FPGA Core Specification

Author: Marko Mlinar
marko.mlinar@campus.fri.uni-lj.si
Damjan Lampret
damjan.lampret@yahoo.com

Rev. 0.6
April 5, 2001

Revision History

Rev.	Date	Author	Description
0.1	13/2/01	Marko Mlinar	First Draft
0.2	20/2/01	MM	More detailed description, more pictures.
0.3	22/2/01	MM	Added legal status
0.4	15/3/01	MM	More routing details
0.5	3/4/01	MM	Bitstream, routing
0.6	5/4/01	MM	Major architecture update, more detailed description, removed preliminary status

1

Introduction

1.1 What is FPGA?

Field-Programmable Gate Arrays (FPGAs) are flexible and reusable high-density circuits that can be (re) configured by the designer, enabling the VLSI design/validation/ simulation cycle to be performed more quickly and cheaply.
The flexibility provided by FPGAs cause a substantial performance penalty due to non-specialized circuit design and signal delay through the programmable routing resources, compared do ASIC designs but FPGAs are still 1000 times faster than circuit simulators.

1.2 FPGA Core Introduction

This core provides plural of high-speed programmable logic. This FPGA has regular structure and consists of two (re) configurable elements: Look-Up-Tables (LUTs), each with 5 inputs and one output and Input-Output Cells (IOCs). It logic size is approximately 2500 gates. The development system offers fully automated logic placement and routing (more about P&R software can be found in FPGA P&R Software document). Every function is stored in static memory, called LUT, during programming phase. Also connections are established to match desired schematics. Programming data should be supplied by any external data source, e.g. main memory, disk or processor built to programming circuit.

NOTE: This version does not support multiple FPGA connection, but FPGA design can be easily adopted, connecting port registers in Input Output Logic module. There is also no tri-state support.

1.3 Legal Status

Although this novel architecture differs from current ones, it may infringe with following patents:
(If you are aware of any other patents or you think these are not of issue for this architecture, please email authors.)

US5760604: Interconnect architecture for field programmable gate array
(Longer lines - those not connecting adjacent units)
US5260611: Programmable logic array having local and long distance conductors
(Clock signals, carry chains between SPCs)
US5744980: Flexible, high-performance static RAM architecture for field-programmable gate arrays
(SRAM LUTs, interconnect architecture, wiring channels)
US4758745: User programmable integrated circuit interconnect architecture and test method
(Interconnect architecture, wiring channels)

(This page is intentionally left blank)

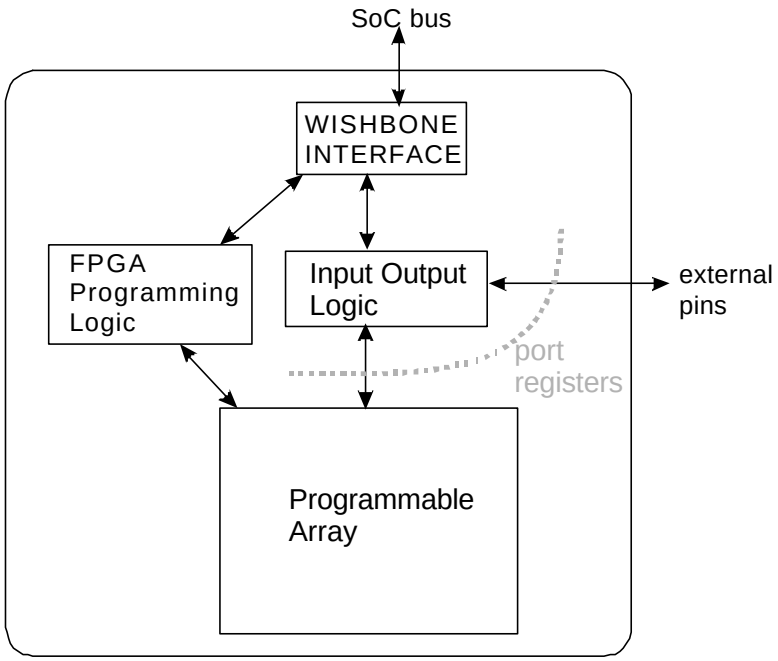
2

Architecture

2.1 Overview

FPGA is connected to WISHBONE SoC bus, and on the other hand it also has access to (multiplexed) chip pins.

