Architecture of M32632 V2

Content

Introduction 3
Top Level 4
Global Control 5
Data Path 9
Caches 13
Timing 18
Ressources 22
Final Remark 23

Introduction

The M32632 is an implementation of the Series 32000 architecture of National Semiconductor. It is 100% software compatible to the NS32532 CPU and NS32381 FPU. The implementation is written in Verilog.

The M32632 is a very powerful 32-bit microprocessor. It features a rich instruction set with datatypes ranging fom bits to strings, a memory management unit which supports demand-paging virtual memory and a floating point unit for single and double precision operands supporting basic arithmetic operations.

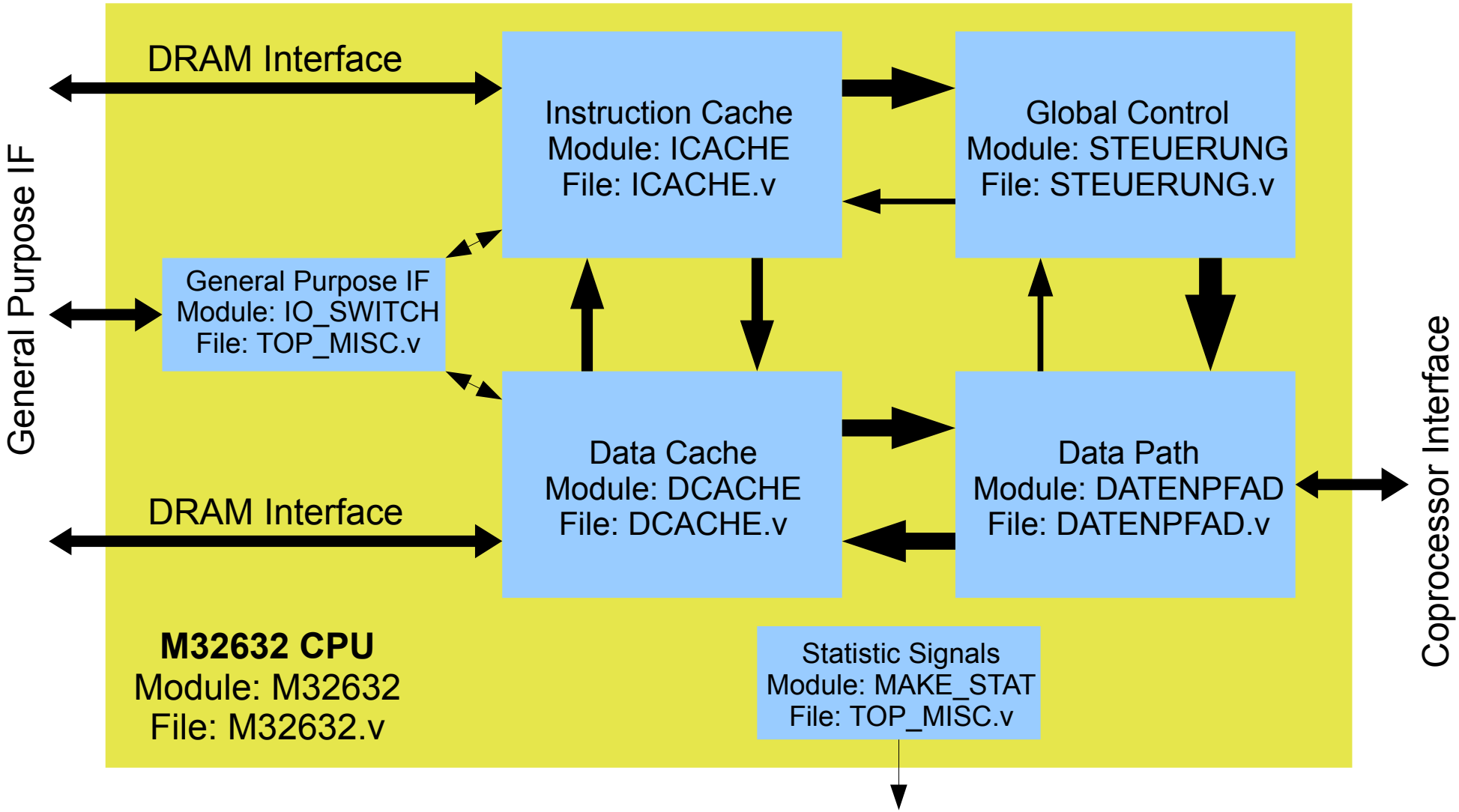
This manual gives an overview of the internal structure and the principal operation of M32632.

The target application for this processor was a software MP3 decoder. The original NS32532 CPU together with the NS32381 FPU at 25 MHz clock speed was 10 times too slow to decode MP3 in real-time. The M32632 V2 achieves 50 MHz in a cheap Cyclone IV -6 FPGA. This clock speed is sufficient to decode MP3 in real-time.

The figure on the next page shows the top level of M32632.

