

EPROM (Electrically Programmable Read Only Memory) is based on floating-gate MOS (Metal Oxide Semiconductor) transistors. In the initial state, the floating gate of every transistor in the EPLD is uncharged. This makes all transistors connected as logical links. The EPLD is programmed by applying a high programming voltage to the drain of each transistor where the connection shall be removed. This forces electrons to the floating gate and disconnects the transistor. When the programming voltage has been removed the electrons remain on the floating gate. By exposing the EPROM to strong ultraviolet light the configuration can be erased. The EEPROM, (Electrically Erasable Programmable Read Only Memory) differs from the EPROM by using an electric field to erase the configuration [13].

3.2.2 CPLD

The AND-OR structure, of the SPLD, can only be scaled to a certain size before the performance is degraded due to occurrences such as capacitive effects, and leakage currents. Also the chip area would be used very inefficiently. In terms of silicon use, it would be more efficient to use several SPLDs rather than one large SPLD. This is also how the structure of a CPLD (Complex Programmable Logic Device) is arranged. The CPLD structure (Figure 3-1) is basically a collection of SPLDs, called macrocells, and a programmable interconnection area in between, on the same silicon die [11] [16]. The fixed structure of the CPLD makes the internal delays easy to predict, but also sets limits to the logic utilization of the area on the CPLD die [18]. CPLDs are commonly based on EPROM or EEPROM techniques [15].

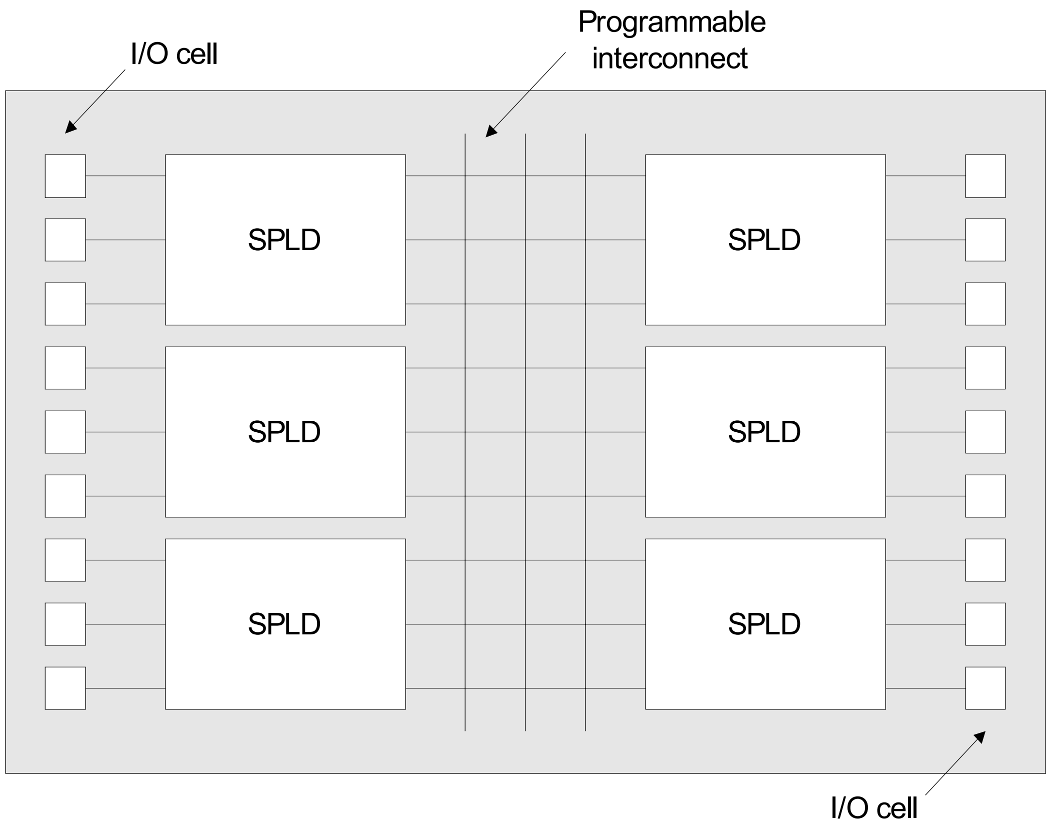


Figure 3-1 Structure of CPLD

3.2.3 FPGA

The FPGA (Field Programmable Gate Array) is another kind of PLD with a different structure than the SPLD and the CPLD. The FPGA was developed around the same time as the CPLD, but with a different approach to create larger programmable logic devices. The FPGA (Figure 3-2) contains a much larger number of small individual logic cells, compared to the CPLD macrocells [11]. It also has a large interconnection matrix structure, surrounding the logic cells, covering the entire chip. The FPGA structure is similar to that of the MGA, but the FPGA can be configured just as easy as any PLD [12]. The programmable logic blocks, or basic logic cells, of an FPGA looks different depending on the manufacturer but are usually based on flip-flops and MUXes or LUTs (Look Up Tables) [17]. The propagation delay of FPGAs is greatly affected on how the routing is made internally. It is therefore difficult to predict the performance of a circuit before it has been completed [18].

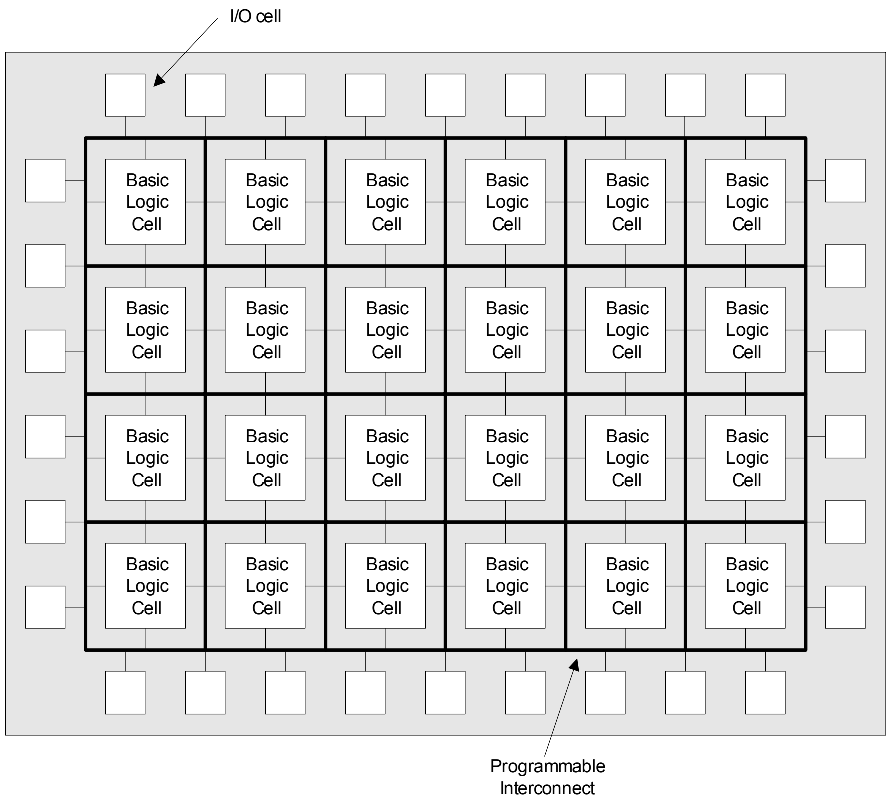


Figure 3-2 Structure of FPGA

There are both reprogrammable and OTP (One Time Programmable) FPGAs. OTP FPGAs are based on antifuses. An antifuse is the contrary of a fuse. Unprogrammed it is an open-circuit, but when a programming current is forced upon it a connection is created [13]. Once programmed the configuration is permanent. Reprogrammable FPGAs use SRAM (Static Random Access Memory) or Flash technology to store the configuration [19].

The SRAM is volatile and needs power supplied to retain its configuration. At every startup an SRAM FPGA must be reconfigured. Usually an external PROM (Programmable Read Only Memory) is used to load the FPGA with its configuration, but the FPGA can also be configured from an MCU (Micro Controller Unit) or an internal PROM. An FPGA with an internal PROM has a power-up time above 200 µs. With an external PROM the startup time is above 200 ms [19].

Flash is a type of compact and fast EEPROM. Flash FPGAs are faster than any other PLD technology, and faster than antifuses, according to Actel, who produces both flash and antifuse FPGAs. The Actel ProASIC3 flash FPGA has a power-up time of only 50 μs whereas an antifuse device needs 60 μs [19].

3.3 JTAG

ASICs and PLDs are often equipped with the JTAG interface. For PLDs this interface is used for programming and debugging, and for ASICs it is used for debugging, since an ASIC can not be programmed.

JTAG (Joint Test Action Group) was originally a group of European manufacturers, which then was called JETAG (Joint European Test Action Group), which formed in 1985, with the purpose to study board testing. When North American companies joined in 1986, JETAG became JTAG [20]. The JTAG introduced two IEEE standards, which will be explained here. IEEE 1149.1, also known as boundary-scan, was first developed, followed by IEEE 1532, which is known as ISP (In-System Programming) [21].

3.3.1 Boundary-scan

The JTAG 2.0 test standard formed the basis of “IEEE 1149.1 Test Access Port and Boundary-Scan Architecture” or simply Boundary-scan. It is a method of testing boards, ASICs, FPGAs or any device supporting the JTAG standard [20].

In an ASIC boundary-scan, logic is added on top of the actual design. Adding boundary-scan may affect the performance of the ASIC negatively, and considerable development time may be needed for optimization of the design [20]. In an FPGA boundary-scan logic is often already included and optimized by the manufacturer [22], and hence performance is not affected.

Boundary-scan uses a four wire test interface (with an optional fifth master reset) called TAP (Test Access Port, see Table 3-2). The TAP interface is connected to the TAP controller, which is located inside the device (see Figure 3-3). The TAP controller is a 4-bit state machine controlling the test event. It is clocked on the rising edge of TCK whereas TMS controls state transitions in the state machine. The optional TRST pin is dedicated to reset the TAP controller. [20] [23].

Table 3-2 Test Access Port

TDI	Test Data Input
TDO	Test Data Output
TMS	Test Mode Select
TCK	Test Clock
TRST (optional)	Test Reset Input Signal

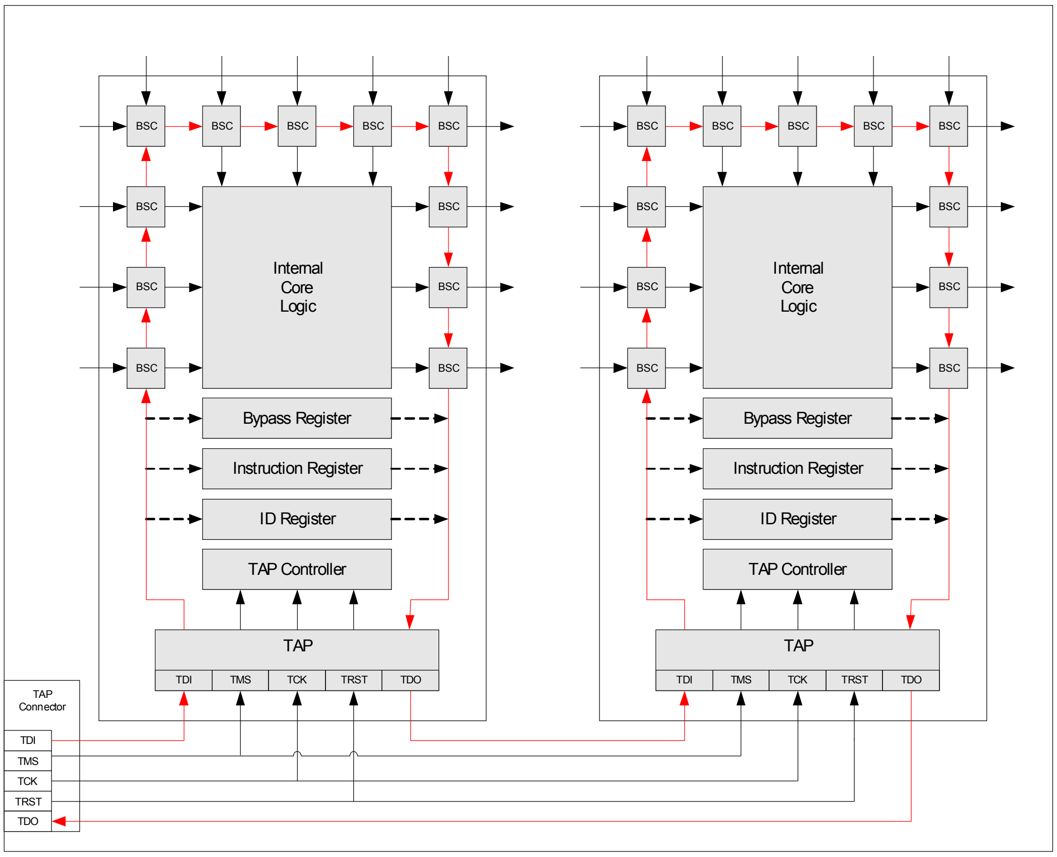


Figure 3-3 PCB with two devices hooked up in a boundary-scan chain

Each I/O pad has a special logic cell, called BSC (Boundary-Scan Cell), to monitor signals. These cells are joined together in a serial chain around the device and form the Boundary-scan register, where TDI is input and TDO is output. When the device is not in test mode, the input and output signals are allowed to just pass the BSCs without any action taken. In test mode however, the TAP controller tells each BSC to either monitor its inputs, capture received data, shift data to the neighboring cell or output data. Defects show as discrepancies between the expected and the actual data [20].

