1. OCP Config Register Shell

	3.0
Last editing date	**2008-10-29**

1.1. Summary

This document describes the functionality and architecture of a config register shell with an SCI OCP target interface.

1.1.1. Description

In the target project the aim is to generate the HDL code for a config register domain from a register-master or equivalent specification. The config register shell thus mainly consists of an OCP target (static component) and a register block which is generated via the MIX tool.

1.1.2. Purpose

The register shell can be applied wherever there is a single OCP access port to the registers from the system point of view (here: OCP *frontend*) and one or more processing units attached to the decoded registers (*backend*). No address information is forwarded to the backend, hence no further downstream bus can be attached.

1.2. List of Changes

	Date	Section/ Page	Changes compared to previous issue	Department	Name
1.1	2005-09-14	all	initial release	IPF	Thorsten Lutscher
1.2	2005-09-22	1.3; 1.5.3	added rd_err signal; some open items solved		
1.3	2005-09-26	some	added chapter “Detailed description”		
1.4	2005-10-06	1.5; 1.4;1.6	- some outputs were wrongly in the input Signal table - added PU response timeout feature - added addressing+timeout-error details - added single/multi-clock-domain configurations and corrected the text with respect to clock domains		
1.5	2005-10-17	1.5;1.6	- added sideband input port for soft-reset of OCP FSM - added description of shadowing		
1.6	2005-10-24	1.6	- added clock-gating etc.		
1.7	2005-11-07	1.4; 1.6	- small updates to graphics and text		
1.8	2005-12-16		- added reference to MIX documentation		
1.9	2006-02-08		- some clarifications for reads		
2.0	2006-04-07	1.4, 1.5, 1.6	added desription of new features		
2.1	2006-04-26	all	some corrections		
2.2	2006-05-05	1.6; 1.5	added description for ::cond feature of register-master; some clarifications for *_trg_p signal		
2.3	2006-03-07	1.4;	changed description about multi-clock-domain configuration		
2.4	2006-07-10	1.6	changed, added more details about error handling		
2.5	2006-09-27	1.4	documented changes in the mcda/pre-decoder module		
2.6	2006-09-28	1.4	small change to drawing		
2.7	2007-06-15	1.5, 1.6	clarifications for update signal handling and clock-gating; mention parameterization for presence of reset synchronizer		
2.8	2007-06-15	1.6	USR field output port name changed; clarifications for update signal timing; added W1C multi-bit feature; added exact timing for update in synchronous configuration		
2.9	2008-06-03				
3.0	2008-10-30		documented feature for 2nd type of handshake protocol		

Table 1–1: List of changes

1.3. Open Items

– add capability to generate interrupts by SW via the embedded control/status register

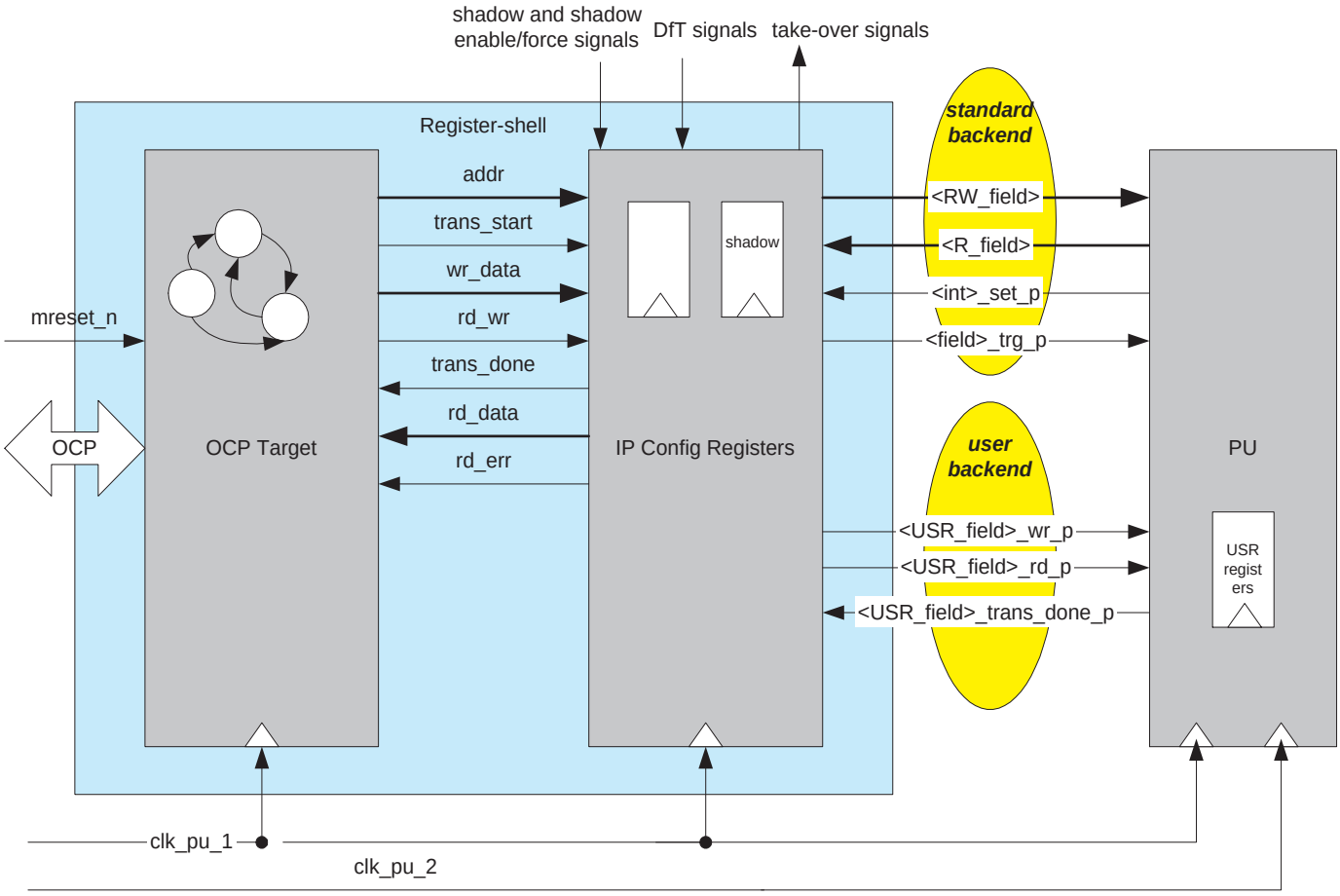
1.4. General Description and Block Diagram