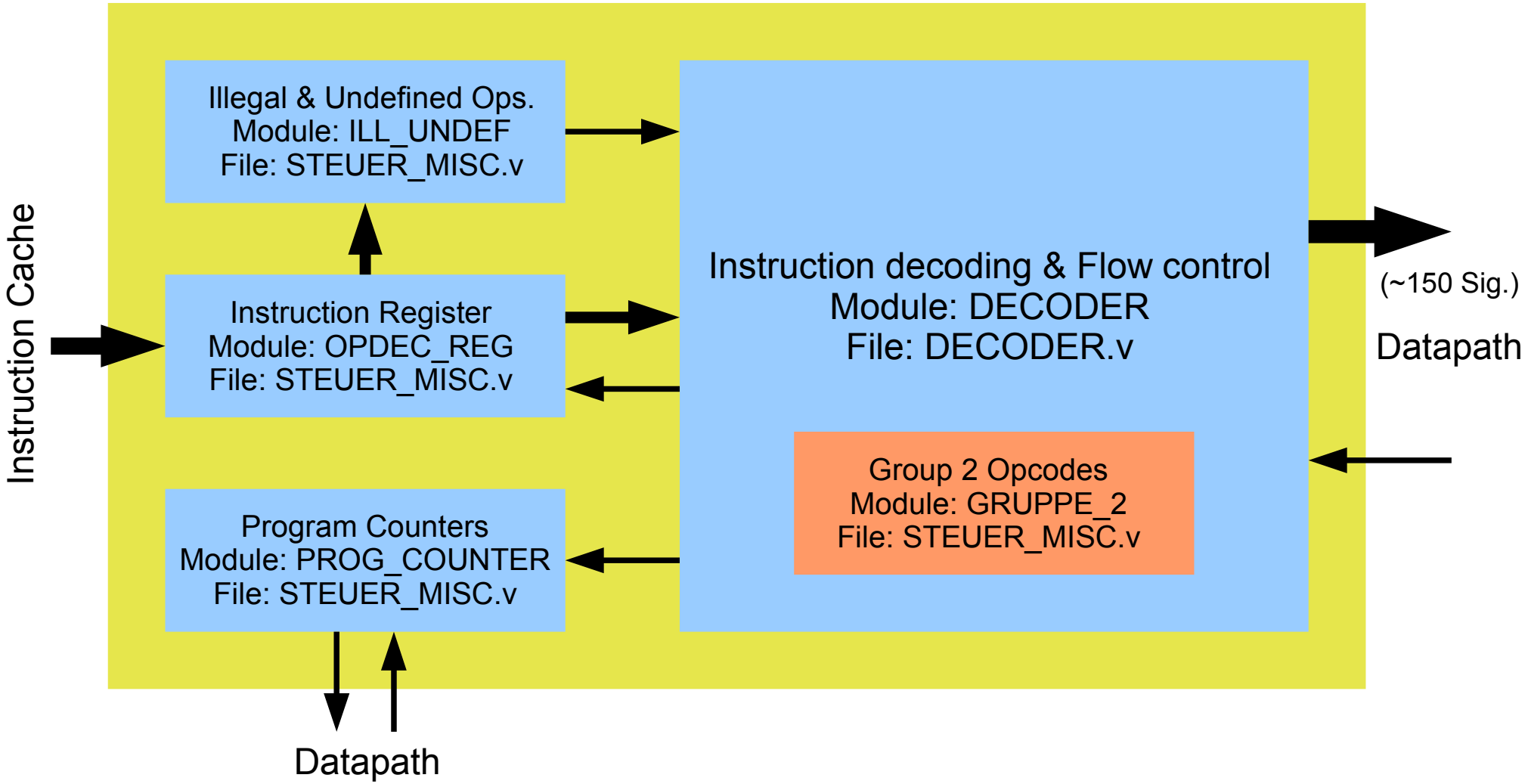
Top Level

The figure below shows the structure of the top level. The CPU logic is divided into 4 big and 2 small modules.



Global Control

The figure below shows the structure of the global control.
It contains the biggest module (in terms of design effort) of the design, the DECODER. 3 small modules add support functions.