The Bypass Register is selected whenever no tests are to be performed on a particular IC during board level test mode. The ID register is read-only and provides information about the device manufacturer, part number and revision. It is used to ensure that the device is at the correct location in the boundary-scan chain [20].

3.3.2 In-System Programmability

ISP (In-System Programmability) means that a programmable device can be programmed while it is installed in a system, as opposed to being programmed prior to assembly. Devices that support ISP typically have hardware logic that can generate the special programming voltage from the voltage supply rail, and serially communicate with the programmer. ISP can be performed from an external programming device or from an external MCU [21].

IEEE 1532 adds programming instructions and associated data registers to JTAG compliant devices. It is complementary to the JEDEC-approved Jam™ STAPL (Standard Test and Programming Language). While IEEE 1532 is a hardware standard that defines the actual ISP algorithm for each device, Jam STAPL is a software standard that defines the file format that stores the programming information [21].

Chapter 4 PLD Requirements

The automotive environment is very hostile for electronic systems. Temperatures are high, there are electromagnetic pulses from the ignition system, and violent transients on the power supply lines [24]. The current control system must be able to tolerate this hostile environment. But most importantly it must produce the current pulses to the injection actuators at exactly the right moment, and it will have to diagnose faults whenever they occur. Furthermore the system must not be too expensive. It is preferable to have a system with as few components as possible since each physical unit in the design will increase the cost of the ECU.

4.1 Temperature Range

Integrated circuits are normally available in different classifications, for commercial, industrial, automotive and military standards. While the chip normally is the same for all standards, the standard specifies what the chip has been tested for. In this case, the PLD should be available in automotive standard with a temperature range from T_A= -40°C to +125°C. T_A specifies the ambient temperature which refers to the temperature of the surrounding environment. Often devices are specified for the junction temperature, T_J, which refers to the temperature on the die surface. The relationship between T_A and T_J is shown in Equation 3.2, which is derived from Equation 3.1. Since the ambient temperature, T_A, is a subset of the junction temperature, T_J, a device specified for a junction temperature in the same range would be less tolerant to the ambient temperature, and would not be sufficient for use in the ECU.

Junction-to-air thermal resistance [25]

$$\theta_{JA} = \frac{T_J - T_A}{P} \qquad \text{(Equation 3.1)}$$

T_J = junction temperature
T_A = ambient temperature
θ_{JA} = junction to air thermal resistance
P = power dissipation

$$T_J = T_A + \theta_{JA} P \qquad \text{(Equation 3.2)}$$

4.2 Price Range

An appropriate price range for a PLD was determined to around $5.00 USD for a volume of 50k units. This was decided from discussions with Tomas Logge, who at the time was manager for the ECU project, and Jens Svensson who is the new ECU project manager and also Thesis supervisor. Cost of development has not been considered in the calculations.

4.3 Real Time Requirements

Controlling the current pulse (see Figure 2-4) is essential for the injection event. It determines if and when the injectors are opened or closed. If the injectors do not open or close at the exact right moment, it could affect the emissions, create more engine noise and degrade the performance of the engine. Or in the worst case the engine would not function at all. Control of the injection event is a hard real time requirement that must be met. The injection event shall be deterministic, meaning that the beginning and end of the injection occurs when determined. The current rise and fall times must be kept within limits, and the current levels (see 2.2.1) held according to the configuration. It is also important that the injection starts immediately as the ignition key is turned, by the driver. There must not be any noticeable delay before the engine starts.

When the injection event is held, at either pull-in or hold level (see 2.2.1), the current pulse to the actuators is switched on and off rapidly, in reaction to the Schmitt-trigger comparators. Maintaining this current "ripple" by switching the current on and off at the exact right moment is the fastest event that will occur on the outputs. This is the maximum frequency that will be required on the outputs. In section 4.3.1 the current ripple is analyzed. In a PLD the internal logic will be similar to that of the ASIC. By analyzing the timers in the ASIC, the requirements could be further specified (see 4.3.2).

Running the diagnoses is another hard requirement. Setting the fault registers may not be critical to the injection event, but when "certain" faults occur immediate action must be taken (e. g. if the injection event must be halted in order to prevent damage). The diagnoses are dependent on the same timers as the rest of the design, and are therefore covered in section 4.3.2.

4.3.1 Analyzing Current Ripple

To get an idea of the switching frequency when a current level is being held, measurements were made with an oscilloscope on the inputs to the actuators (screen dumps from the oscilloscope are found in Appendix A). It is shown that the fastest transition between peak and valley is 8.60 µs. But the impedance and operating voltage of other types of injectors can be different and thus change the slope of the rising and falling current. According to Equation 3.4, which is derived from Equation 3.3, the current slope over time is dependent on the electrical characteristics of the solenoid coils, used in the injector actuators. A higher operating voltage or a lower coil resistance will cause a steeper current slope.

The current, i_L, during the inductor storage phase [26]

$$i_L = \frac{V}{R}\left(1 - e^{-t/(L/R)}\right) \qquad \text{(Equation 3.3)}$$

i_L = current through coil
R = coil resistance
V = voltage applied over coil
L = coil inductance

$$\frac{di}{dt} = \frac{V}{L} e^{-t\frac{R}{L}} \qquad \text{(Equation 3.4)}$$

Knowing this however, does not provide many details about how fast the PLD will have to operate. The current ripple does not reveal much about the internal operation of the ASIC. Therefore the ASIC had to be analyzed.

4.3.2 Analyzing Timer Resolutions

By analyzing the various timers used in the ASIC it is possible to determine the minimum clock frequency of the ASIC. For example, a frequency of 1 MHz is required for a timer with the resolution of 1μs, and 2 MHz is required for a 0.5 μs resolution. In the ASIC there is one timer with a resolution at 62.5 ns [27]. With a counter that triggers on every positive clock edge the corresponding frequency will be 16 MHz (see Equation 3.5).

$$f = \frac{1}{T} = \frac{1}{62.5ns} = 16.0MHz \qquad \text{(Equation 3.5)}$$

f = frequency
T = period

A PLD solution would be constructed with digital hardware logic like the ASIC. Since 16 MHz is the highest frequency required, for any timer in the ASIC, it can be used as a guideline for the PLD. Basically it proves that most PLD solutions are fast enough since they commonly can operate in frequencies above 100 MHz.

In a micro processor solution, it would be necessary to schedule every timing critical event, and make sure that the clock frequency is fast enough to handle them all. The micro processor kernel executes instructions in sequential order, and such a solution would be implemented very differently from the ASIC, or a PLD, that operates in parallel without the need of any kernel.

Chapter 5 Design and Implementation

In this section, it is explained how a PLD technology was chosen, and how the PLD based fuel injection system was designed, and implemented.

5.1 Choice of PLD

Before even considering designing a PLD based solution, it was necessary to determine if there were any feasible PLDs available on the commercial market, which would meet the requirements (stated in Chapter 4). Otherwise a PLD solution would have been rejected by Scania.

To decide which type of PLD to use, it was necessary to know the approximate amount of logic implemented in the ASIC. Scania only possess a document describing the functionality of the ASIC [28], but no HDL code or hardware netlist is available. Since the ASIC is custom made, it is hard to estimate the number of logical gates implemented in it. There is however an earlier prototype for the ASIC, made on an FPGA. Scania possess no information about the design in the FPGA, but datasheets for the FPGA can easily be downloaded from the manufacturer's website.

No CPLD[4] was found large enough to match the size of the prototype FPGA. The corresponding CPLD solution would require more than one CPLD, if all the logic cells of the prototype FPGA are utilized. Therefore, an FPGA solution was considered more suitable than a CPLD solution. However since both CPLDs and FPGAs are coded with a HDL, the code can be moved between the two technologies.

SRAM FPGAs are not very suitable for this particular application, mainly because the configuration must be reloaded on every start-up (see 3.2.3). The SRAM FPGA has a power-up time above 200 ms [19] which may be reasonable, if it is booted in parallel with the rest of the system. But adding this delay to the start-up of the entire system would not be acceptable, since it would delay the start of the engine, when the ignition is turned on. SRAM FPGAs also normally require an external non-volatile memory, to store the configuration (see 3.2.3). Since one of the aims of the thesis was to minimize the number of external units, this was undesirable. The SRAM FPGA is further sensitive to transients on the supply voltage (see section 3.2.3). It might need to reboot if a temporary current dip or brown-out in the electric system occurs.

Antifuse FPGAs add a little flexibility since the device can be programmed once. Antifuses are non-volatile and do not require any additional components. But once programmed the configuration is permanent. No field updates can be made to an already programmed device, if it is not entirely replaced (see 3.2.3).

Flash is a type of fast and compact EEPROM [29]. Flash FPGAs are both non-volatile and reprogrammable, but are not as common as SRAM FPGAs. Lattice Semiconductor has made SRAM FPGAs non-volatile by storing the configuration in internal flash memory. The startup time of such an FPGA is 1 ms [30].

[4] SPLD/CPLD manufacturers checked (October 2005): Altera, Atmel, Cypress Products, ICT, Lattice Semiconductor, Texas Instruments, and Xilinx.

Actel is the only manufacturer of FPGAs based solely on flash[5]. Because the flash technology is faster and more reliable than SRAM technology, and because it allows reprogrammability (which the antifuse technology does not), a family of flash FPGAs called ProASIC3 (described in Appendix C) was chosen from Actel. This FPGA family will be available in the automotive standard during the later half of 2006 or during 2007 [31]. In the mean time it is possible to start prototyping on a non-automotive ProASIC3 FPGA, since this project is still in a pre-study phase.

Actel tests their automotive flash FPGAs for a junction temperature, T_J, range between -40°C and +125°C [32]. This is not a limitation of flash, but is the standard test range for automotive flash FPGAs. Actel also produces military flash FPGAs which are defined for $T_{Ambient}$ = -55°C to + 125°C. The silicon chip is the same for all standards. Only the tests and possibly the packages are different [31]. For large sales volumes Actel can perform custom tests.