The OCP config register shell consists at least of an SCI-profile OCP target interface and a script-generated config register block. There are two possible configurations.

1.4.1. Single-Clock Domain-Configuration

One configuration is called *single-clock-domain* (*SCD*), i.e. all config registers reside in one clock domain. This can also be the OCP bus clock domain (*SCD-sync*), or a different clock (*SCD-async*), but in the Monterey project the OCP bus clock is usually identical to the main PU clock and the clock-domain-crossing is done in the system interconnect. This results in a completely synchronous scenario (see fig. 1–1). An exception shall be allowed where the main PU clock cannot be controlled by the system itself but a lock-up of the OCP bus must be prevented. However, in any case the registers/fields are synchronous to the main PU clock and no further synchronization is necessary.

Figure 1–1: Overview of Blocks in SCD Configuration

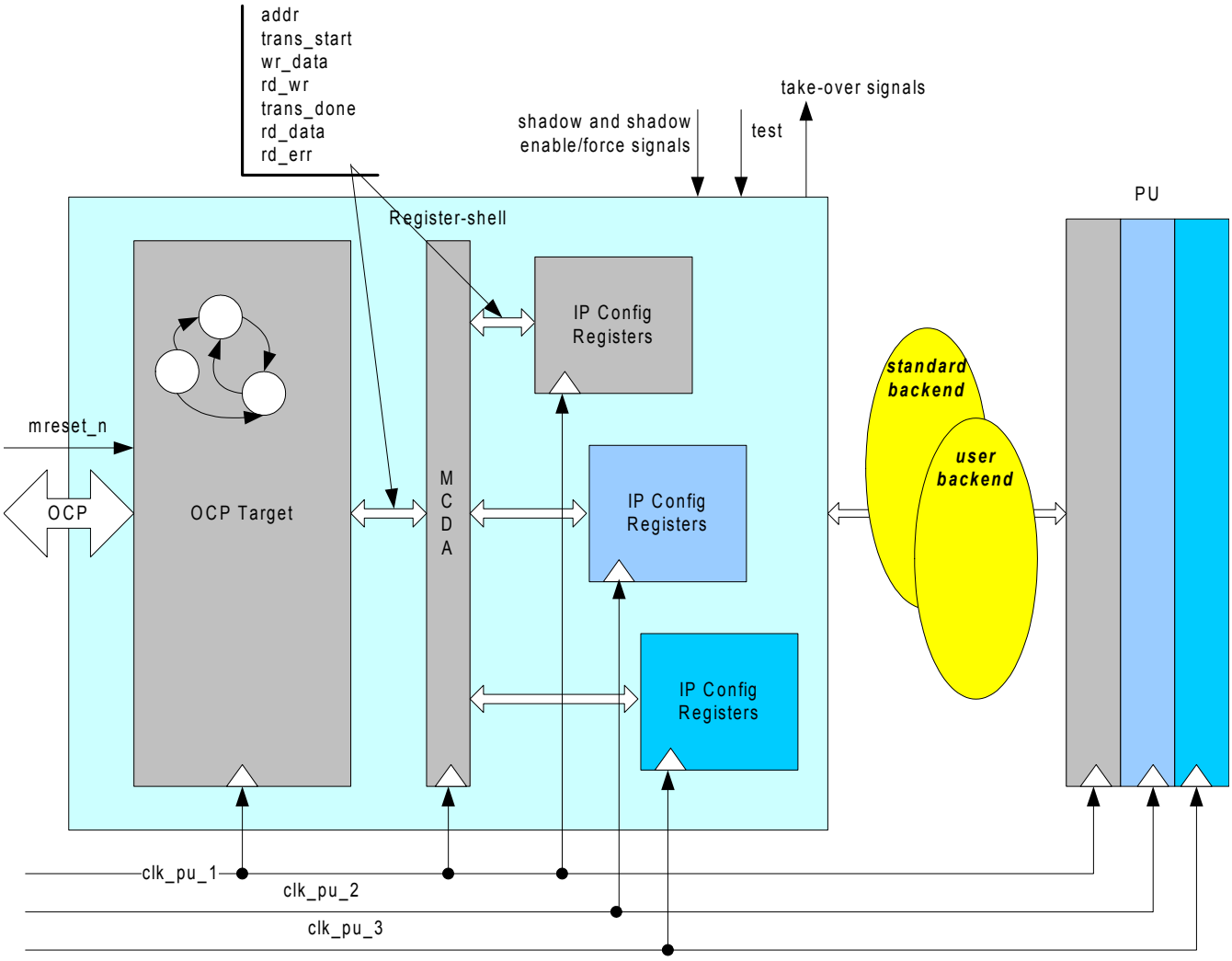


1.4.2. Multi-Clock-Domain Configuration

The other one is a *multi-clock-domain* configuration (*MCD*), where the config registers reside in different PU clock domains (see fig. 1–2). This requires another block called Multi-Clock-Domain Adapter which handles the CDC and

multiplexing of handshake and data signals. One or all of the config register blocks are asynchronous to the OCP bus clock.

Figure 1–2: Overview of Blocks in MCD Configuration



1.4.3. OCP Target Interface Block Features

– Generic design, can also be used for other purposes (configurable address/data width/OCP profile etc.)

– State machine handles OCP basic set of signals plus the simple extension *MRespAccept, MReset_n* and the *SInterrupt* signal (see [1] for required command subset). The profile is called “SCI” and is similar to the Register profile described in [2].

– SystemVerilog assertions for protocol checking

– Write and read acknowledge (*trans_done*) which supports both posted and delayed writes and low-latency reads

– Only one transaction at a time (no transaction buffering)

– Write data is buffered until the next OCP transaction is accepted

– Read data is buffered until the OCP master has accepted

– The handshake interface to the Register block uses toggle signals (this way it is easier to hook up a module that resides in a synchronous or asynchronous clock domain using the same handshaking protocol). Note: Micronas patent #C-2147.
It is also possible to configure the block to use a level-based protocol (is now supported by config register shell in *SCD-sync* configuration)

– Optional PU response timeout handling via an embedded control/status register, timeout value configurable via

software parameter

– No byte-enable support

– Optional OCP writeresp_enable and write-error feedback support (not supported by config register shell)

1.4.4. Config Register Block Features

– Script generated from single-source register description

– Registers/fields reside in PU clock-domain (default, but not mandatory)

– Handles address decoding

– Read data multiplexing (pipelined for large address spaces)

– Storage FFs for static registers

– Transaction forwarding for registers implemented in the PU

– Set and clear logic for status register bits

– Trigger signals to the PU to notify about register software access

– Shadow registers with take-over triggered by external event

– Lean CDC circuits, only handshaking, resets and shadowing signals are synchronized

– Clock-gating in RTL for dynamic power reduction

Fields that are implemented in the Register block or forwarded field data are connected via field inputs/outputs to the PU. Additionally, there are set-pulses from the PU for status bits (all fields that have the attribute *W1C* - write-one-to-clear) and - optionally - trigger pulses to indicate read or write access to a field. This is the *standard backend* in Fig. 1–1.

For registers that are implemented in the PU (Memory ports, mailbox registers etc. that have the attribute *USR*), dedicated read/write pulses are forwarded. This is the *user backend* in Fig. 1–1. The PU finishes the transactions by setting a dedicated transaction-done pulse.