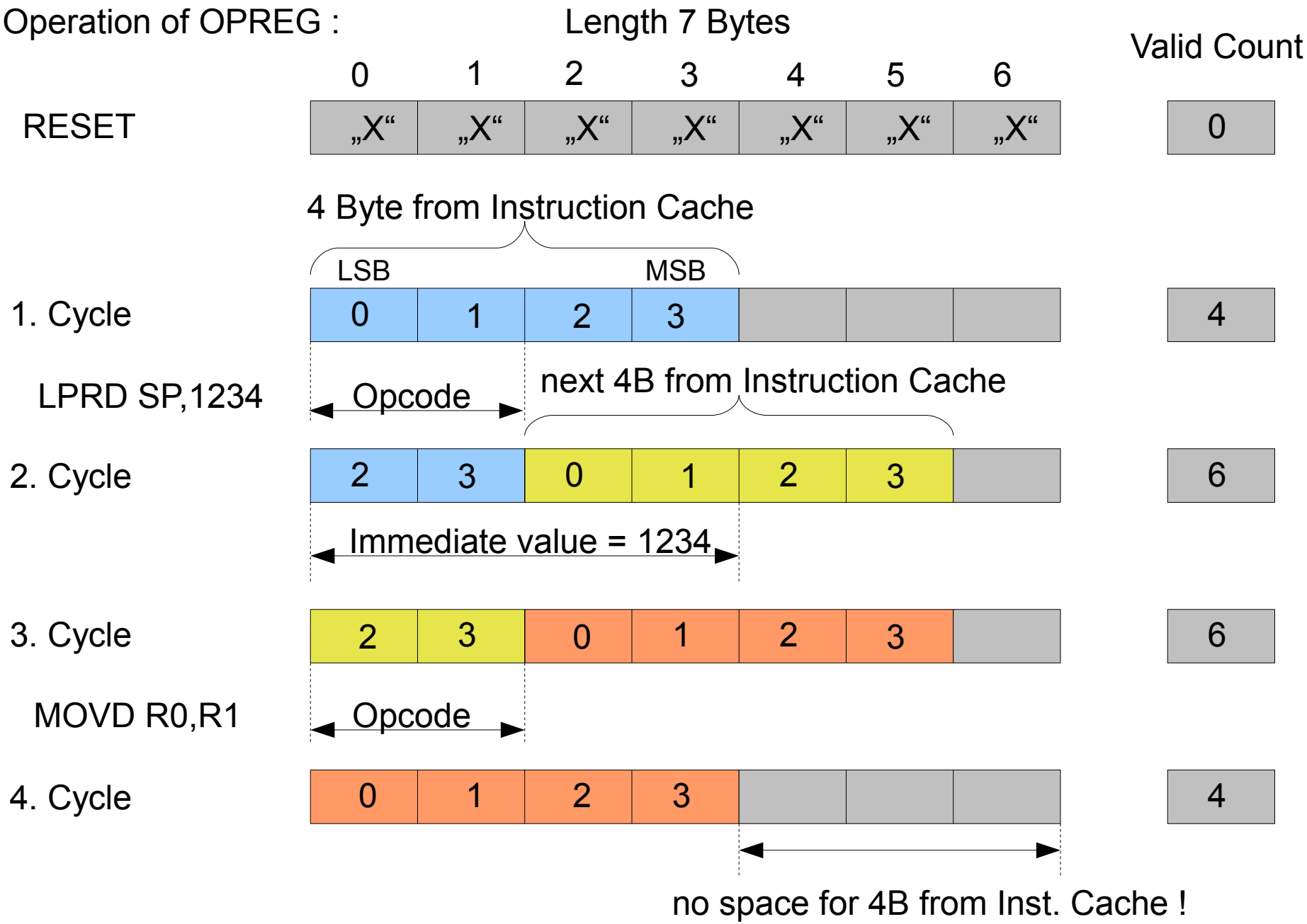


The three most important elements of STEUERUNG are the architecture program counter PC_ARCHI inside the module PROG_COUNTER, the register OPREG in the module OPDEC_REG and the register phase_reg in the module DECODER.

PC_ARCHI is the program counter (PC) which holds the address of the first byte of an instruction. In the Series 32000 architecture the PC is used for the addressing mode PC relative. For branches the PC is used to calculate the target address. There are two more program counters inside the module PROG_COUNTER. pc_adduse is advancing in small steps. A step is an opcode (1..3 bytes), an index operand (1..2 bytes), an immediate value (1..4 bytes) or a displacement (1..4 bytes). Therefore pc_adduse points to every part of an instruction. The third program counter is PC_ICACHE. It is used for accessing the instruction cache and advances in steps of 4 bytes until it is updated to a new location by a jump.

The register OPREG in the module OPDEC_REG holds the data from the instruction cache for decoding by the module DECODER. It is 7 bytes wide and can be updated every clock cycle. The data to decode is located starting at byte 0. The lenght depends on the kind of information to evaluate. For example instruction definitions can be 1, 2 or 3 bytes long. The instruction cache always fill the OPREG with 4 bytes until there is no space available. On the next page the operation of OPREG is visualized.

The register phase_reg in the module DECODER is the state register of M32632. It is 8 bits wide. Sequences of states are defined for all instructions. Similar instructions have the same sequence. For example the instructions for add (ADD) and subtract (SUB) differ only in the data operation. State 0 is the opcode decoding state of any instruction.