5.2 System Design

The rough estimation of size based on the prototype FPGA does not reveal any details of how much logic, inside the FPGA, is actually utilized. Since this information is not available to Scania it was necessary to develop a new design. The description of the Fuel System ASIC [28] served as a guideline for the FPGA design. An attempt has been made to minimize the FPGA by moving as much logic as possible into the already available *PU* (explained in 2.2.2). The system was modeled as shown in Figure 5-1. The internal design of the FPGA is unique, but the components surrounding the FPGA are basically identical to the ASIC solution (the schematic is greatly simplified). Some "rewiring" was made due to the functionality that was moved to the *PU*.

The FPGA design is synchronous and clocked with a 16 MHz external oscillator, but is asynchronous to the *PU* circuit. The *PU* signals to the FPGA when an injection event will start and the FPGA controls the different states in the injection event. The FPGA design is divided into two parts, called *Bank A* and *Bank B*. Each bank controls four cylinders with the source and boost drivers.

The description of the FPGA design is in this report is restricted, due to confidentiality issues.

[5] FPGA manufacturers checked (October 2005): Actel, Altera, Aeroflex UTMC, Atmel, Lattice Semiconductor, NEC, QuickLogic, and Xilinx.

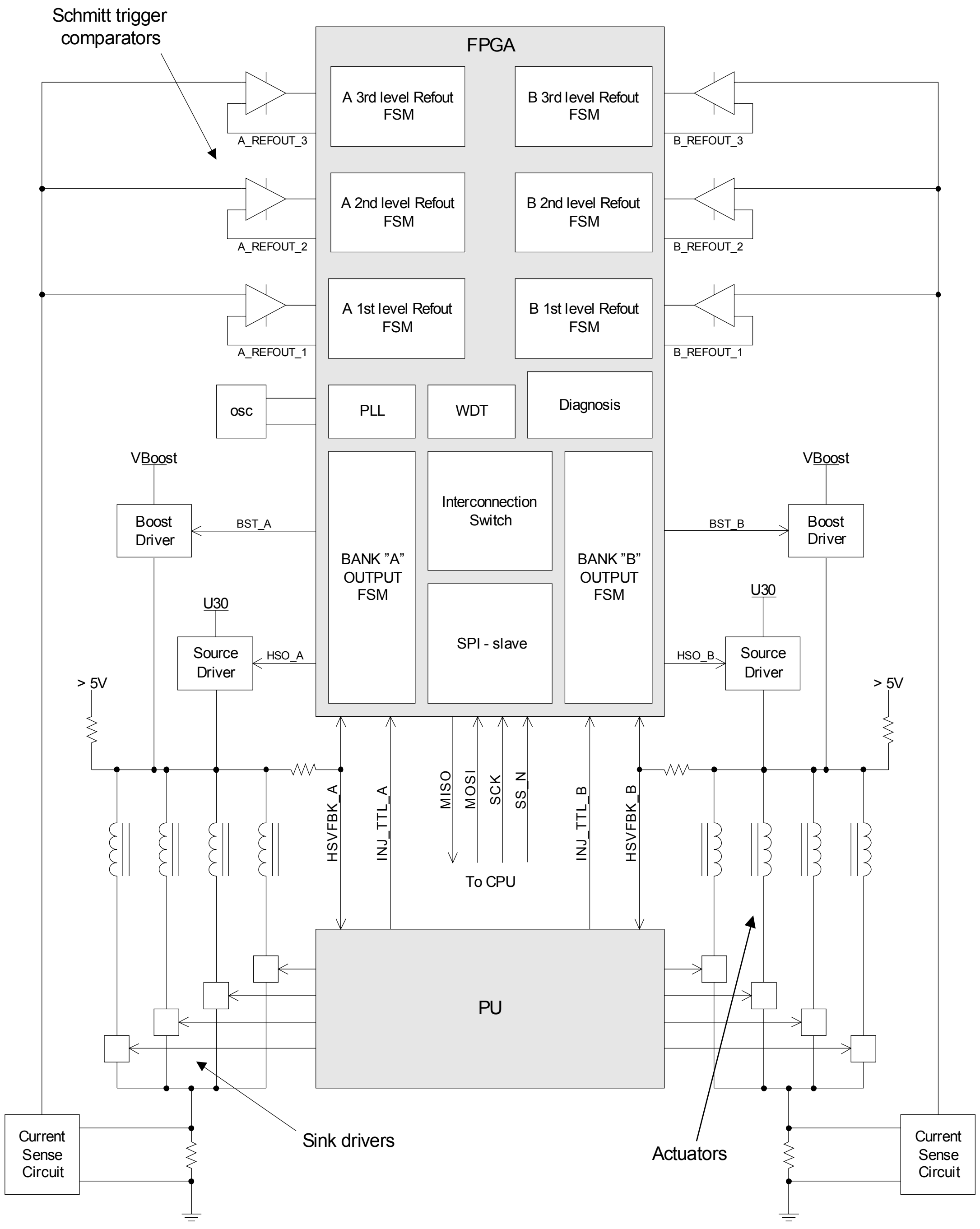


Figure 5-1 FPGA Fuel System Schematic

5.2.1 Cylinder Selection

The greatest difference from the ASIC is the cylinder selection, which is controlled by the ASIC in the currently used design. By moving this part to the *PU* it was possible to get rid of a set of control signals between the ASIC and the *PU*. Some of the not timing critical diagnoses were also implemented in the *PU*.

5.2.2 Output Finite State Machines

The most essential modules in the FPGA are two finite state machines (FSMs), called *Output FSMs*, which control the current driver circuits that maintain the current pulse. One *Output FSM* belongs to each bank, and controls the source and boost driver of that bank. The *Output FSMs* are enabled by control signals from the *PU* and uses the input signals from the comparators, and a set of timers to advance between the states. Details about the *Output FSMs* will not be explained here due to confidentiality issues, but is available for Scania personnel in document NE06165.

5.2.3 Diagnosis

The diagnoses run in parallel with the *Output FSM* on each bank. When a fault occurs, a signal called *fault_temp_x* (where x represent the fault bit) is set. This is the fault check, which reports when a fault has occurred. The fault checks are implemented as flip-flops. When the *fault_temp_x* signal is set, a bit corresponding for that fault bit is set in a "fault vector". Each bit in the fault vector has its own register where the bit is stored. The bit will remain set until the register is reset, or until the power to the FPGA is turned off. There is one fault vector for each bank. Details about the diagnosis will not be explained here due to confidentiality issues, but is available for Scania personnel in document NE06165.

5.2.4 Refout Finite State Machines

There are a total of six FSMs, called *Refout FSMs*, which control the reference signals to the Schmitt trigger comparators. The reference signals are created with PWM (Pulse Width Modulation) and define the current levels that the FPGA must keep. Three *Refout FSMs* belong to each bank. The *Refout FSMs* are identical but are typically configured to produce different reference signals, which are equal on each bank.

The *Refout FSM* has three states: *On* state in which the reference output signal is turned on, *Off* state in which the reference output signal is turned off, and *Idle* state in which is the output is also turned off. Two clock signals are used: the 16 MHz system clock (*clk*) and a 62.5 kHz clock (*clk_2*), with a period 256 times the period of the system clock.

The FSM is always initiated in *Idle* state but is advanced to *On* state in the first positive clock edge of *clk_2*, if the *enable* signal is '1' and if *reset* is '0'. Every time the FSM is in *On* state the *Refout* output is turned on and a counter, clocked by *clk*, is enabled. When this counter reaches a value of *X*, between 0 and 255, which is specified in an 8-bit register, the FSM moves to *Off* state. In *Off* state, *Refout* is turned off and the counter is reset. The FSM remains in this state until the next positive clock edge of *clk_2* occurs. It then moves back to *On* state again, and the whole procedure is repeated. If *enable* turns to '0' or *reset* turns to '1' the FSM is returned to *Idle* state.

The value *X* divided by 255 determines the duty cycle of the PWM, Figure 5-2. If *X* is '0' the output will constantly be low and if *X* is 255 the output will constantly be high.

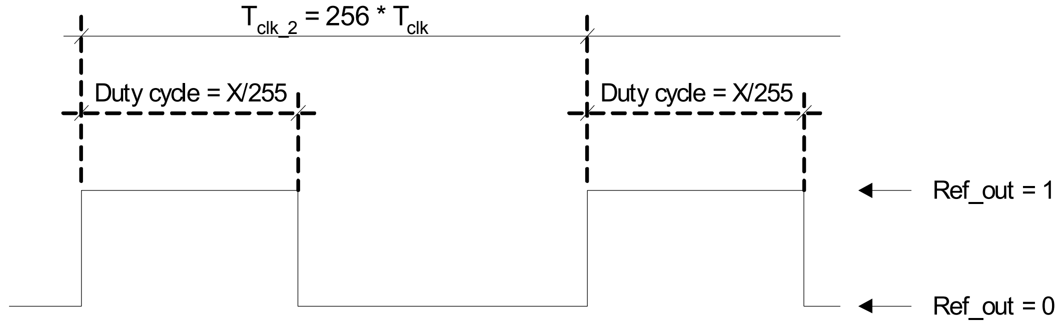


Figure 5-2 Refout PWM Signal

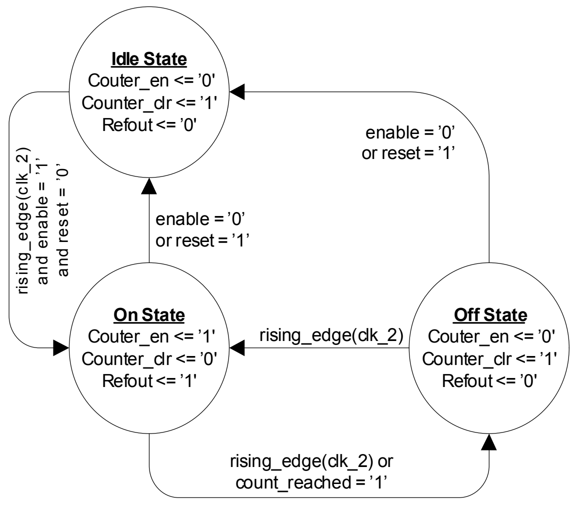


Figure 5-3 Refout FSM State Diagram

5.2.5 SPI-Slave and Interconnection Switch

An SPI-Slave, IP (Intellectual Property) core, will be implemented for communications with the CPU. Register values for timers in the FPGA are configured by the CPU via the SPI-interface. An interconnection module translates the address requests from the SPI and routes the address to the corresponding register.

5.2.6 In-System Programmability

ISP is made from the CPU via the JTAG-interface. For this purpose a special software module called Direct C must be implemented in the CPU. This module provides C code files containing the ISP programming algorithm. DirectC requires 100 KB of the system memory and the programming file size depend on the size of the FPGA [33]. DirectC is available to download for free from Actel's website [34].

5.2.7 Phase Locked Loop

The PLL (Phase Locked Loop) is a circuit available on the FPGA chip, which can be used for frequency division or multiplication. It can produce up to three global clock signals, in a range from 0.75 MHz to 350 MHz, which are synchronous to the oscillator input frequency. The PLL will be used in the design to provide different clock frequencies for the timers with different resolutions, which are utilized throughout the entire design.

5.2.8 Watchdog Timer

A watchdog timer module resets the FPGA to its initial state if the oscillator signal is absent. This prevents the FPGA from getting stuck in a state in which the outputs are opened, therefore preventing a situation with an ever increasing current level from occurring. This protects the hardware components from possible damage. The watchdog needs a reference clock signal from the CPU. The reference clock has a higher frequency than the oscillator clock. On every clock cycle of the reference clock, a counter increases its value. On every clock cycle of the oscillator clock the counter is reset. If the oscillator clock signal, for some reason would be absent, the counter value would continue to increase, and eventually exceed its timeout value (defined in a register). In this case the watchdog timer would reset the entire FPGA circuit. The reset signal forces all FSMs to return to idle state, where they will remain until the oscillator clock is active again.

5.3 FPGA Functional Description

The *PU* combines information from the CPU with calculations of the position of the engine crankshaft angle to determine the length of the injection pulses, when they will occur, and which cylinder to activate. In an injection event the *PU* first selects a cylinder by activating its sink driver, and then it activates the FPGA by setting *INJ_TTL* for that side of the FPGA to logic high. The *PU* code was implemented and verified by Johan Hansson, Thesis student at Scania, and will not be explained any further in this report. Documents about this implementation are available for Scania personnel only.