1.4.5. Multi-clock-domain Adapter Features

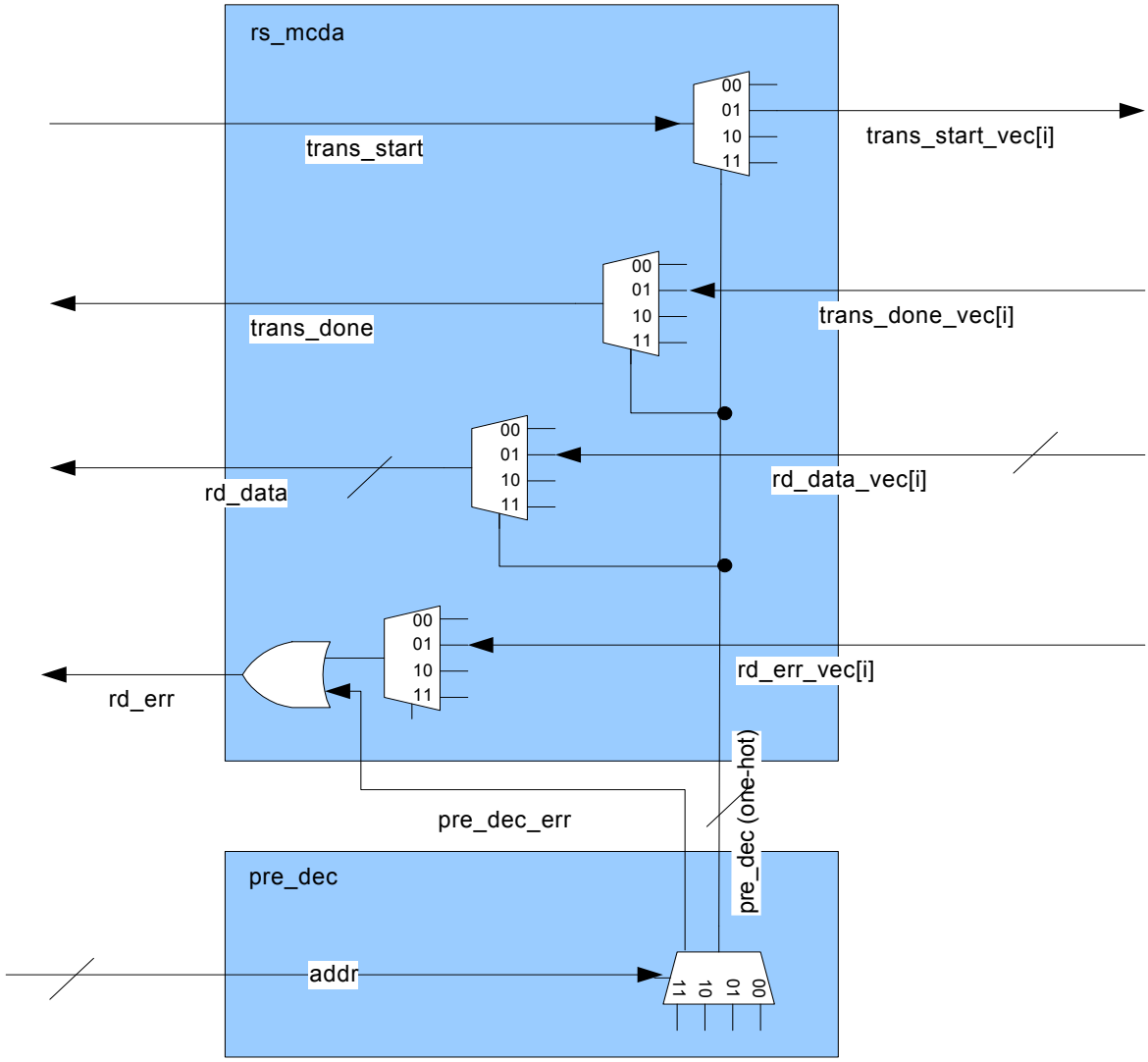
- Handles forwarding of transaction acknowledge, read data and read error signals from multiple clock domains to the OCP target block (multiplexing). This requires synchronizing all acknowledge signals into the OCP clock domain.

- Performs pre-decoding of register addresses to address only the one Config Register block that is the target of the transaction. In the implementation, the pre-decoding is performed in an adjacent module generated by the MIX software (see fig. 1–3 on page 6). This module also generates read-error in case of read to unknown address. **Note**: for the purpose of pre-decoding, the registers of different clock domains should be grouped into address ranges to prevent oversized multiplexers.

- Generic design, can be configured for number of clock domains and data width.

- For optimization, one target clock domain can be specified as synchronous to bus clock domain.

Figure 1–3: Multi-Clock-Domain Adapter and Pre-Decoder



1.5. Interface Signals

Note that some signals are optional, depending on the configuration of the register-shell (MIX parameters) and used register-types.