Figure 1: FPGA Core Architecture Overview



This architecture implementation does not have special dedicated cells (for e.g. adders, MUXes…) which would shorten delay time and increase silicon utilization for common circuits. Those features were cutout because of simplicity reasons, and will probably be included in future implementations.

2.2 Programmable Array

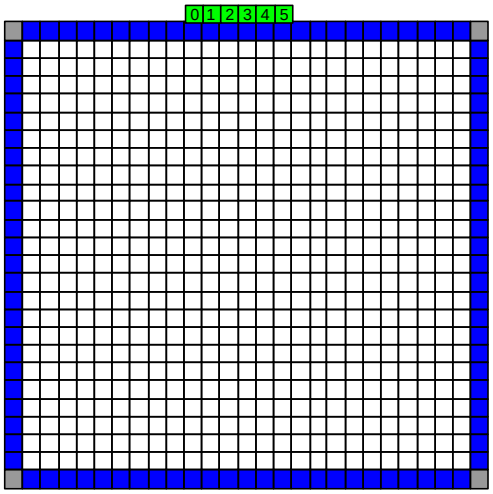
Programmable array consist of 26x26[1] array of elements, as shown on Figure 2:

- GPC (General Purpose Cell), total count 576
- IOC (Input Output Cell) on perimeter, total count 96
- SR (Special Resources) outside array, total count 6

Programmable perimeter of IOCs allows inner programmable array to communicate with input/output logic. Logic functions are implemented using GPCs. Cells are connected by unidirectional metal segments, which can be routed inside each cell and form flexible interconnecting network. All these connections are established by software. Special resources are explained in detail in chapter 2.2.3 Special Resources.

Programmable cells can be programmed only using programming circuitry (column scan chains). Columns can be randomly accessed and can be reprogrammed on the fly.

Figure 2: Array (26x26), blue blocks represent IOCs, white GPCs (LUTs) and green special resources (SR0-SR5, see Table 5)



[1] Such FPGA size was chosen to fit into 6mm^2 silicon area, using standard cell process 0.25µm. Performance and size can be improved using custom design techniques.

2.2.1 General Purpose Cell - GPC

This multipurpose cell is automatically programmed during programming phase, and serves as a general function with up to 5 inputs[2] (selected out of 8 neighbours) and one output. It also has some routing capabilities - it allows pass trough to each of eight outputs from any of eight inputs.

Figure 3: Two different functions of a GPC

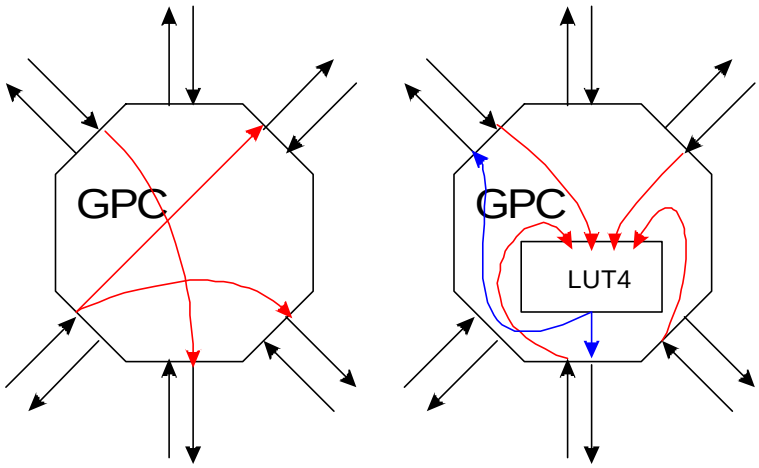
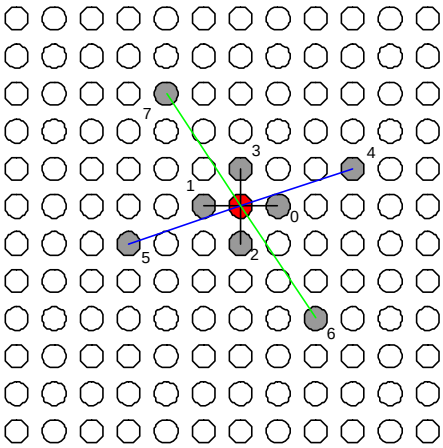


Figure 4: GPC's neighbours and their indexes



Neighbours are shown on Figure 4[3]. Longer connections were chosen, to decrease delays and increase silicon utilization (e.g. wire with length 3 would require 3 wires of length 1, thus we would require up to three times the channel width). Cells near the perimeter do not have all the neighbours available, so their inputs are set to zero. Note that IOC can be a neighbour of GPC.