At startup, the FPGA configuration registers are loaded from the CPU via the SPI interface. These values specify current rise and fall times, transition times and diagnosis values.

When the FPGA is in idle state, it waits on either one of the INJ_TTL control signals. If a signal turns to logic high, the *Output FSM* is initiated for the corresponding bank. The main task of the *Output FSM* is to control the source and boost drivers. When the source driver is opened, current is allowed through the active actuator and the current level starts to rise. The boost driver controls a voltage source higher than the normal supply voltage, which can be used to raise the current level faster.

The current level is measured by the current sense circuit and monitored by the Schmitt trigger comparators. Each comparator monitors only one current level. The current level is controlled by the corresponding *Refout FSMs* which uses PWM (Pulse Width Modulation) to create the reference current level for the comparator.

When a comparator detects that a current level is reached a signal is sent to the *Output FSM*. This activates the corresponding bank to switch to the next state. The diagnosis, running in parallel with the FSM, also monitors the Schmitt trigger inputs and the timers utilized by the *Output FSM*. If the values are out of range it is indicated in a fault register, which later can be read from the CPU via the SPI-interface.

Because information about the ASIC is confidential and details about this implementation reveals much about the ASIC's functionality, the full details on the implementation are available to Scania personnel only, in document NE06165.

5.4 Implementing the FPGA

The implementation was made according to the design explained above. Some modules are, however, not yet implemented. The time has been the limiting factor of the implementation.

All eight FSMs have been implemented, including the timers, counters and registers needed for their operation. The diagnoses have to a large part been implemented, but there is still no action taken when faults occur. When a fault occurs, a bit specific for that fault is set in the fault register. But there are still fault checks unimplemented. The reason for this is that some fault checks corrupted the output signals of the *Output FSM*, which became obvious after place & route. These fault checks were simply removed, because the troubleshooting would have been too time-consuming. Place & route is explained in 5.4.5.

The PLL has not yet been implemented in the design, because it does not affect the size of the design in whole (there is dedicated PLL logic on the FPGA chip).

The watchdog timer is not essential for the functionality of the FPGA, but it is recommended for the final design. It would be unwise to not have one, in case of a failure on the oscillator signal. The watchdog timer has not yet been created.

The SPI-slave module will possibly be based on a "black box" IP-core, possibly developed by Inicore Corporation. It is not necessary to implement the SPI-slave module in the design to determine its size on the FPGA. Figures of the amount of logic that various IP-cores occupy on different Actel FPGA families are available from Actel [35]. Hence both development time and money is saved. Actel, also, provides their own SPI-module which can be configured as either a master or a slave, but Inicore's SPI-module requires less logical cells [35].

The design has first been developed, simulated and verified for one side of the FPGA, *Bank A*. *Bank B* has then been copied from *Bank A*, renamed and integrated with the rest of the design. There has not been time to simulate and verify both banks together. Only *Bank A* has been verified, but *Bank B* is identical to *Bank A* and should therefore behave identically.

Table 5-1 shows which modules have been implemented and verified. Changes and additions that need to be made are minor and will not greatly affect the number of logic tiles the design occupies on the FPGA.

Table 5-1 Modules in the FPGA Design

Module	**Implemented**	**Verified**
Bank A *Output FSM*	Yes	Yes
A 1st level Refout FSM	Yes	Yes
A 2nd level Refout FSM	Yes	Yes
A 3rd level Refout FSM	Yes	Yes
Interconnection switch	Yes	Yes (with Bank A)
Bank B *Output FSM*	Yes	No
B 1st level Refout FSM	Yes	No
B 2nd level Refout FSM	Yes	No
B 3rd level Refout FSM	Yes	No
PLL	No (it will be based on a SmartGen core)	-
SPI-Slave	No (it will be based on a black box IP-core)	-
Watchdog Timer	No	-

5.4.1 Design Flow and Design Environment

Actel's Libero IDE (Integrated Design Environment) was used during the design process. In Libero, all the EDA (Electronic Design Automation) tools for the design process are included (see Table 5-2) together with a design manager that keeps track of the design flow (Figure 5-4). Each step in the design process will be explained in 5.4.2-5.4.5.

Table 5-2 EDA Tools Included in Libero IDE

Tool	Vendor	Description
HDL-editor	Actel	VHDL or Verilog editor
SmartGen	Actel	Core generator
ViewDraw	Actel	Schematic capture tool
ModelSim	Mentor Graphics	Simulator
WaveFormer Lite	SynaptiCAD	Test bench generator
Synplify	Synopsys	Synthesis
Designer	Actel	Physical implementation

To run Libero IDE a license must be acquired. There are three types of licenses available: Gold, Platinum and Eval. In this project the Gold license, which is free of charge proved to be sufficient. The Platinum license offers more features but must be purchased. The Eval license offers the same features as Platinum but is valid for only 45 days [36].

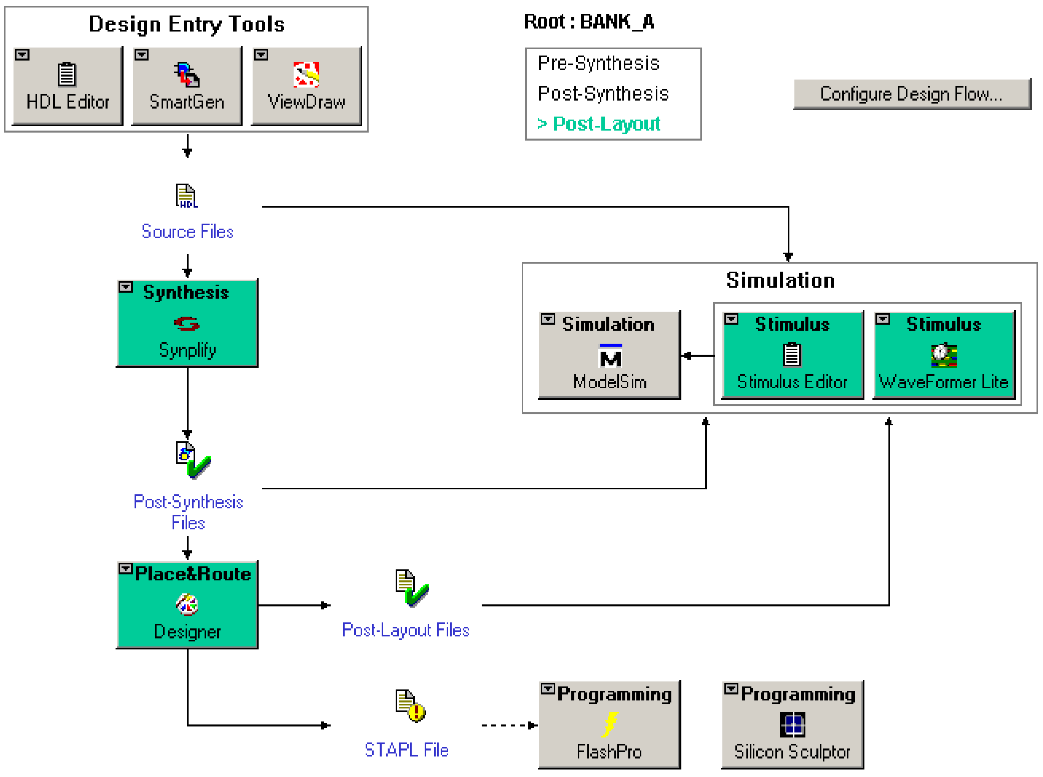


Figure 5-4 FPGA Design Flow In Libero IDE

Each module was coded in VHDL, and then simulated in ModelSim (pre-synthesis simulation). Each module was then verified individually, before being integrated with the rest of the modules. To connect all modules a schematic capture was drawn with the ViewDraw software. The schematic capture was, in turn, simulated (pre-synthesis simulation) and verified in ModelSim. In the next step the design was run through the synthesis software, Synplify, which creates a hardware netlist from logic described in VHDL. After simulating (post-synthesis simulation) and verifying the netlist, it was prepared for physical implementation, using the Designer software. At this point the layout was planned on the FPGA, and resources such as logic cells and I/O (Input/Output) cells were allocated. From this last step it is possible to see exactly how much logic the design occupied on the FPGA. After the place and route, the design was again simulated (post-layout simulation) and verified.

5.4.2 VHDL Design entry and Schematic Capture

Since the FPGA design must be realizable in hardware, all the modules inside the FPGA in (see Figure 5-1) have been modeled with synthesizable VHDL (Very high speed integrated circuit Hardware Description Language) syntax [37] [38]. While developing VHDL code for the design, scalability has been a main issue. The design is built from smaller modules in a structural design style to allow for easy replacement of individual parts in the design. In this approach a module can easily be replaced as long as the new or modified unit has the same set of input and output signals as the old part. In the design environment there is software called SmartGen which can generate a number of simple logic modules, such as registers and counters etc. These modules, or cores, have been used whenever possible since they are based on Actel's own library of hardware components, and hence are optimized for Actel FPGAs. The downside is that these cores only are applicable for the specified Actel FPGA family. But because of the scalability of the design they can easily be replaced with generic VHDL code.

5.4.2.1 Timers

Along with the major modules explained above there is a set of timers. Each timer has been built from a counter, a register and a comparator, that were all created in SmartGen. Conditional statements control the operation of the timer. The timer resolution is dependent on the frequency of the PLL clock signal that is connected to it. The counter value increases on every clock cycle. The register stores the timeout value for the counter, and the comparator checks the value reached by the counter with the register value. In timers used with the diagnoses, an additional register stores the last value reached by the counter. This allows the timer value to be controlled (e. g. when a fault code appears).

5.4.2.2 Schematic Capture

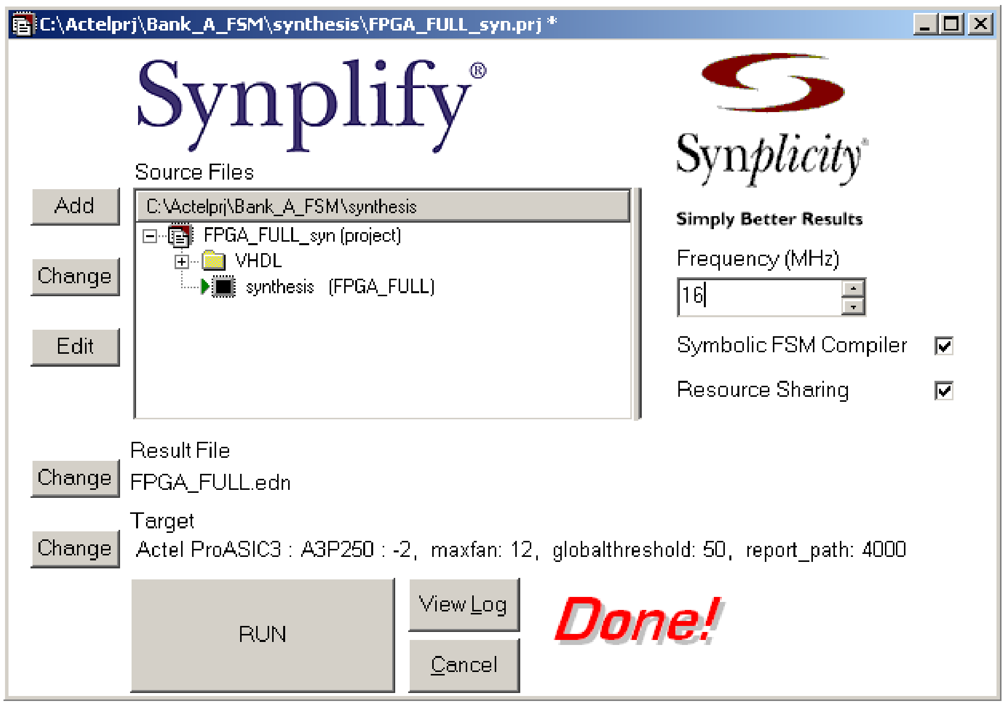
The design was put together with Actel's ViewDraw, which is a schematics tool that creates VHDL code from schematic files. ViewDraw automatically creates a VHDL file with port maps for all components in the design (if the schematic capture is feasible). Unfortunately ViewDraw did create redundant interconnections in VHDL that appeared as warnings during synthesis. These signals were automatically removed by Synplify, and the design was unaffected. To avoid getting these warnings during synthesis the redundant code can be removed or commented away manually.