1.5.1. Input Port Signals

	Word Width	Clock domain	Description
mcmd	3	clk_ocp	OCP basic signals (see [1] and [2]).
maddr	14	clk_ocp	
mdata	32	clk_ocp	
mrespaccept	1	clk_ocp	OCP simple extension master response accept (see [1] and [2]).
mreset_n	1	clk_ocp	If 1, resets OCP target state-machine and MCD module; can be used by transaction initiator to abort a transaction (master timeout).
test_en	1	clk_ocp	Test input connected to the internal reset synchronizer. Connect to tm_scanmode_i in Monterey system.
scan_shift_enable	1	clk_ocp	Test input connected to the internal clock-gating cell. Connect to tmc_scan_shift_en_i in Monterey system.
<update>	1	any	If 1 and resp. <update_enable> has been sampled, respective shadow register will be loaded. **Note:** this signal must stay high for some time such that it can be sampled in the config register block clock domain. It is strongly recommended that this signal is stable for at least three clock cycles in the clk_pu_<n> domain.
<update_enable>	1	any	If 1, shadowing with the next (and only next) <update> is enabled. **Note:** this signal must stay high for some time such that it can be sampled in the config register block clock domain.
<update_force>	1	any	If 1, shadow register will be loaded on every clock cycle.
<R_field>	1..32	clk_pu_<n>	The value of a read-only register, provided by the PU; **Note:** it cannot be assumed that the read-data is buffered in the register-shell (clk_pu_<n>) domain. There is at least a read-data buffer in the clk_ocp domain.
<USR_field>_in	1..32	clk_pu_<n>	The value of a readable register that is implemented in the PU. The read-data must be buffered in the PU. It is strongly recommended to hold the read-data until the next read is started (indicated by <USR_field>_rd_p).
<int>_set_p	1..32	clk_pu_<n>	High-active pulse of one clock cycle length that sets a status register bit, i.e. a field of size 1 (or a bit in a larger field) that carries a W1C (write-one-to-clear) attribute.
<USR_field>_trans_done_p	1	clk_pu_<n>	High-active pulse of one clock cycle lengh that acknowledges a forwarded read/write transaction to the field <USR_field> that is implemented in the PU. For writes, this should indicate that the write-data has been transfered. For reads, this merely indicates that the PU has acknowledged the read, but it is not a means to qualify the read-data. Any buffering has to be done in the PU.

Table 1–2: Input port signals

Note: Every shadow-update-signal (e.g. vertical-sync in video data stream) is assigned an enable and a force signal. Every shadowed field has a an update-signal assigned.

1.5.2. Output Port Signals

	Word Width	Clock domain	Description
scmdaccept	1	clk_ocp	OCP basic signals (see [1] and [2]).
sresp	2	clk_ocp	
sdata	32	clk_ocp	
sinterrupt	1	clk_ocp	Sideband signal to (optionally) indicate access time-out
<RW_field>	1..32	clk_pu_<n>	The value of a write-only or read-writable field. For fields that are implemented in the PU, the data must be qualified with <*USR_field*>*_wr_p.*
<USR_field>_wr_p	1	clk_pu_<n>	High-active pulse of one clock cycle lengh that indicates a forwarded write transaction to the field <USR_field> implemented in the PU.
<USR_field>_rd_p	1	clk_pu_<n>	High-active pulse of one clock cycle lengh that indicates a forwarded read transaction to the field <USR_field> implemented in the PU.
to_<update>_<clock>	1	clk_pu_<n>	High-active pulse of one clock cycle length that indicates that the values of the shadowed fields belonging to the <update> group have been taken over to the PU-visible register; if the corresponding <update_force> signal of the fields is set to 1, this signal remains high (-> not a pulse in this case).
<field>_trg_p	1	clk_pu_<n>	High-active pulse of one clock cycle length that indicates to the PU that the field has been written (in case of RW or W fields) or read (in case of R fields) by the OCP bus master. **Note:** the read signal only indicates the read-request in the clk_pu_<n> domain; it does not indicate that the OCP initiator has received the read-data

Table 1–3: Output port signals

1.5.3. Interface OCP Target <-> Register Block/MCDA

Name	Word Width	Direction	Clock Domain	Description
addr	32	->	ocp_clk	Transaction address forwarded from OCP bus
trans_start	1	->	ocp_clk	*SCD-async* and *MCD* configuration: a level change indicates the start of a transaction *SCD-sync* configuration: a high-active pulse of one clock cycle length indicates the start of a transaction
wr_data	32	->	ocp_clk	Write data forwarded from OCP bus. It is valid from the clock cycle where *trans_start* has a new value at least until the clock cycle where *trans_done* has a new value.
rd_wr	1	->	ocp_clk	If 1, indicates a read transaction. If 0, indicates a write transaction. It is valid from the clock cycle where *trans_start* has a new value at least until the clock cycle where *trans_done* has a new value.
trans_done	1	<-	clk_pu_<n>	*SCD-async* and *MCD* configuration: a level change indicates that a transaction has finished on the Register block side *SCD-sync* configuration: a high-active pulse of one clock cycle length indicates that a transaction has finished on the Register block side Depending on the transaction direction and the location of the register storage elements, the *trans_done* is asserted immediately or with a delay (response latency).
rd_data	32	<-	clk_pu_<n>	Read data from the Register block. It is valid from the clock cycle where *trans_done* has a new value at least until the clock cycle where *trans_start* has a new value.