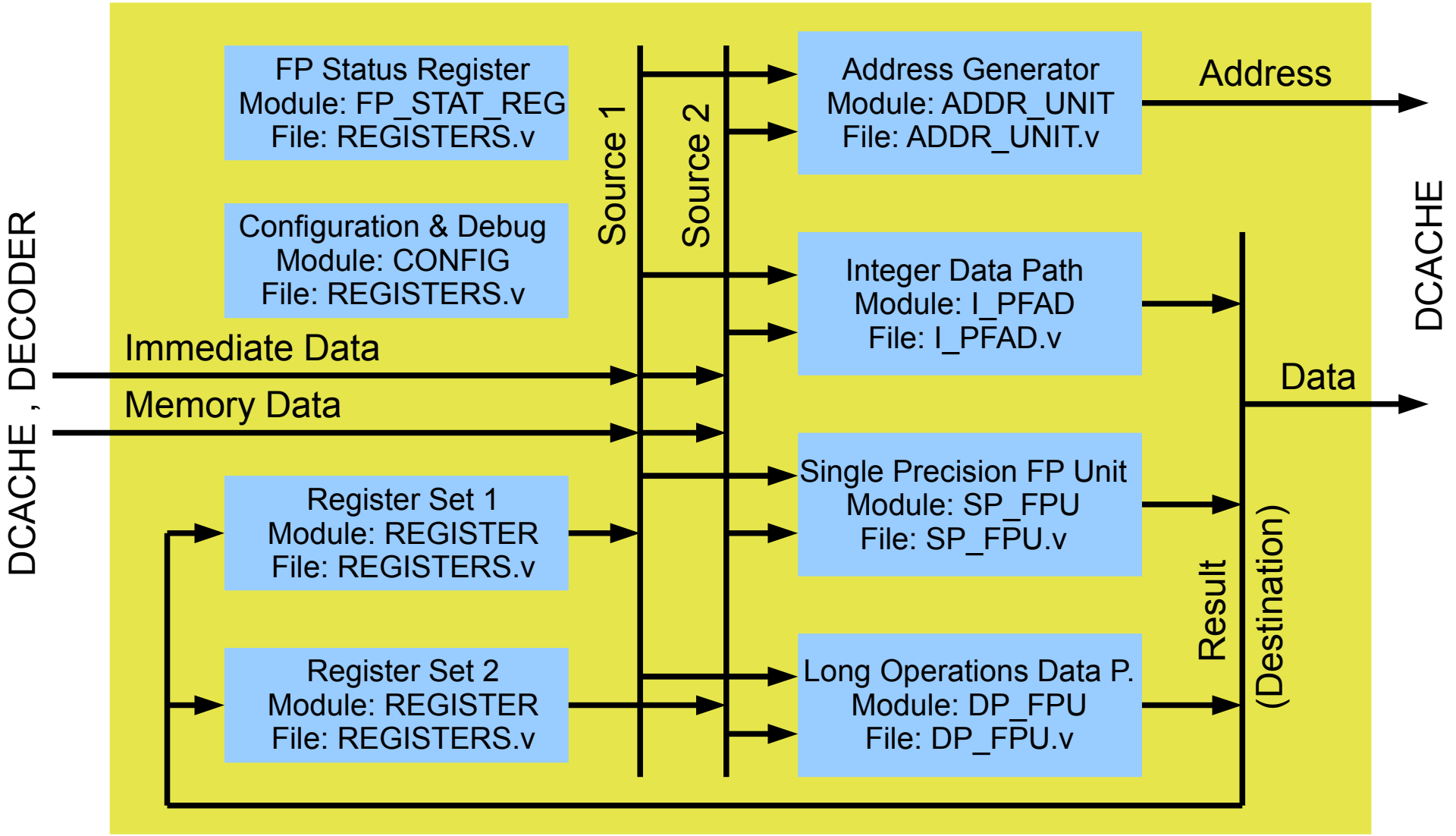

Operation of OPREG :
Length 7 Bytes
Valid Count
0 1 2 3 4 5 6
RESET
„X“ „X“ „X“ „X“ „X“ „X“ „X“
0
4 Byte from Instruction Cache
LSB
MSB
1. Cycle
0 1 2 3
4
LPRD SP,1234
Opcode
next 4B from Instruction Cache
2. Cycle
2 3 0 1 2 3
6
Immediate value = 1234
3. Cycle
2 3 0 1 2 3
6
MOVD R0,R1
Opcode
4. Cycle
0 1 2 3
4
no space for 4B from Inst. Cache !

Normally the phase_reg steps from one instruction to the next instruction. The only exception is a trap. Traps are special events in the Series 32000 architecture. They can happen at any time. For example the instructions for division (i.e. DIVi,DIVf etc.) can be trapped if the source 1 operand is zero. Other traps are interrupts or aborts of the memory management unit.

Most of the instructions have a regular work flow. They read one or two operands, do something and store one result. The instructions with an individual work flow are evaluated in the module GRUPPE_2.

Data Path

The figure below shows the data path structure. Only the data connections are indicated. Every arrow is a 32 bit wide bus.



The module DATENPFAD is the place where new data is generated. Different modules are used for different data types. The module I_PFAD generate integers, strings and bits. The module SP_FPU generates single precision results. The module DP_FPU is responsible for double precision results, binay coded decimals (BCD) results and all variants of division. In other words DP_FPU executes all operations which require multiple clock cycles to finish. Therefore the data path of the coprocessor interface is located also in this module.

For high speed operation it is necessary to get two operands in one clock cycle. It is also best to write one result in each clock cycle. Memory blocks of FPGAs support one read and one write simultanously. Therefore two memory blocks with the same content are working in parallel to get two reads and one write in one clock cycle (modules REGISTER). The result of an operation is written in each memory block.

The memory block of REGISTER is organized in 64 words of 32 bits. For 64-bit double precision operands the memory block must be read or written twice. The figure on the next page shows the mapping of the Series 32000 registers to the memory block.

The module ADDR_UNIT generates the addresses for operands in memory. If an operand is not stored on an aligned address, the module generates the sequence of addresses. The addresses are used in the data cache. The response from the data cache can either be access ok (signal ACC_STAT[0]) or access error (signals ACC_STAT[1] or ACC_STAT[3]).

Register File

Empty fields can be used as temporary registers.

		+0	+1	+2	+3
CPU	x'0	R0	R1	R2	R3
	x'4	R4	R5	R6	R7
MMU	x'8		MCR	MSR	TEAR
	x'C	PTB0	PTB1	IVAR0	IVAR1
CPU	x'10	UPSR	DCR	BPC	DSR
	x'14	CAR			FSR
Exception x'17 = FSR	x'18	FP	SP[0]	SB	USP[0]
	x'1C	CFG	PSR	INTBASE	MOD
FPU	x'20	F0:L / F0:F & F1:F		F1:L	
	x'24	F2:L / F2:F & F3:F		F3:L	
	x'28	F4:L / F4:F & F5:F		F5:L	
	x'2C	F6:L / F6:F & F7:F		F7:L	
Used by opcode CINV	x'30				
	x'34				
The Stackpointer exist twice ! →	x'38		SP[1]		USP[1]
	x'3C	TEMP_L	TEMP_H	TEMP_1	TEMP_2

Access errors can only happen if virtual addresses are used. In case of an error the ADDR_UNIT stops the access and informs the DECODER that an abort had occured.

The Series 32000 architecture implements two stack pointers (SP). One is for user programs, the other is for programs in supervisor mode. The M32632 doubles each stack pointer. This is done because every instruction is restartable in case of an MMU abort. For exampe the instruction MOVD R0,TOS moves the content of register R0 to the stack. The stack pointer is decremented before the write operation. In case of an abort during the write operation the modified stack pointer has to be restored. This is the purpose of the second register for the same stack pointer which always store the old content of the SP.