[2] Many studies indicate that LUTs with 4-6 inputs are most suitable for FPGAs
[3] See R. Trobec, Two-Dimensional Regular *d*-Meshes, Parallel Computing

As shown on Figure 5, any five signals, D[0:4], can be selected from I[0:7]. It can be easily shown that this selection is always possible[4].
Configuration memory is linked into scan-chain. CF0 is connected to previous cell's CF71 and CF71 is connected to next cell's CF0.

Figure 5: General-Purpose Cell schematic

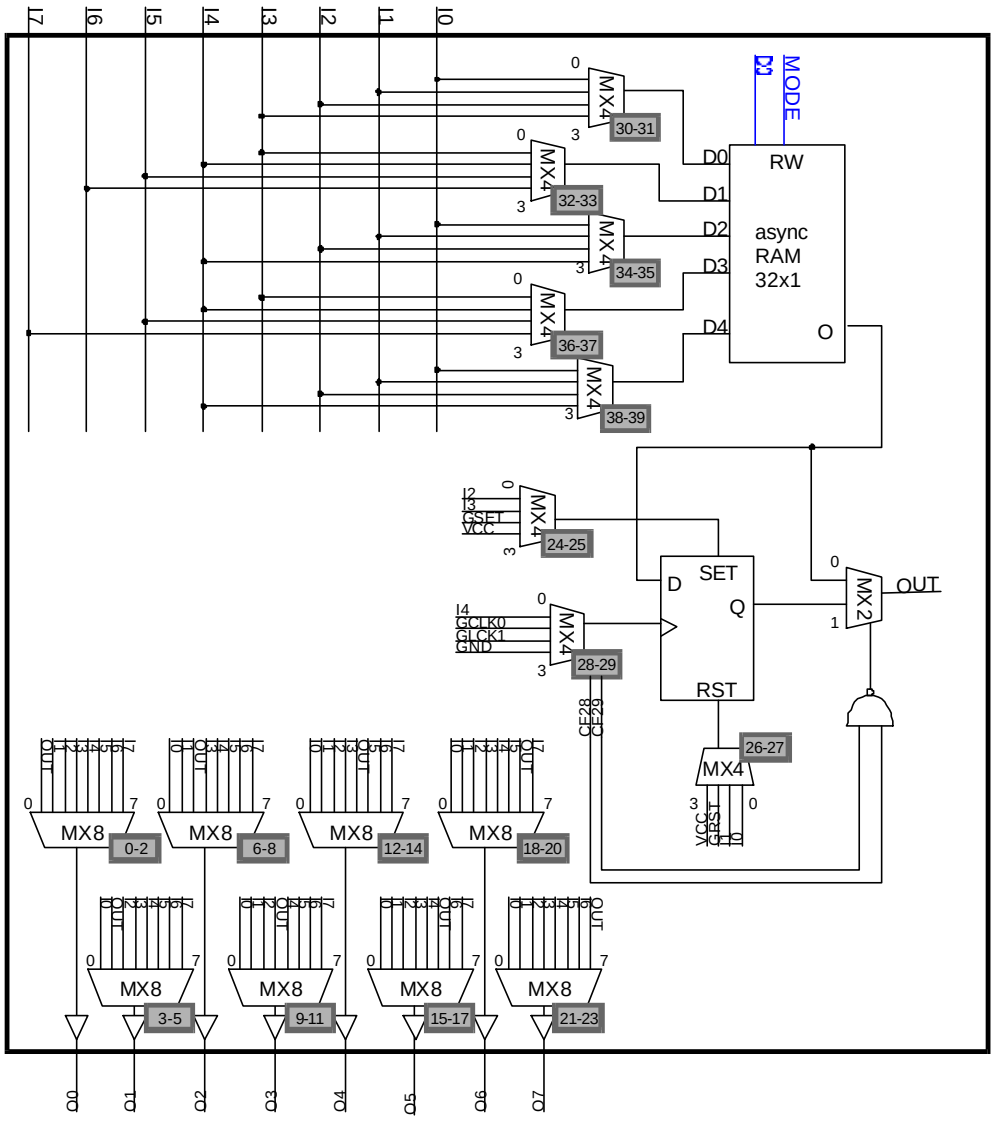


Table 1: GPC Configuration Registers

[4] All selections can be represented by only 6bits, since C(5, 8) = 48, but circuit size would be much larger than it is now. This selection scheme is not only one possible - but this has minimal maximum fan-out per input - 3. Furthermore there exists no such circuit, that would allow such selection with four MUX4 and one MUX2.

Name	Description
CF0-CF23	Cell bypass select
CF24-CF25	Asynchronous SET select
CF26-CF27	Asynchronous RESET select
CF28-CF29	Clock select
CF30-CF39	Input select
CF40-CF71	LUT data

Table 2: GPC Signals (external/programming/internal)

Name	Width	Direction	Description
I	8	I	Eight inputs from neighbours, see Figure 4 for indexes
O	8	O	Eight outputs to neighbours, see Figure 4 for indexes
GCLK	2	I	Two global clocks (H-trees)
GRST	1	I	Global asynchronous reset signal