Table 1–4: Output port signals

Name	Word Width	Direction	Clock Domain	Description
rd_err	1	<-	clk_pu_<n>	If 1, indicates a read error (see section 1.6.7.). The signal is valid from the clock cycle where *trans_done* has a new value at least until the clock cycle where *trans_start* has a new value.
wr_err	1	n.a.	clk_pu_<n>	If 1, indicates a write error (see section 1.6.7.). The signal is valid from the clock cycle where *trans_done* has a new value at least until the clock cycle where *trans_start* has a new value. **Note**: the port is not connected in the config register shell, but it is documented here for stand-alone usage of the OCP target.

Table 1–4: Output port signals

1.5.4. Reset and Input Clock Signals

	Word Width	Clock domain	Description
clk_ocp	1	n.a.	Clock of attached OCP bus, identical to the main PU clock.
clk_pu_<n>	1	n.a.	Secondary PU clocks.
rst_ocp_n	1	clk_ocp	Low active asynchronous reset (synchronous de-assertion).
rst_pu_<n>_n	1	clk_pu_<n>	Low active asynchronous reset (synchronous de-assertion).

Table 1–5: Reset and clock signals (names are not mandatory)

1.6. Detailed Description

1.6.1. Transactions

Read and write access is either posted or delayed. *Posted* means that the transaction is acknowledged immediately. For the interface between the OCP target and the Config Register Block this means that *trans_done* is given immediately. For the OCP interface *Delayed* means that the acknowledge is delayed until the register is read or written.

The initiator of a register/field access can be either software (i.e. the CPU) or hardware (the PU). While software read-only, write-only and read/write is supported, only hardware read-only, write-only and read/write-to-clear (for 1-bit status bits) is supported. If this opposes a restriction to the PU, the field must be defined with the USR attribute (see table 1–6)

1.6.2. Register-Master Table

The config register block is *script-generated* using the register-master Excel sheet as a single source. The script is part of the MIX tool. See [3] for further documentation.

Field Access and Attributes

The register-master columns *::dir* and *::spec* specify the field access direction and special access attributes. Table 1–6 lists the possible field/register attributes.

Some *::spec* attributes can be combined where it makes sense. The TRG attribute creates a read-or-write notifier pulse to the PU and can be used to implement mailboxes, trigger a hardware function or notify a hardware function that the CPU has read a status register.

	Hardware access direction	Attribute (::spec)	Transaction type	Comment
R	W	[TRG]	delayed	read-only field
	n.a.[1)]	USR	delayed	read-only field; user-field, transaction is forwarded to PU where the register is implemented; one one such field per register allowed
	n.a.	SHA [, TRG]	delayed	read-only field; shadowed field with associated update signal and update enable/force signals
W	R	[TRG]	posted	write-only field
	n.a.	USR	delayed	
	R	SHA [, TRG]	delayed	
RW	R	[TRG]	posted write, delayed read	read-writable field
	n.a.	USR	delayed	read-writable field; user-field, transaction is forwarded to PU where the register is implemented; **Note:** only one such field per register allowed
	R	SHA [, TRG]	posted write, delayed read	read-writable field; shadowed field with associated update signal and update enable/force signals
	RW	W1C [, TRG]	posted write, delayed read	write-one-to-clear (usually interrupt status bits); hardware is restricted to setting the bit.

Table 1–6: Supported Field-types

1) the register is implemented in the PU instead of the config register block

Conditional Instantiation Of FFs

A feature of the MIX software has been introduced, that makes it optional at compile time (via Verilog parameter), whether the sequential elements for a field are implemented or not. This is useful, if there are many similar register-shells and it is required, that they consist of the same Verilog modules. The distinction can then be made via the parameters in the register-shell instantiation.

If the register-master column with the heading *::cond* exists and the cell value is 1, the respective FFs for field *<field-name>* will be conditionally implemented. The integer parameter that controls this is called *P__<field-name>*. If it is -1, the field is implemented normally. If it is >= 0, the field output will be driven constantly with the value of *P__<field-name>*.

1.6.3. Addressing

All register addresses shall be aligned to the OCP data width, e.g. in the Monterey project they are DWORD-aligned and the two LSBs of the address will not be decoded by the Config Register Block.

1.6.4. Shadowed Fields

A lot of applications require software-controlled take-over of configuration parameters at a definite time of an update signal, e.g. the vertical sync in a video frame. This is the sole purpose of the shadowing mechanism. It is not intended for transferring data from one clock-domain to the other, as it was in earlier implementations of register-shells.

The update signal is generated by the functional logic, usually by the PU the register shell is attached to or to a PU located earlier in the data-pipe. Every update signal has *two* control bits assigned. The control bits are implemented in a central module and routed to the respective register shell.

If both control bits are 0, SW can write new values to the bitfields with the PU still using the old values.

If the enable-bit is written by software the PU will get the new values *only* at the next positive edge of the update signal.

If the force-bit is set, the PU will get the new values immediately and always, without the need to wait for the next edge of the update signal.

Shadowing is supported for reads and writes. For writes, the update signal is typically the vsync in a video frame. Read data is updated once it is valid, e.g. at the end of a measurement.