The signal OPCODE inside DATENPFAD shows which operation is executed. The signal is 8 bits wide. It is build from the instruction table of Series 32000 architecture. For example the instruction MOVi (i = integer) is a format 4 instruction with a subcode of 5. The result for OPCODE is 8'h45.

Other important signals are BWD[1:0] = size of integers and FL = size of FP operands. Together with OPCODE they are transferred from the DECODER as the signal OPER.

Caches

Caches are essential to achieve high speed operation. Separate caches for instructions and data speeds up even further.

The M32632 implements instruction and data caches with a size of 8 Kbytes each. They are 2 way-associativ, physically indexed and have a line size of 16 bytes. Each cache delivers 4 bytes each clock cycle.

Both caches are not identical. The main difference is that the data cache can be written from the data path. The write policy of the data cache is „write through“. If a write occurs the external memory is updated instantly. Another important difference is that the instruction cache contains a second set of tag memories. The second set is used for detection of address collisions between the instruction cache and data writes. If a write occurs it is possible that the address written has been used for storing instructions in the past. Therefore the write invalidates the old content and the collision indicates to the instruction cache that the entry has to be marked invalid.

Two sets are used in parallel for fetching of instructions and detection of collisions. This is necessary because a write operation can happen at any time.

On the next two pages the components of the caches are listed.

Data Cache

Data Cache Memories

Type :	Words * Bits
Data Set 0 :	1024 * 32
Data Set 1 :	1024 * 32
Tag Set Data 0 :	256 * 16
Tag Set Data 1 :	256 * 16
Data Valid :	32 * 24
MMU Tag Set :	256 * 36
MMU Valid :	16 * 32

Data Cache Logic

Modules:
- DCACHE_SM
- DCA_CONTROL
- CA_MATCH
- MMU_MATCH
- MMU_UP
- DEBUG_AE
- FILTCMP

File: CACHE_LOGIC.v

Modules:
- RD_ALIGNER
- WR_ALIGNER

File: ALIGNER.v

Instruction Cache

Instruction Cache Memories

Type :	Words * Bits
Data Set 0 :	1024 * 32
Data Set 1 :	1024 * 32
Tag Set Data 0 :	256 * 16
Tag Set Data 1 :	256 * 16
Data Valid :	32 * 24
MMU Tag Set :	256 * 36
MMU Valid :	16 * 32
Collision Tag Set Data 0 :	256 * 16
Collision Tag Set Data 1 :	256 * 16
Collision Data Valid :	32 * 24

Instruction Cache Logic

Modules:
- DCA_CONTROL
- CA_MATCH
- MMU_MATCH
- MMU_UP
- FILTCMP

File: CACHE_LOGIC.v

Modules:
- ICACHE_SM
- KOLDETECT

File: ICACHE_SM.v

Both caches can only be feeded from the DRAM interface. Each cache reads a stream of 16 bytes (the line size) from the DRAM. The speed between the M32632 and the DRAM can be different and the caches serve as the bridge. The general purpose interface is limited to M32632 clock speed and can therefore not be used for bridgeing.

Both caches contain the memory and the logic for address translation to support virtual memory. The translation schema used in the M32632 is paging (another way is segmentation). The page size is 4096 bytes. The memory used in the cache for address translation is called „translation look-aside buffer“ (TLB). It has 256 entries. If a miss occurs during address translation the page table stored in DRAM is accessed by an operation called „page table walk“. This is done in hardware.

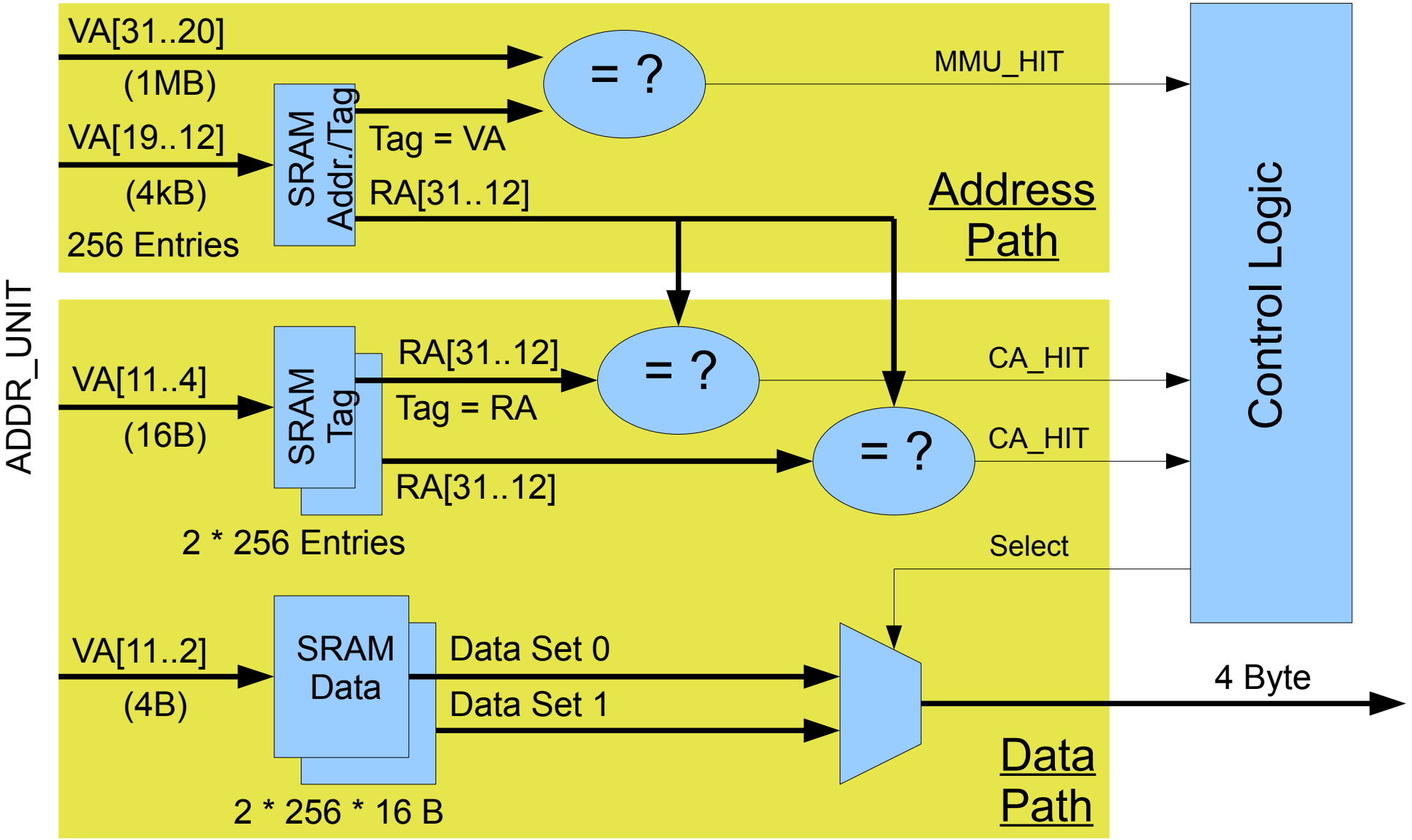
Only the data cache can do a page table walk. If the instruction cache needs a page table walk it signals a request to the data cache (signal IC_PREQ). The reason behind this strategy is that both caches work independent in general. But in virtual memory mode both caches update the page table stored in DRAM. To avoid any misseting of the page table only one access at a time is allowed.