5.4.3 Simulation

For every different module in the design a unique test bench was created. The same test bench was used for pre-synthesis, post-synthesis and post-layout simulation. It was created with the WaveFormer Lite software. When creating a test bench in WaveFormer Lite, a stimulus file is first created. The stimulus file contains waveforms that are intended as input signals to the test object. The stimulus file is then exported as a VHDL test bench, to be used in ModelSim. Generally, only typical input waveforms have been tested. There has not been enough time to create test benches that cover every possible event that can occur. Nor has it, during simulation, been possible to analyze all internal signals of the simulated unit. The main focus has been to verify that the output signals, of the design, behave as expected.

5.4.4 Synthesis

After the schematic capture was created the design was synthesized in Synopsys Synplify, where the hardware netlist was created. In Synplify, the option: *Symbolic FSM compiler* was enabled, which means that Synopsys recognizes state machines in the design and optimizes them. *Resource Sharing* was also enabled, meaning that resources are shared, to optimize the chip area of the FPGA. The frequency was set at 16 MHz. Figure 5-5 shows the project view in Synplify where these choices were made. All other settings were default.


C:\Actelprj\Bank_A_FSM\synthesis\FPGA_FULL_syn.prj *

Source Files
C:\Actelprj\Bank_A_FSM\synthesis
FPGA_FULL_syn (project)
VHDL
synthesis (FPGA_FULL)

Frequency (MHz): 16
Symbolic FSM Compiler ✓
Resource Sharing ✓

Result File
FPGA_FULL.edn

Target
Actel ProASIC3 : A3P250 : -2, maxfan: 12, globalthreshold: 50, report_path: 4000

Figure 5-5 Synplify Project View

Figure 5-6 shows, as an example, the netlist generated in synthesis for one of the FSMs. When the design passed the synthesis it was simulated again in ModelSim (post-synthesis simulation) and verified. The identical test bench was used for both post-synthesis simulation and pre-synthesis simulation.

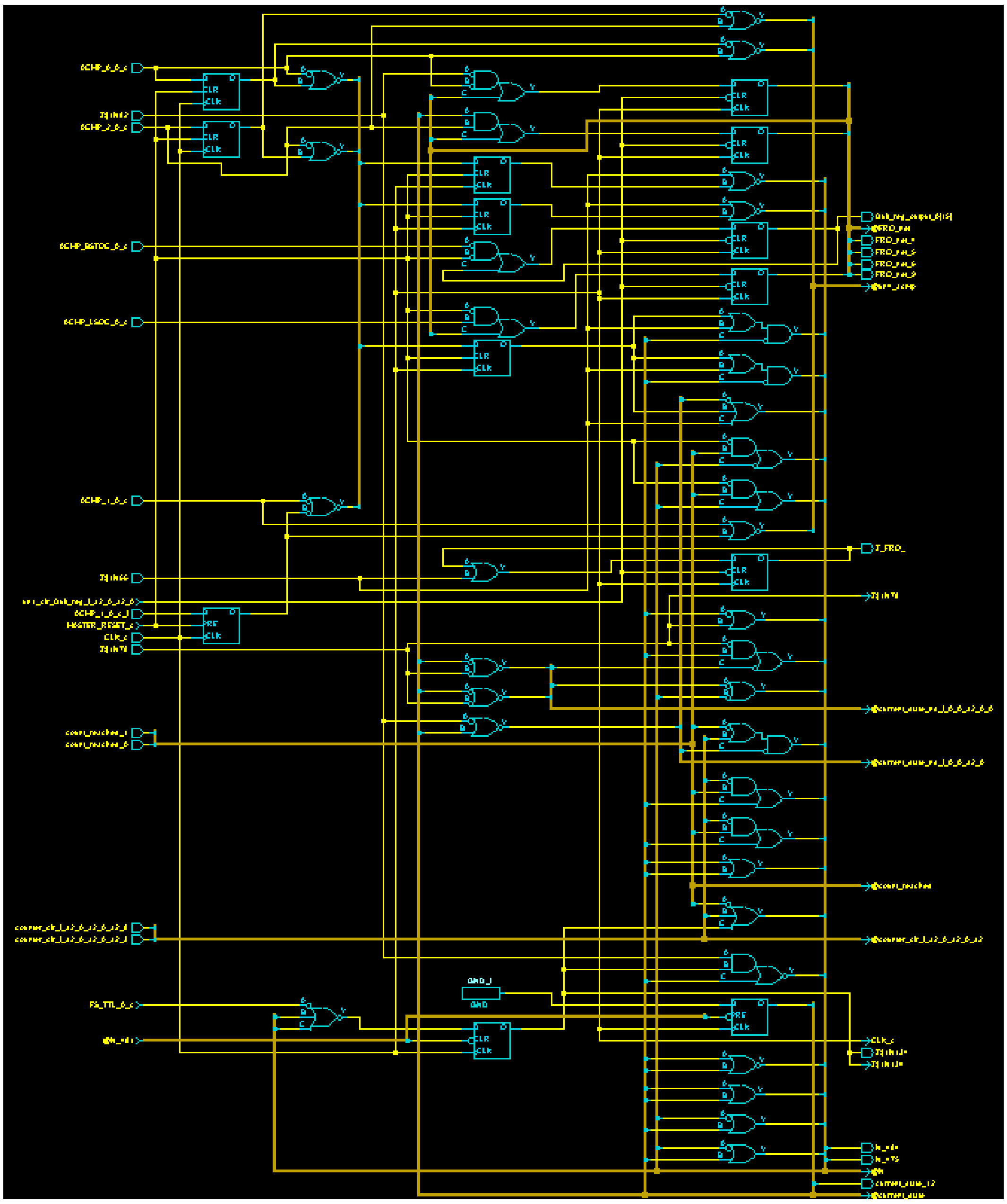


Figure 5-6 Example of Generated Netlist for One of the FSMs

5.4.5 Place & Route

Following the synthesis, the netlist was prepared for placement on the FPGA with Actel's Designer software. Designer optimizes the netlist, checks for errors and verifies that the design fits into the selected device. The chip layout is affected by the settings and constraints, which are set in Designer. LVCMOS 3.3V was chosen as the default I/O standard. The junction temperature range was specified to: -40°C to +150°C. All other constraints and settings were default. The goal here was to determine the required FPGA size, and get an idea of how much logic the netlist occupies on the FPGA die. The design can, and will most certainly be modified in the continuation of this project, and therefore no time was spent on optimization. ProASIC3 A3P250 144 FPGA was the smallest FPGA that the design could be implemented in. The resulting layout can be viewed in Figure 5-7, which is a view from the Designer software. Filled squares in Figure 5-7 represent utilized tiles.

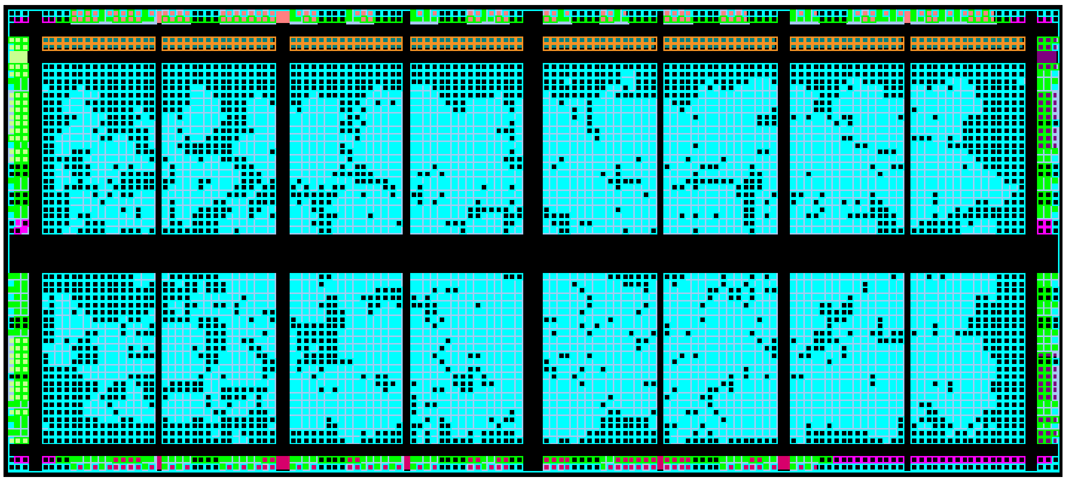


Figure 5-7 Tiles Occupied by the Design in the FPGA

From the generated log file, it is possible to see exactly how much of the FPGA's resources were utilized, see Table 5-3. The core logic was utilized to 63 %. Global (chip + quadrant) shows that 11% of the routing resources were utilized. When implemented, the PLL will not occupy any core cells, since it has its own dedicated logic cells. No Differential I/O ports, or RAM/FIFO, or FlashROM cells have been used in the design.

Table 5-3 Resources Utilized in ProASIC3 A3P250, 144 FBGA

FPGA utilities	Used	Total	Percentage used
CORE	3872	6144	63.02%
IO (W/ clocks)	77	97	79.38%
Differential IO	0	24	0.00%
GLOBAL (Chip+Quadrant)	2	18	11.11%
PLL	0	1	0.00%
RAM/FIFO	0	8	0.00%
FlashROM	0	1	0.00%

An estimated 80 % to 90 % of the design has been implemented. Inicore's SPI-Slave module occupies 128 tiles [35] which would give a total utilization of 4000 tiles (65.10 %). Actel's Core-SPI module occupies 328 tiles [35] and gives a total of 4200 tiles (68.84 %). It can however be configured as slave only, occupying 142 tiles [39], which would give a total of 4014 tiles (65.33 %).

It is not likely that the remaining circuits will occupy any substantial amount of logic cells. Still to implement are: the WDT, which probably will be based on a counter, a comparator, a register and a few flip-flops; and the remaining diagnoses, which would use a few flip-flops and registers. Originally most of the diagnoses were implemented, and behaved as expected during pre-synthesis and post-synthesis simulation. During post-layout simulation, however, the output signals were corrupted. One of the likely reasons for this is that no timing constrains were set in Designer, during place & route. After removing some diagnoses the output signals were correct again.

The utilization of I/O cells is high mainly because the SPI-slave not yet has been implemented. Because of this, two 16 bit buses[6], one 8-bit bus, and two control signals were routed directly to I/O cells instead of routed to the SPI module. This alone occupies 42 I/O cells. The SPI-slave will use only four I/O cells, for the *miso*, *mosi*, *sck* and *ss_n* signals (see Appendix B). Moreover since the PLL is not yet implemented there are three input clock signals instead of one. The PLL will use only one or possibly two I/O cells. By implementing these two modules the number of occupied I/O cells can be reduced to a total of 37 (instead of 77), therefore a smaller FPGA package will be required.

The post-layout simulation showed that the output signals were unaffected by the place & route. This was however only tested for Bank A. There was no time to create a test bench for the whole design, or to simulate and verify it. Bank B is however identical to Bank A, and should have identical output signals, if the same test bench is used for both banks. The only other change made in the design is the Interconnection Switch, which must handle communication signals to and from both banks. It is therefore the double size, when both banks are implemented, compared to when only one bank is used.

[6] These buses are temporarily 16-bit. To match the input pins on the SPI-slave module they must be changed to 8-bit.

Chapter 6 Discussion

In section 6.1 the conclusions from the project are discussed. In 6.2 further areas of work are suggested.