The take-over event can be optionally forwarded (controlled by a configuration parameter of MIX). This is used by the SCC modules of the Monterey project.

Figure 1–4: Timing diagram for shadowing

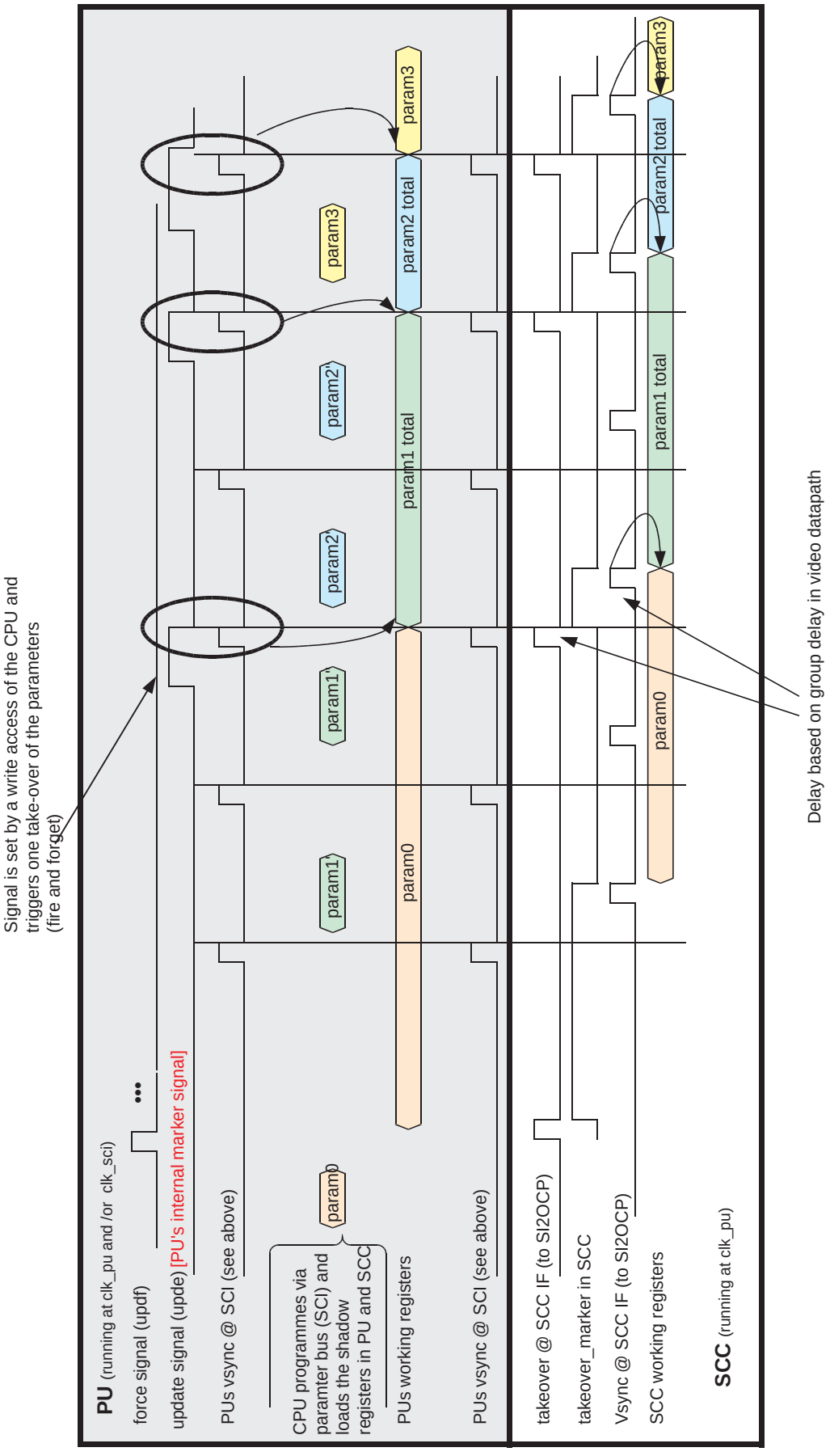
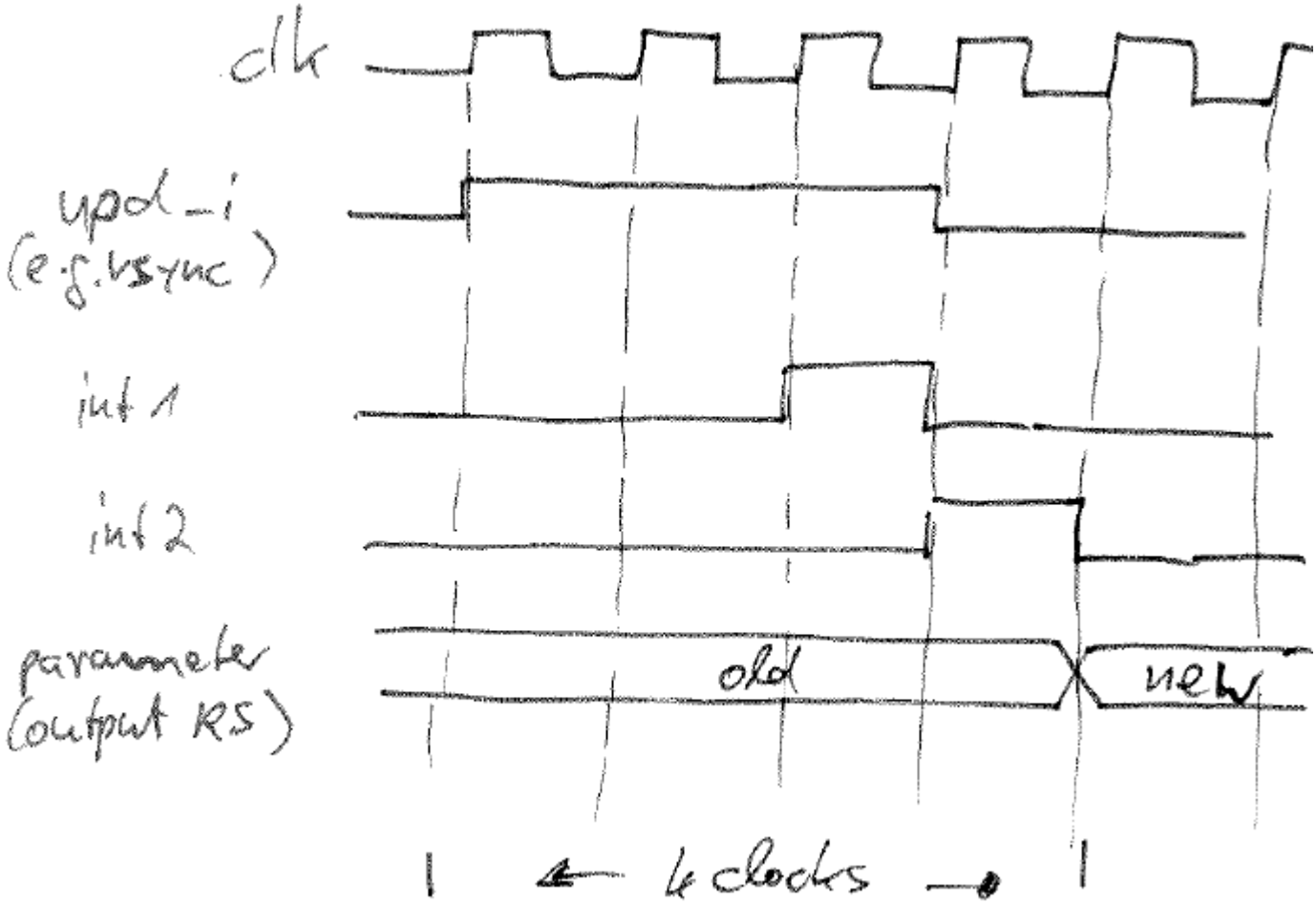