GSET	1	I	Global asynchronous set signal
RW	1	I	Programming circuit uses this signal to program SRAM
D	1	I	Programming circuit uses this signal to program SRAM
RST	1	I	Asynchronous flip-flop reset
SET	1	I	Asynchronous flip-flop set
OUT	1	I	This cell result

2.2.2 Input/Output Cell - IOC

Each IOC allows connection between I/O logic (port registers and possibly external package pins). IOC holds one register for input and one register for output. Various synchronization signals can be used (PIO_CLK and both global clocks), allowing several modes of operation:

- clock rise/fall
- asynchronous

IOC functions much like GPC - it also has same routing capabilities - it allows pass trough of any of eight inputs to any of eight outputs. But since IOCs are on perimeter, up to five inputs can be selected (others are set to zero). Note that IOCs neighbours can also be IOCs.
Configuration memory is linked into scan-chain. CF0 is connected to previous cell's CF32 and CF32 is connected to next cell's CF0.

Table 3: IOC Signals (external/internal)

Name	Width	Direction	Description
I	8	I	Eight inputs from neighbours, see Figure 4 for indexes
O	8	O	Eight outputs to neighbours, see Figure 4 for indexes
GCLK	2	I	Two global clocks (H-trees)
PORT_I	1	I	Port input
PORT_O	1	O	Port output
RESET	1	I	FPGA core reset signal
OUT	1	I	This cell result