6.1 Conclusions

6.1.1 Flash FPGAs

Flash FPGAs are currently the only FPGAs that are non-volatile, live-at-power-up, reprogrammable, and tolerant to the extended automotive temperature range ($T_{Ambient}$ = -40°C to +125°C). Actel is the only developer of flash FPGAs.

High temperatures affects the performance retention time of the FPGA (Table 6-1). This may be the greatest weakness of the flash technology for this application. Actel does not guarantee the FPGA's performance if the performance retention time is exceeded. Therefore the FPGA must be reprogrammed within the performance retention time [41], either manually or automatically. Actel automotive flash FPGAs are only classified for $T_{Junction}$ in the range -40°C to +125°C. But Actel can perform custom tests, for large sales volumes [31], if required[7].

Table 6-1 Performance Retention Time

Junction Temperature 100 % of the time	Minimum Performance Retention Time
110°C	20 years
125°C	10 years
135°C	5.0 years
150°C	2.5 years

6.1.2 The Implementation

The implementation, accounted for in 5.4, shows that it is possible to implement the current control circuit in an Actel, ProASIC3 A3P250, flash FPGA. An exact price quote has not been available for this FPGA, but an estimate in the range of $5.00 USD has been given by both Acal and Actel, for sales volumes of 50k units [42] [43].

6.1.3 The Design

Dividing the circuit into two modules (the *PU* and FPGA), as explained in 5.2, is eligible if it allows the design to fit into a smaller FPGA. The only other advantage gained by doing this, would be that fewer control signals are required between the two components. A disadvantage of having the current control divided into two units is that it is difficult to overview, particularly since one part is implemented in software (the *PU*) and the other part in hardware (the FPGA).

If the FPGA size is large enough, it is probably better to move everything implemented in the *PU* to the FPGA, and let the FPGA have full control of the actuators. Adding control of the sink drivers to the FPGA does not require any major changes of the design. Possibly one or two more states will be needed in the *Output FSMs* to determine that one, and only

[7] Flash memory cells currently used in the ECU are only classified for T(junction)= -40°C to + 125°C [40]

one, cylinder is selected at a time. The sink driver output is just another signal assignment in each state of the FSM. The diagnoses (implemented in the *PU*) will also require a few conditional statements to be coded in VHDL, but it is convenient to have all the diagnoses for the injection event gathered at the same place (rather than shared between the *PU* and the FPGA).

6.1.4 Soft Kernels

Another Thesis project, performed in parallel with this Thesis [44] evaluated current control with processor based solutions. One of the conclusions was that real time requirements were difficult to meet. One option, which has not been evaluated, is a soft CPU kernel as a module inside the FPGA. With this approach, parts of the design that are not timing critical can be implemented in software for the soft kernel, while timing critical parts can be implemented as hardware logic modules, in a similar fashion as the implementation was made in this Thesis. If implemented wisely, no internal modules within the FPGA would need to be reprogrammed, when changes are made in the sequence of the injection event. Such changes could all be made in software, coded for the soft kernel.

The ProASIC3 family has the option of using an ARM7, 16 or 32-bit, CPU as a soft kernel. The ARM7 kernel occupies around 6000 tiles and the smallest ProASIC3 that allows this option is the A3P250, which then is called M7A3P250. An ARM7 CPU kernel implemented in this FPGA would occupy all of its core logic and hence no other modules could be implemented. For this project the next size of FPGA, M7A3P400, would be required. M7A3P400 has 9,216 logic tiles which would leave approximately 3000 tiles for other modules, when the kernel is implemented.

Even though Actel provides the ARM7 kernel without any licensing fees or royalties [45], it is questionable as to whether the cost would stay close to $5.00 USD per unit, considering the added cost of a larger FPGA. Besides, more development time would be required for the FPGA because of the added complexity of the system. This solution should be considered, only if it is found desirable to configure the FPGA in software.

6.1.5 Mixed Signal FPGAs

The added flexibility that an FPGA provides only applies to the digital circuits, which are easily configured in an FPGA. In the ECU, however, there are also analog circuits around the FPGA, such as the current sense circuits, the driver circuits and others. These analog circuits cannot be changed without replacing the components. To change the analog circuits in the ECU, the whole PCB (Printed Circuit Board) would have to be replaced.

An alternative not investigated in this thesis is the use of a mixed signal FPGA with support for both analog and digital signals. Actel is the only producer of this kind of FPGA, which they call Fusion. While the smallest Fusion, AFS090, is in the $5.00 USD segment, for large volumes, it only has 2,304 core tiles. Since the implementation in this Thesis required 3,872 core tiles, the AFS090 is not feasible unless it is possible to optimize the logic very much. The next size of FPGA is the AFS250, which has the same amount of core tiles, 6,144, as the A3P250. This device will cost approximately $12 USD/unit, for large volumes [46]. This could be an interesting alternative, depending on how much of the system's analog circuits it can implement. Fusion further eliminates the need of an external oscillator, since it is provided on the chip.

6.2 Further Areas of Work

- To finish the design, all modules in Table 5-1 should be implemented, simulated and verified thoroughly. Modules that have not been implemented are the PLL, SPI-Slave and the Watchdog timer.
- The PLL is easily generated in SmartGen but it should be determined if and how the *powerdown* and *lock* signals should be used in the design.
- The watchdog timer is explained in the description of the ASIC. A watchdog module for the FPGA can be modeled with this description as a basis.
- In the current implementation all timers use the same clock signal (with the exception of one timer that has been given its own clock signal). The FPGA will however require timers with different resolutions. For this purpose, different clock frequencies will be needed. As a guideline, the timer resolutions of the ASIC can be viewed [27]. Some timers used in the ASIC require lower frequencies than the PLL can create. This can either be solved by: dividing down the clock signal, with a module that counts clock pulses; or by setting higher timeout values for the timers.
- The registers should be optimized. In the current design all implemented registers are 16-bit, and every configuration value has its own register. In the ASIC the configuration values often require less than 16-bits, and one register is often used for configuration of several values. It should be investigated if the RAM/FIFO cells on the FPGA can be used to replace some or all of the registers. If this is possible the required number of core logic cells on the FPGA can be reduced.
- It should be determined if the register values should be loaded through the SPI interface or if it is possible to use the internal FROM (Flash Read Only Memory) in the FPGA to store the configuration values. The FROM can only be written through the JTAG interface, but it can be read from the FPGA.
- A decision should be taken if the SPI-Slave module should be based on Inicore's module or Actel's module, or a module from another developer. It may be worthwhile for Scania to develop an SPI-Slave module. In any case this module should be integrated with the rest of the design.
- There are filters in the ASIC to reduce the noise susceptibility. These still need to be implemented in the FPGA.
- It is necessary to create a set of test benches that thoroughly tests the design. Also it must be verified that the design functions perfectly, before synthesis, after synthesis and after layout.
- During synthesis and layout it should be determined which constraints can be used to optimize the design for area.
- During "place & route" it is necessary to set timing constraints for the design to function properly.
- Actel devices have support for encryption of the configuration data and this should be used to protect the design.
- An FPGA Starter Kit should be invested in so that the design can be downloaded into an actual FPGA that can be tested with the CPU and *PU* hardware.

References

[1] Scania, *XPI Training*, 2005

[2] Scania, *PDE*, 2002

[3] G Blomberg, *Scania Projekt DT12*, Scania, November 2000

[4] Horst Bauer, Karl-Heinz Dietsche, Jürgen Crepin, Folkhart Dinkler, *Bosch Automotiv Handbook*, 5th edition, Robert Bosch GmbH, September 2000, pp. 535-551

[5] Ola Dovnäs, *TB1673 Scania S7, Electronic Control Unit*, June 2005, pp. 3-5

[6] Horst Bauer, Karl-Heinz Dietsche, Jürgen Crepin, Folkhart Dinkler, *Bosch Automotiv Handbook*, 5th edition, Robert Bosch GmbH, September 2000, pp. 126

[7] Ola Dovnäs, *TB1673 Scania S7, Electronic Control Unit*, June 2005, pp. 45-48

[8] Motorola, *Cummins Ten Cylinder Fuel System ASIC*, November 2003, pp. 56-85

[9] Ola Dovnäs, *TB1673 Scania S7, Electronic Control Unit*, June 2005, pp. 42

[10] Motorola, *Cummins Ten Cylinder Fuel System ASIC*, November 2003, pp. 94-103

[11] John F. Wakerly, *Digital design: principles and practices*, 3rd edition, Prentice Hall, 2001, pp. 12-17

[12] Michael John and Sebastian Smith, *Application Specific Integrated Circuit*, Addison Wesley, 1997, pp. 1-18

[13] Michael John and Sebastian Smith, *Application Specific Integrated Circuit*, Addison Wesley, 1997, pp. 169-176

[14] Karen Parnell and Nick Mehta, *Programmable Logic Design: Quick Start Handbook*, Xilinx, April 2004, pp. 1-20

[15] Neil Storey, *Electronics: a systems approach*, 2nd edition, Addison Wesley Longman Ltd, 1998, pp. 528-550

[16] John F. Wakerly, *Digital design: principles and practices*, 3rd edition, Prentice Hall, 2001, pp. 872-873

[17] Michael John and Sebastian Smith, *Application Specific Integrated Circuit*, Addison Wesley, 1997, pp. 191-223

[18] Michael Barr, *How Programmable Logic Works*, http://www.netrino.com/Articles/ProgrammableLogic/, June 1999, (checked April 16th, 2006)

[19] Actel Corporation, *Live at Power-up: PLD effects on System Design*, August 2005

[20] Michael John and Sebastian Smith, *Application Specific Integrated Circuit*, Addison Wesley, 1997, pp. 714-736

[21] Application Note, *In-System Programming (ISP) in ProASIC3/E Using FlashPro3*, Actel Corporation, January 2005

[22] Claes Jennel, *Test och testbarhet*, compendium, KTH Ingenjörsskolan, August 2000, pp. 40

[23] Application Note, *Using JTAG Boundary-Scan with ProASIC™ 500K Devices*, Actel Corporation, November 2000

[24] Tom Williamson, *Designing Microcontroller Systems for Electrically Noisy Environments*, Intel, December 1993, pp. 18-19

[25] Application note, *Package Thermal Characteristics*, page 1, Actel Corporation, February 2005

[26] Robert L. Boylestad, *Introductory Circuit Analysis*, 9th edition, Prentice Hall, 2000, pp. 473-474

[27] Motorola, *Cummins Ten Cylinder Fuel System ASIC*, November 2003, pp. 112-147

[28] Motorola, *Cummins Ten Cylinder Fuel System ASIC*, November 2003

[29] ST Microelectronics, *Flash Memories*, June 2003, http://www.st.com/stonline/ press/ news/back2005/b979h.htm, (checked March 30, 2006)

[30] Lattice Semiconductor Corporation, *Non-Volatile Reconfigurable FPGA*, http://www.latticesemi.com/products/fpga/xp/nonvolatilereconfigurabil.cfm, (checked March 30, 2006)

[31] Zaf Mahmood, FAE – Northern Europe, Actel, Acal seminar in Electrum, Kista April 4, 2006

[32] Actel Corporation, *Automotive Solutions: FAQ*, February 2004, pp. 3

[33] Actel Corporation, *Direct C: v1.3 for ProASIC3/E Devices User's Guide*, March 2006

[34] Actel Corporation, *Programming: DirectC*, http://www.actel.com/products/tools /directc/index.html, (checked March 27, 2006)

[35] Actel Corporation, *IP Solutions: Improve Time-to-Market and Reduce Design Risk*, 2005

[36] Actel Corporation, *Libero IDE , Designer, and PALACE License Types*, http://www.actel.com/products/tools/SWlicenses.html, (checked April 3, 2006)

[37] Peter J. Ashenden, *The Designer's guide to VHDL*, 2nd edition, Morgan Kaufmann Publishers, 2002, pp. 639-654

[38] Michael John and Sebastian Smith, *Application Specific Integrated Circuit*, Addison Wesley, 1997, pp. 593-613

[39] David Källberg, FAE – Semiconductors, ACAL AB, email, April 7, 2006

[40] Mats Henriksson, FAE, Freescale, email, April 12, 2006

[41] Actel Corporation, *ProASIC^{PLUS} Flash Family FPGAs*, February 2004, pp. 33-34

[42] Per Andersson, Sales Engineer Semiconductors, ACAL AB, conference at Scania, Södertälje November 2, 2005

[43] Göran Rosén, Nordic Sales Manager, Actel Europe Ltd, conference at Scania, Södertälje, November 2, 2005

[44] Johan Hansson, *Current Control and Diagnosis of XPI fuel Injector Pilot Valves*, Uppsala Universitet, April 2006

[45] Actel Corporation, *Core MP7*, http://www.actel.com/products/arm7/, (checked July 16th, 2006)

[46] David Källberg, FAE – Semiconductors, ACAL AB, email, August 14, 2006

Appendix A The Current Pulse

The Current pulse was measured with an oscilloscope when a Cummins XPI injector was connected to the ASIC's high- and low- side driver circuits. Figure A-1 shows that the hysteresis fall time, during pull-in level, is approximately 18 µs. This is the shortest interval in which the ASIC output signal, controlling the high side driver circuit, will have to switch from low to high.