The frequency of page table accesses is low compared to any other operation. The statistic port has two signals, one for instruction and one for data, to count these events.

The module CACHE_SM inside the data cache is responsible for the page table walk. Any page table access of CACHE_SM targets the DRAM interface.

The figure on the next page shows the address and data paths of both caches.

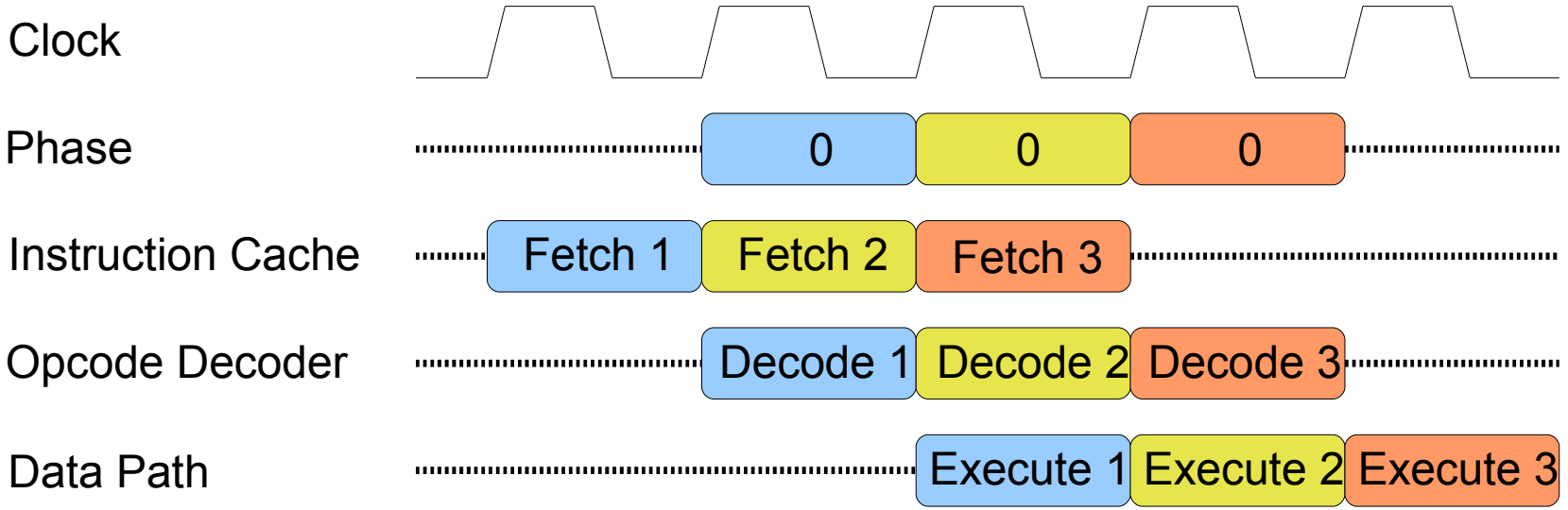
Use of addresses in cache : VA = virtual address , RA = real address