Figure 6: Input/Output Cell Schematic

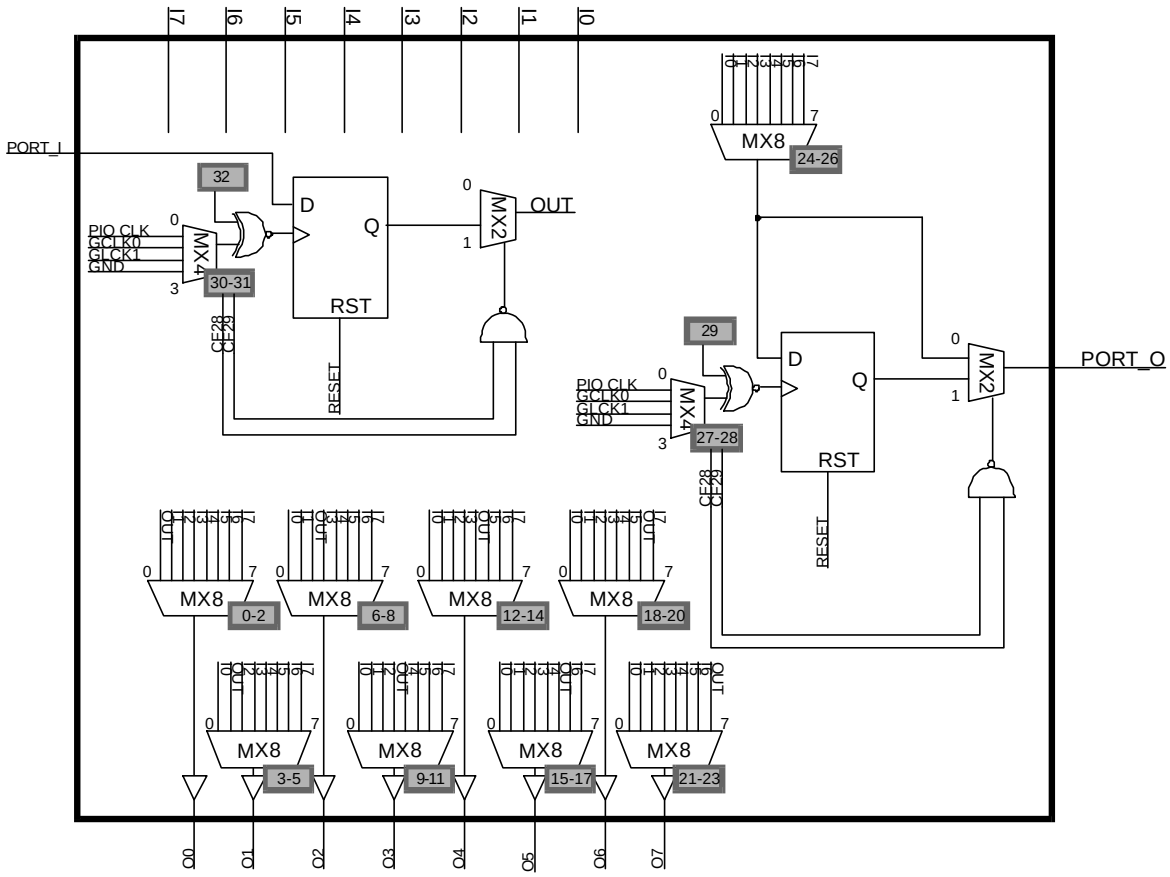


Table 4: IOC Configuration Registers

Name	Description
CF0-CF23	Cell bypass select
CF24-CF26	Input select
CF27-CF29	PORT_O clock select
CF30-CF32	PORT_I clock select

Up to 32 adjacent IOCs (in same column or row) form 32-bit port register (bi-directional). These registers can be accessed by I/O logic module. For each side, registers are allocated from left-to-right or top-down, where every IOC holds one bit. Sides are ordered: top, bottom, left, right. In this implementation upper 8 bits of port register are unused and are set to zero.

2.2.3 Special Resources

Special resources are (abstract) cells, which allow extra functionality required in common circuits. These include two global clocks (buffered using H trees) and global asynchronous reset and set, to which every flip-flop can attach.
Special resources does not have routing capabilities, or any other configurable elements, since they are only entry/exit points for special signals not available elsewhere. Both IOCs and GPCs can connect to special resources.

Table 5: Special Resource Cells

Name	Access (relative to RM)	Description
SR0	I	External clock
SR1	O	GCLK0 entry
SR2	I	WISHBONE clock
SR3	O	GCLK1 entry
SR4	O	Asynchronous SET select
SR5	O	Asynchronous RESET select

2.3 I/O Logic Module

This module connects 5 port registers with SoC bus or external package pins. 4 port registers are and one 32-bit register can be connected to chip pins. Only four registers can be accessed directly during runtime by using register mapping, others can be accessed via register addressing. Registers are explained further in chapter 4.

2.4 Programming Logic Module

Next module is responsible for programming the array. It is connected to FPGA and SoC bus interface. Every time value is sent to PDATA register, data in selected row/column is shifted trough the scan-chain and new values are inserted. If an error occurred while programming PERR bit in COR register is set.

Programming is divided in eight phases:

1. reset, initializes flip-flops, counters initialization
2. set default flip-flop values
3. clearing unused FPGA, to prevent unnecessary power consumption
4. GPC programming
5. IOC programming
6. port registers allocation and setup
7. programming external pins (see Pin Multiplexing Module specification)
8. connection to external pins

Note that FPGA can be fully functional while reprogramming[5], not affecting performance of running circuit.