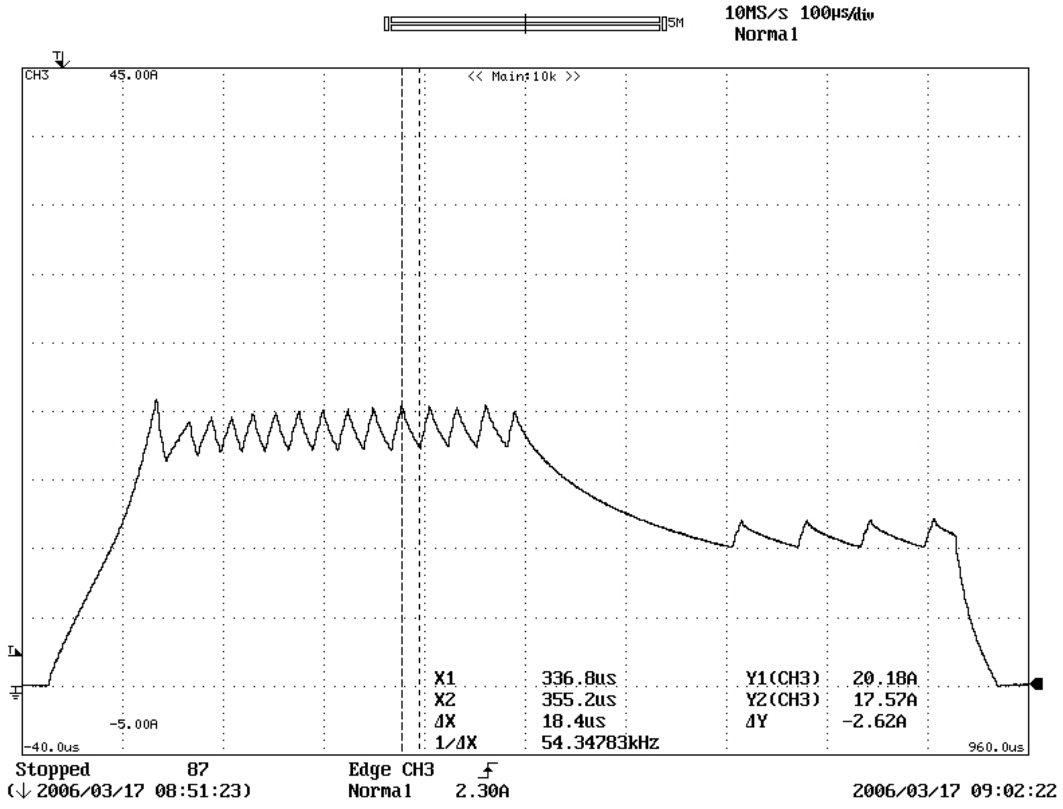
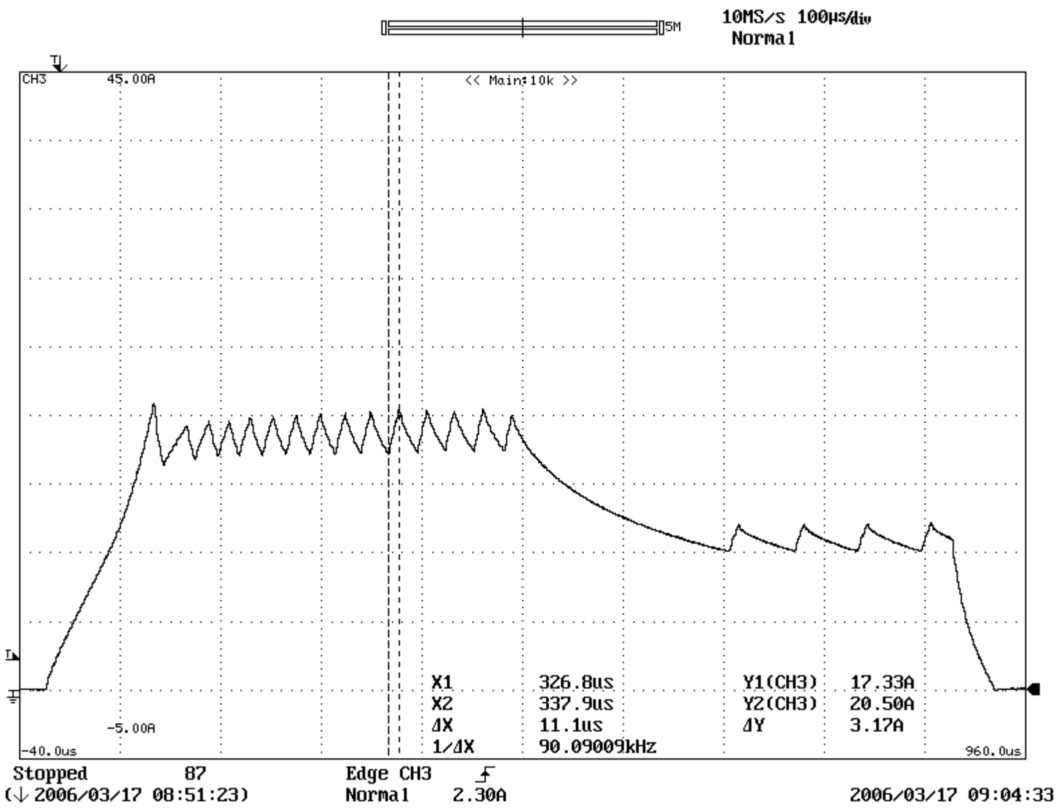


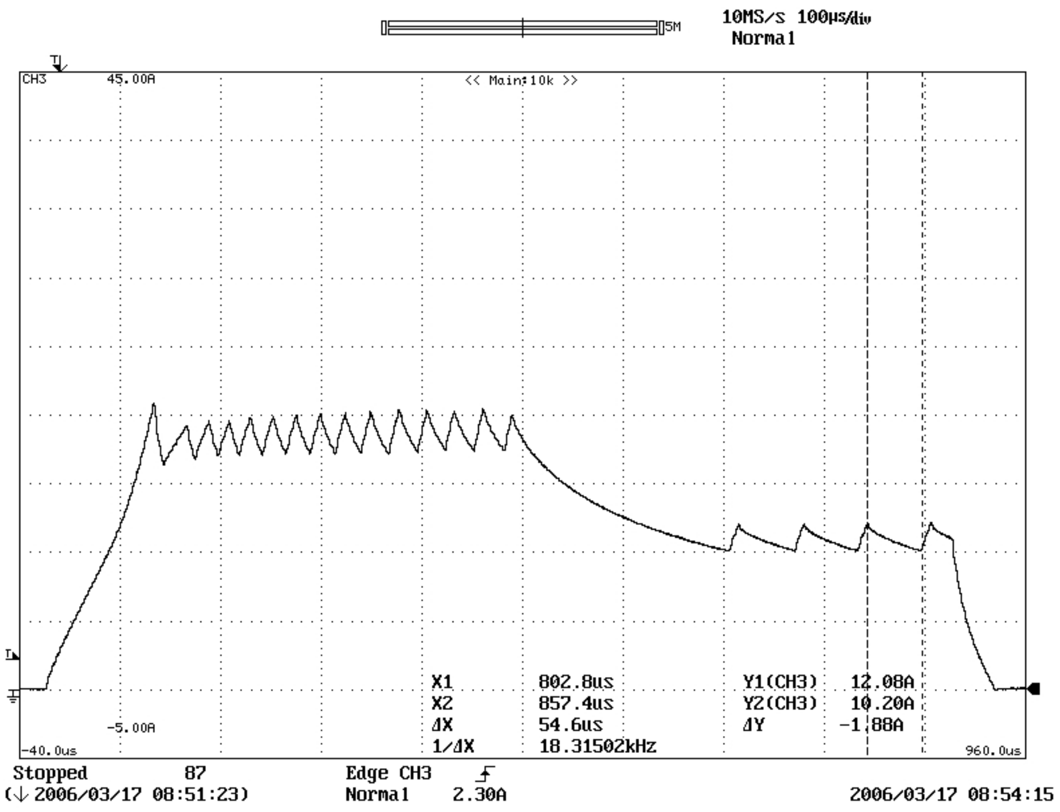
Figure A-1 Pull-in Level, Peak to Valley Hysteresis

Figure A-2 shows that the hysteresis rise time, during pull-in level, is approximately 11 µs. Figure A-3 shows that the hysteresis fall time, during hold level, is approximately 55 µs. Figure A-4 shows that the hysteresis rise time, during hold level, is approximately 9 µs. This is the shortest interval in which the ASIC output signal, controlling the high side driver circuit, will have to switch from high to low.



```
X1      326.8us          Y1(CH3)   17.33A
X2      337.9us          Y2(CH3)   20.50A
ΔX      11.1us           ΔY        3.17A
1/ΔX    90.09009kHz
```

Figure A-2 Pull-in Level, Valley to Peak Hysteresis



```
X1      802.8us          Y1(CH3)   12.08A
X2      857.4us          Y2(CH3)   10.20A
ΔX      54.6us           ΔY        -1.88A
1/ΔX    18.31502kHz
```

Figure A-3 Hold level, Peak to Valley Hysteresis

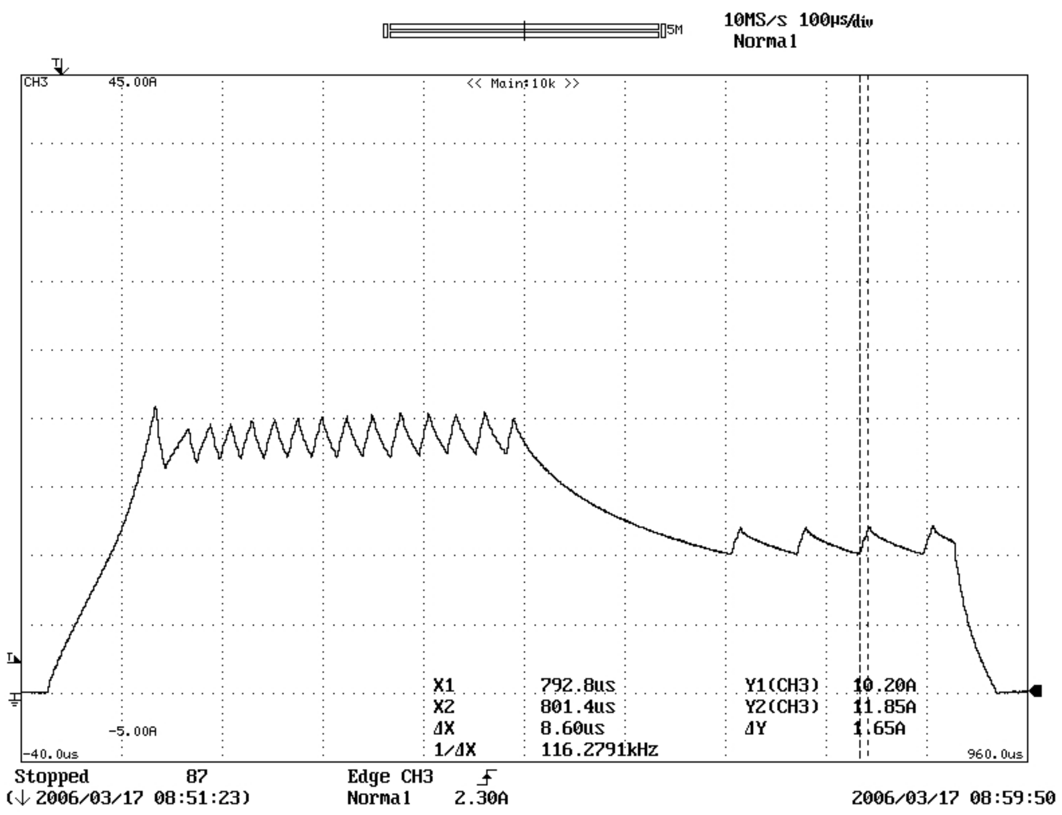


Figure A-4 Hold level, Valley to Peak Hysteresis

Appendix B The SPI-Bus

The SPI (Serial Peripheral Interface) is a four wire serial communication interface, which was established by Motorola. When two devices communicate on the SPI-bus, one is referred to as the "master" and the other as the "slave". Communication is synchronous, and operates in full duplex (both directions). It is initiated as well as controlled by the master. Table B-1 shows the four SPI communication pins[8].