Timing

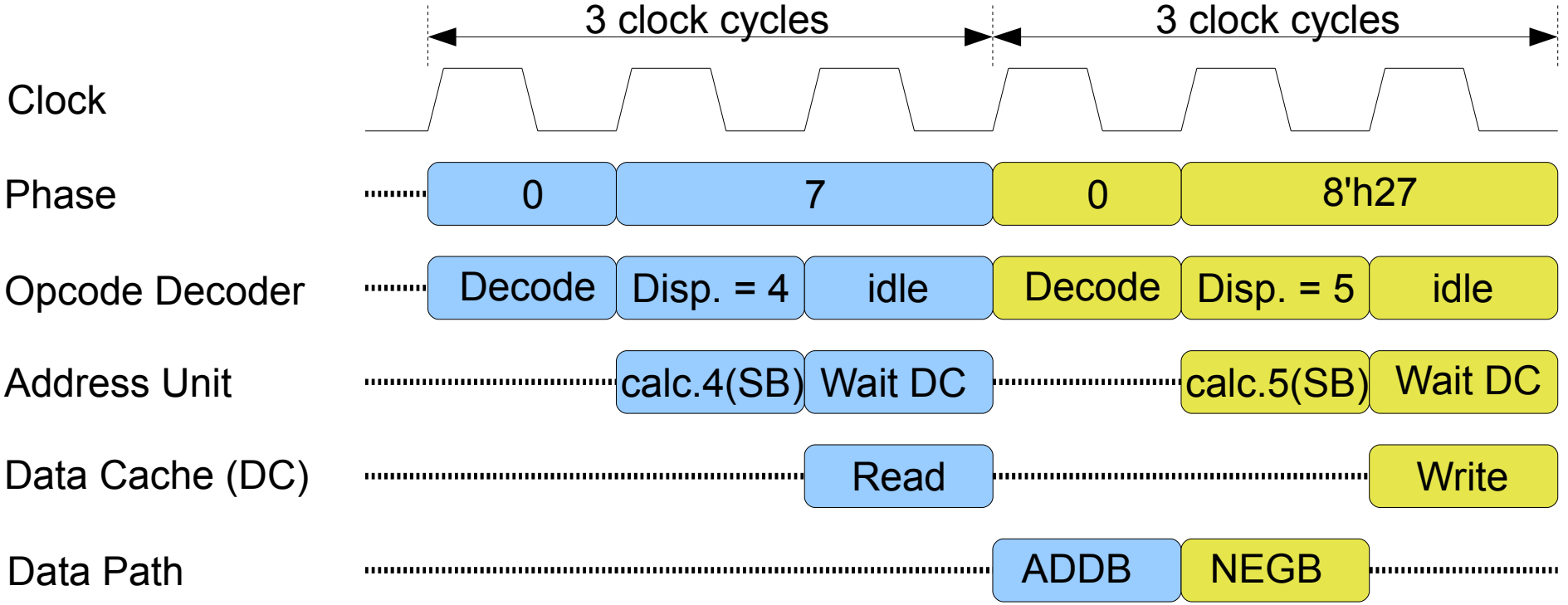
The M32632 has a three stage pipeline. The first stage is instruction fetching from the instruction cache. The second stage is instruction decoding in the module DECODER. Finally the instruction is executed in the module DATANPFAD.

All three stages work in parallel. The figure below shows a sequence of three instructions running through the pipeline. The throughput in this case is one instruction per clock cycle. But this sequence is only possible if the instructions have their operands in internal registers.

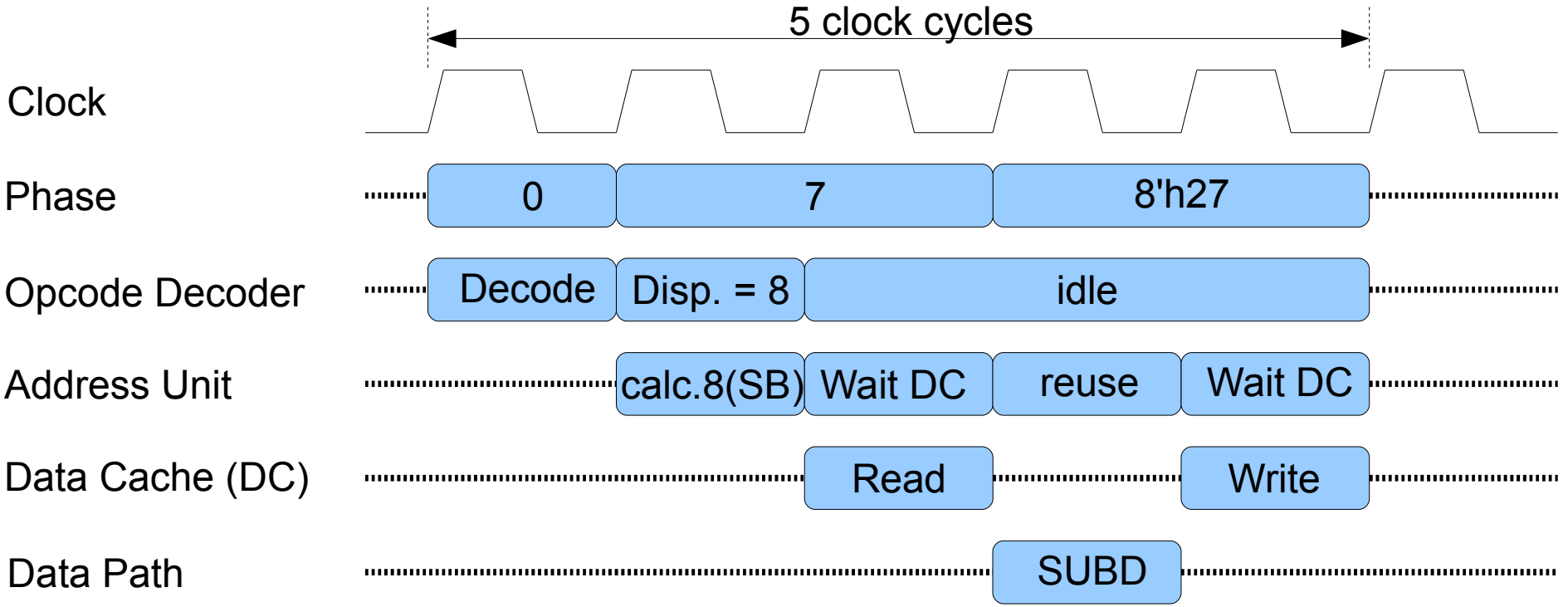


If one operand is in memory throughput goes down to one instruction per 3 clock cycles. But this is true only for a cache hit.