2.5 In Circuit Debugging

This architecture implementation does not yet support such functionality. User should supply extra programming circuitry for debugging.

[5] Circuits must not overlap.

(This page is intentionally left blank)

3

IO Ports

This section lists all IOs of the FPGA Programming Logic.
All ports are active HIGH unless otherwise noted.

3.1 WISHBONE Interface

Table 6: WISHBONE interface (not used are not listed)

Name	Width	Access	Description
CSR_I	1	I	Core Select Register
ADDR_I	32	I	Address Input
DATA_I	32	I	Data Input
DATA_O	32	O	Data Output
SEL_I	4	I	Indicates which bytes are valid on the data bus. Whenever this signal is not 1111b during a 32b access, the ERR is asserted.
ACK_O	1	O	Acknowledges normal cycle termination.
ERR_O	1	O	Error acknowledgment output. Indicates an abnormal cycle termination.
RTY_O	1	O	Retry - whenever interface is not ready this signal is asserted. Master should retry.
WE_I	1	I	Write Enable - indicates write cycle, when high.
STB_I	1	I	Indicates the beginning of valid transfer cycle.
CLK_I	1	I	Clock input (not necessarily same clock for programmable array).
RST_I	1	I	Reset input.
INTA_O	1	O	Interrupt Output A
INTB_O	1	O	Interrupt Output B

3.2 Pin Multiplexing Module Interface

Table 7: Pin Multiplexing Module Interface

Name	Width	Access	Description
PIN_I	32	I	Pin Data Input. Only part of register can be valid.
PIN_O	32	O	Pin Data Output. Only part of register can be valid.
PINV_I	1	I	PIN_I data valid.
PINV_O	1	O	PIN_O data valid.

4

Registers

This section describes all control and port registers inside the FPGA core. Port registers are the main communication tool between FPGA and other cores. Each FPGA core at programming time allocates one or more port registers (this implementation has only four of them). Four of them can be mapped directly to MPR*x*. This mapping is set in MPRA register. Other registers, which cannot be accessed directly, can be read from/written to using PORTA and PORTD registers.
Package pin configuration is set in pin multiplexing module.