Table B-1 The SPI pins

SS	–	Slave Select
SCK	–	Serial Clock
MOSI	–	Master Out Slave In
MISO	–	Master In Slave Out

The master selects which slave to communicate with, by setting the corresponding "slave select" (SS) signal to logic low. When the SS signal is logic high, the slave ignores the system clock, and sets its MISO (output) pin to high impedance. The number of slaves that can be connected to the SPI-bus is only limited by the number of SS output pins on the master device. Each Slave device on the SPI bus requires a separate SS output pin on the master device[8] (see Figure B-1).

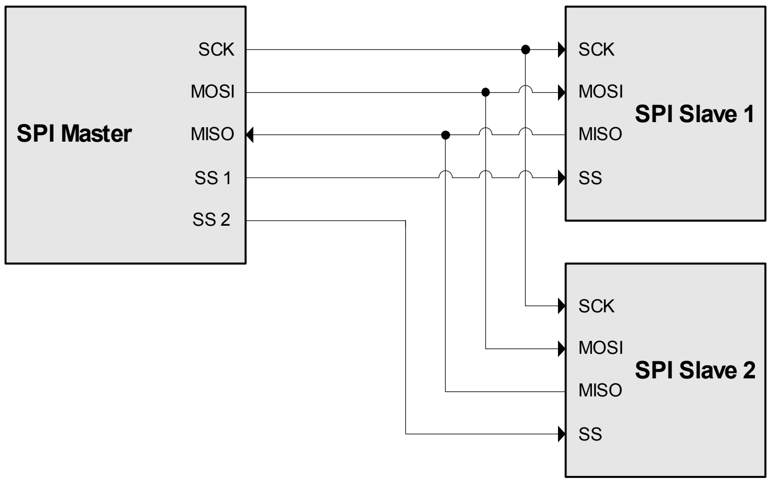


Figure B-1 Master and Slave(s) on the SPI bus

[8] Motorola, M68HC11 Reference Manual, 1996, pp. 8.1-8.22

Appendix C The Actel ProASIC3 Flash FPGA

This is a brief introduction to the Actel ProASIC3 Flash FPGA, which was utilized in the Thesis. For a full and detailed view of all the features available in this FPGA family please refer to the datasheet[9].

C.1 Flash Technology in ProASIC3

The ProASIC3 chip is covered with flash switches (Figure C-1) which are used for programming of the device. Each Switch is based on two transistors, which share a floating gate. The floating gate stores the configuration of the switch. One of the transistors is called the sensing transistor. It is used for writing, and verification of the floating gate voltage. The other transistor is called the switching transistor. It is used to connect, or separate routing nets; and to erase the floating gate.

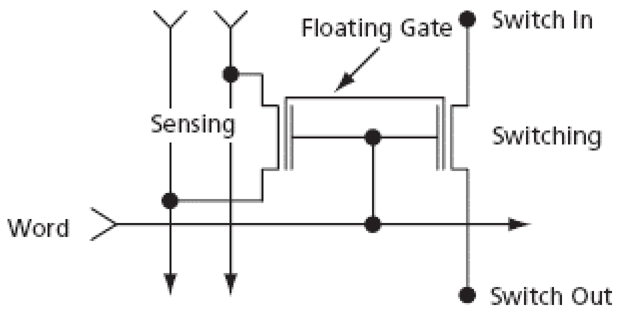


Figure C-1 ProASIC3 Flash-Based Switch

C.2 Device Architecture

Figure C-2 shows the architecture of the ProASIC3, A3P250 device, which was utilized in this Thesis. This device is equipped with six Clock Conditioning Circuits (CCCs) that each can implement up to three global clock buffers, and can create phase shifts and delays. A phase locked loop (PLL) can alternatively be implemented in one of the CCCs. The PLL can be used for frequency division/multiplication, and can produce up to three global clock signals.

On chip the ProASIC3 FPGA has embedded blocks dedicated for SRAM/FIFO operation. A FROM (Flash Read Only Memory) of 1kbit is also available on chip, which can be utilized for non-volatile storage. The FROM can only be written to from the IEEE 1532 JTAG interface, but can be read from the internal logic of the FPGA.

[9] Actel Corporation, *ProASIC3 Flash Family FPGAs*, *Advanced v.05*, January 2006

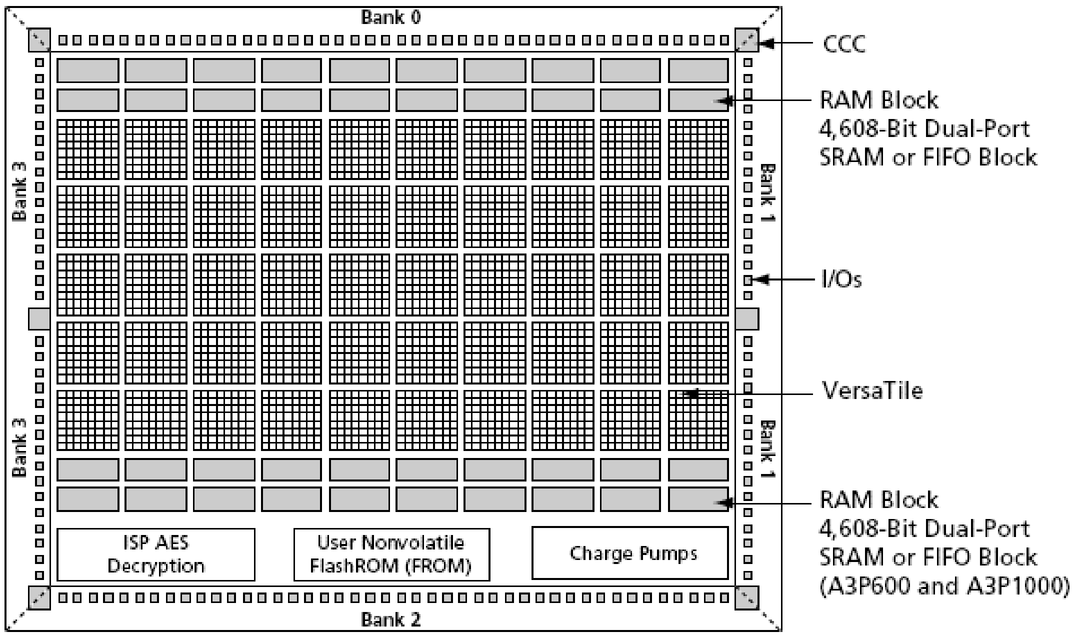


Figure C-2 A3P250 Device Architecture

The routing structure is designed in four different levels: ultra-fast local lines, efficient long-line resources, high-speed, very long-line resources, and the high-performance VersaNet global network (see Table C-1).

Table C-1 Routing Resources

Ultra-fast local lines	Connects the output of each VersaTile directly to the inputs of the adjacent VersaTiles.
Efficient long-line resources	Spanning over 1, 2 or 4 VersaTiles, runs both horizontally and vertically and cover the entire FPGA area
High-speed, very long-line resources	Spanning the entire device, with minimal delay, for high fan-out nets of 12 VersaTiles in the vertical direction, and 16 VersaTiles in the horizontal direction
High-performance VersaNet global network	Accessible from both external pins and internal logic. It is typically used for clock, reset and other high-fan-out signals which require minimum skew

The largest chip area is occupied by the core logic cells, which are called VersaTiles. These are the basic logic cells that can be defined as: combinatorial devices, such as AND, OR, NAND, NOR, XOR; or as sequential devices, such as flip-flops and latches. Figure C-3 shows the architecture of a VersaTile cell.

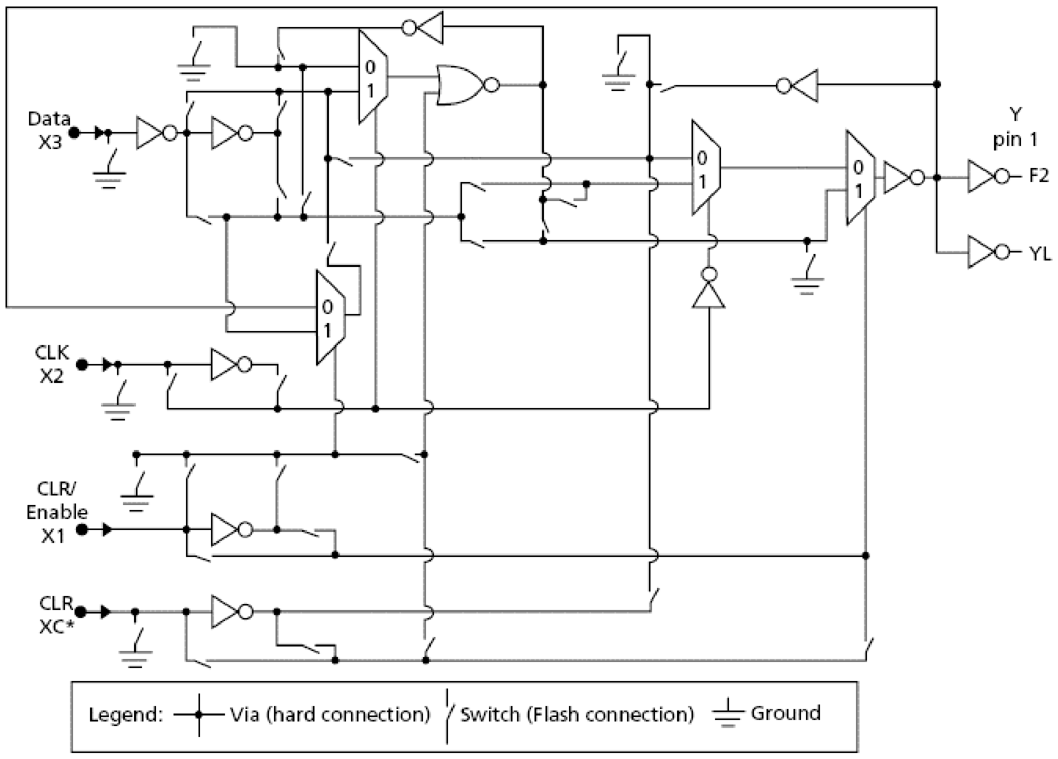


Figure C-3 ProASIC3 Core VersaTile