The figure below shows the internal operations if the data cache is read or written. The first instruction (blue) is for example an „ADDB 4(SB),R0“. It reads a byte from the address in the register „Static Base“ (SB) offset by 4 bytes and adds the byte to register 0. The next instruction (yellow) is for example „NEGB R0,5(SB)“. It reads the byte from register 0, negates it and stores the result at address 5(SB).



If the operand in memory is of type read-modify-write (rmw) the instruction will need a minumum of 5 clock cycles. The figure below shows the internal operations for this case. For example the instruction „SUBD R1,8(SB)“ subtracts the double-word content of register 1 from the memory at address 8(SB). The address unit simply reuses the calculated address from the read access for the write access.



If the data cache is accessed the module DECODER is not active every clock cycle (= idle). The reason for this not ideal behaviour is the virtual memory support. If the cache detects an abort the state of M32632 is reset to the beginning of the instruction. The task of reseting is simplified if the DECODER is waiting for the data cache finishing the current access.

If an operand of type „rmw“ is accessed the cache checks during the read access whether a write access is not allowed. If this is true the read access is already aborted. This early abort prevents changing the content of the „processor status register“ (PSR) by the operation.

The limiting timing path of the M32632 is the detection of an overflow during a single precision FP operation. An overflow occurs if a result is too big to be represented in the desired format. In this case the instruction is aborted and the result must not be written either to register nor to memory. For the register write the signal DOWR in the module DATENPFAD must be suppressed. The combinatorial path to achieve this is the longest path in the design.

In version 2.0 the single precision floating-point operations need 2 clock cylces. This allows a higher clock speed for the whole M32632.

Ressources

The table below shows the usage of ressources for the modules of M32632. The FPGA used is a Cyclone IV E from Altera.

Compilation Hierarchy Node	Logic Cells	Dedicated Logic Registers	I/O Registers	Memory Bits	M9Ks	DSP Elements
|M32632:CPU|	15530 (0)	2976 (0)	0 (0)	180992	31	26
|DATENPFAD:BAUCH|	9671 (153)	1926 (130)	0 (0)	4096	2	26
|ADDR_UNIT:ADDRU|	806 (806)	145 (145)	0 (0)	0	0	0
|BITMASK:BITROM|	84 (84)	0 (0)	0 (0)	0	0	0
|CONFIG_REGS:CONFIG|	129 (129)	102 (102)	0 (0)	0	0	0
|DP_FPU:DOUBLE|	4669 (134)	1230 (119)	0 (0)	0	0	18
|FP_STAT_REG:FPSREG|	40 (40)	22 (22)	0 (0)	0	0	0
|I_PFAD:GANZ|	2132 (1766)	96 (96)	0 (0)	0	0	0
|MULFILTER:MFILTER|	49 (49)	0 (0)	0 (0)	0	0	0
|REGISTER:REG_SET_A|	9 (9)	4 (4)	0 (0)	2048	1	0
|REGISTER:REG_SET_B|	8 (7)	4 (4)	0 (0)	2048	1	0
|SIGNMUL:SMULTI|	60 (0)	0 (0)	0 (0)	0	0	8
|SP_FPU:SINGLE|	1317 (192)	193 (2)	0 (0)	0	0	0
|busmux:DINMUX|	32 (0)	0 (0)	0 (0)	0	0	0
|busmux:MUX1H|	44 (0)	0 (0)	0 (0)	0	0	0
|busmux:MUX2H|	55 (0)	0 (0)	0 (0)	0	0	0
|busmux:OUTMUX|	215 (0)	0 (0)	0 (0)	0	0	0
|DCACHE:ARME|	1169 (245)	391 (225)	0 (0)	84224	13	0
|ICACHE:BEINE|	734 (121)	230 (119)	0 (0)	92672	16	0
|IO_SWITCH:ISWITCH|	35 (35)	3 (3)	0 (0)	0	0	0
|STEUERUNG:HIRN|	4082 (0)	418 (0)	0 (0)	0	0	0
|DECODER:BEFEHLS_DEC|	3473 (2494)	219 (207)	0 (0)	0	0	0
|GRUPPE_2:reste_ops|	962 (962)	12 (12)	0 (0)	0	0	0
|REG_LIST:scanner|	17 (17)	0 (0)	0 (0)	0	0	0
|ILL_UNDEF:CHECKER|	72 (72)	0 (0)	0 (0)	0	0	0
|OPDEC_REG:OPC_REG|	253 (253)	70 (70)	0 (0)	0	0	0
|PROG_COUNTER:PCS|	285 (285)	129 (129)	0 (0)	0	0	0
|make_stat:mkstat|	8 (8)	8 (8)	0 (0)	0	0	0

Final Remark

One of the last sentences in the „Final Remark“ of the first version of this document said

„*The version 1.0 of M32632 is to my knowledge free of errors.*“

This was pretty optimistic. Every new system showed new bugs. One bug became a feature. The NS32016 is able to multiply and divide an odd numbered register pair. This is not specified but it was used in the BBC 32016 Second Processor. I had to implement this functionality also in the M32632 to be able to run the BBC software.

Version 2.0 has all bugs fixed. In addition it delivers a higher clock speed. 50 MHz compared to 35 MHz is a big improvement. The drawback is a slower speed for single precision floating-point operations.

This time I am quiet sure (again) that there are no bugs.