Figure 7: Port registers and register mapping

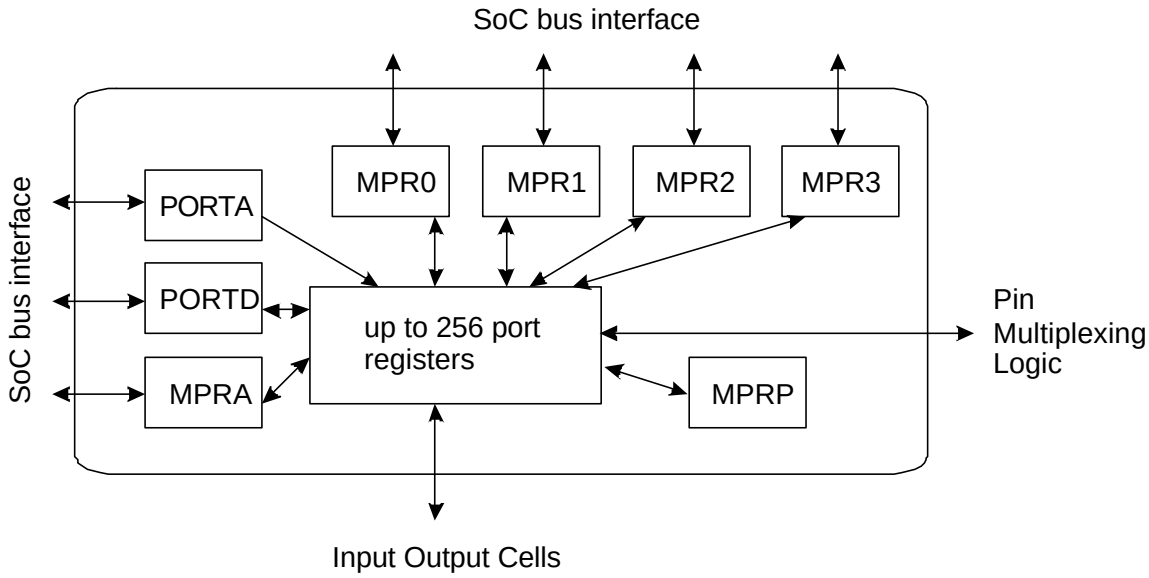


Table 8: Port/Control/Data Registers

Name	Address	Width	Access	Description
COR	00	32	RW	Configuration Register
PDATA	01	32	WO	Data port (for programming only)
PORTA	02	8	RW	Port register address

PORTD	03	32	RW	Port register data
MPRA	04	32	RW	Mapped port register addresses
PPR	05	32	RW	Pin port register
	06			RESERVED
ICR	07	32	RW	Interrupt Configuration Register
MPR0	08	32	RW	Mapped port register 0
MPR1	09	32	RW	Mapped port register 1
MPR2	0A	32	RW	Mapped port register 2
MPR3	0B	32	RW	Mapped port register 3

4.1 Configuration Register (COR)

This is main configuration register for FPGA core.

Table 9: Configuration Register

Bit #	Access	Description
31:9		RESERVED
8	RW	PERR bit indicates, that error has occurred while programming, it should be set to 0 to allow further programming
7	RW	INTB enabled when set to 1
6	RW	INTA enabled when set to 1
5	RW	Pins access allowed if set to 1.
4	RW	SoC bus access allowed if set to 1.
3	RW	Port registers access rights 0: read only 1: read and write
2:1	RW	Set programming mode 0: block 1-3 RESERVED
0	RW	Indicates if FPGA is enabled. Resets programming logic (and flip-flops, using GRST) and programmable array, if set to 0. If set to 1, all functions set by current configuration are enabled.

Reset Value:

COR: 00000000h

4.2 Programming Data Register (PDATA)

Data used for programming is sent to this address. Programming logic decodes it and writes it into next destination, IOC or GPC box. If an error occurs, all data sent to PDATA is ignored, until PERR bit is manually set to zero. Bitstream is covered in detail on page 22.

Reset Value:

PDATA: 00000000h

4.3 Port Registers

There are up to 256 port registers, which can be accessed via PORTD. Register address is specified in PORTA.

Reset Value:

PORTA: 00h
PORTD: 00000000h

4.4 Mapped Port Register (MPRx)

Four directly mapped port registers: MPR0, MPR1, MPR2 and MPR3.

Table 10: Mapped Port Register Address - MPRA

Bit #	Access	Description
31:24	RW	MPR3 address
23:16	RW	MPR2 address
15:8	RW	MPR1 address
7:0	RW	MPR0 address

Reset Value:

MSRA: 00000000h

4.5 Pin Port Register (PPR)

One register can be linked with external pins, providing direct connection to 32 IOCs.

Table 11: Pin Port Register - PPR

Bit #	Access	Description
7:0	RW	Linked register address

Reset Value:

PSR: 00000000h

4.6 Interrupt Configuration Register (ICR)

This is interrupt configuration register for FPGA core. It specifies interrupt activation criteria for both INTA and INTB. Interrupts can occur when specified port register bit change.

Table 12: Interrupt Configuration Register

Bit #	Access	Description
31:24	RW	Selects register for INTB.
23:19	RW	Indicates which bit of selected register will signal INTB.
18		RESERVED
17:16	RW	Interrupt mode for selected signal, INTB 00 - on change 01 - signal rise 10 - signal fall 11 - unused
15:8	RW	Selects register for INTA.
7:3	RW	Indicates which bit of selected register will signal INTA.
2		RESERVED
1:0	RW	Interrupt mode for selected signal, INTA 00 - on change 01 - signal rise 10 - signal fall 11 - unused

Reset Value:

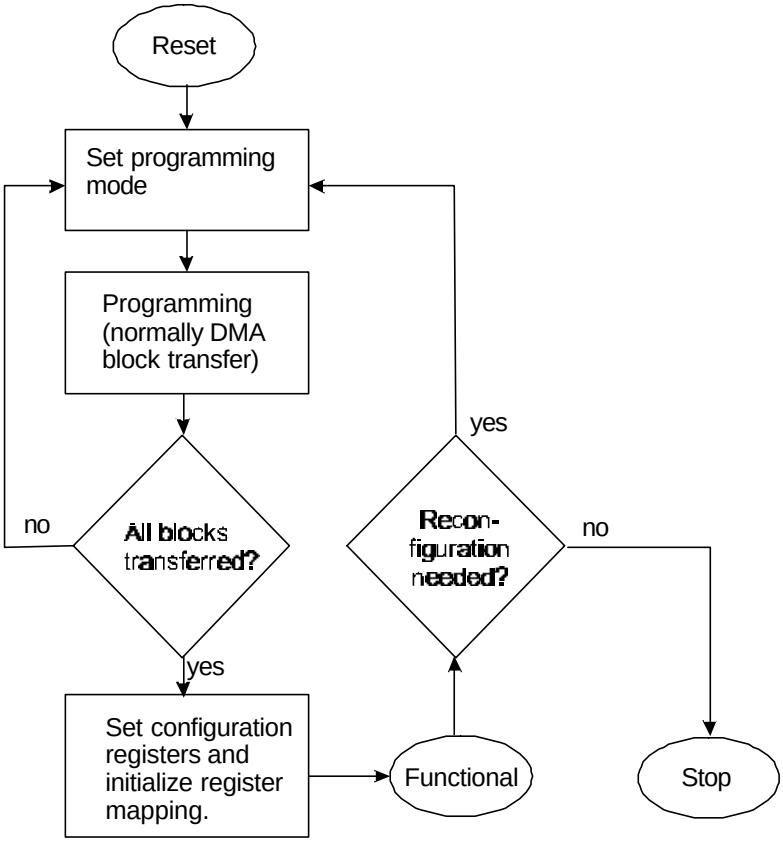
ICR: 00000000h

5 Operation

5.1 Normal FPGA Operation

Before FPGA can perform desired functionality, it has to be programmed. Bitstream data required for programming is acquired from P&R software. Programming mode must be set to block transfer, and then unmodified bitstream block is transferred (e.g. using DMA transfer). If FPGA is to have multiple designs, we have to send bitstream of each one. Configuration registers are set via WISHBONE interface. Register mapping is established and software has to modify its SoC addresses accordingly. Pin multiplexing core is initialized, if needed. Then COR[0] is set to 1, enabling FPGA core. At this point FPGA is fully functional. If there is any need of runtime reconfiguration, new bitstreams can be sent in later. Normally procedure is very similar to first time programming, except bit 0 of COR register should not be changed. Note that we have to carefully reconfigure resources, not disturbing current operations.

Figure 8: Normal FPGA operation



(This page is intentionally left blank)

6

Appendix A: Programming Bitstream Specification

6.1 Bit and Byte Ordering

Bitstream representation uses MSB ordering (big endian, as it is sometimes called). Basic (and minimal) data unit for bitstream is one word (32b).

6.2 GPC column

For GPC configuration 72b are needed so we need 1728b for column with 24 rows. Most significant bit comes first and represents first GPC configuration bit CF0. Last bit in stream represents CF71 bit of the lowest GPC.

Table 13: Bitstream ordering

Address [words]	Length [words]	Description	Value
0000h	1	Header	00C00000h+column
0001h	1	Data length [words]	54
0002h-0038h	54	Bitstream column data	

6.3 IOC row/column

IOC requires 33b for configuration, so we need 792b (25 words) for row/column with 24 rows/columns. Most significant bit comes first and represents first IOC configuration bit CF0. Last bit in stream represents CF32 bit of last IOC. Least significant eight bits are ignored.

Table 14: Bitstream ordering

Address [words]	Length [words]	Description	Value
0000h	1	Header	00C10000h + row/column
0001h	1	Data length [words]	25
0002h-001Bh	25	Bitstream row/column data	

6.4 Programming Entire FPGA

In order to program entire FPGA bitstream of length 34656 words (around 136KB) should be sent to programming module. Data is split into packets, each packet representing one row/column, as shown in previous section. There is no header or any extra data before or after the programming. See chapter 5 for FPGA operation.