PU (running at clk_pu and /or clk_sci)
force signal (updf)
update signal (upde) [PU's internal marker signal]
PUs vsync @ SCI (see above)
CPU programmes via paramter bus (SCI) and loads the shadow registers in PU and SCC
PUs working registers
PUs vsync @ SCI (see above)
param0
param1'
param2'
param3
param1 total
param2 total
param3
takeover @ SCC IF (to SI2OCP)
takeover_marker in SCC
Vsync @ SCC IF (to SI2OCP)
SCC working registers
SCC (running at clk_pu)
param0
param1 total
param2 total
param3
Signal is set by a write access of the CPU and triggers one take-over of the parameters (fire and forget)
Delay based on group delay in video datapath

Figure 1–5: Exact timing for the synchronous case (PU-clock= OCP bus clock)



1.6.5. Clock-gating For Power Reduction

In order to reduce switching power, the generated HDL can include clock-gating cells for the registers that reside inside the Config Register Block (limited to shadow registers and writable registers). This is controlled by a configuration parameter of the MIX software. The clock to these registers will be enabled for the one clock cycle where the load enable of the registers is active.

The clock gating cell can be controlled/configured by some inputs (see table 1–2), Verilog parameters or Verilog defines to set it in bypass mode or select between possible implementations (technology cell or RTL implementation).

1.6.6. Reset

The *mreset_n* signal synchronously resets the OCP target state-machine. The power-on reset input (from the main PU clock domain) can be synchronized where it crosses a clock-boundary (asynchronous assertion, synchronous deassertion). This is controlled by a configuration parameter of the MIX software. It is connected to the asynchronous reset inputs of all FFs.

1.6.7. Error Handling

Write To Non-Writable Address

The write is discarded. No error signalling available. The OCP target module - if used stand-alone - optionally supports the writeresp_enable OCP parameter. Using this it is possible to forward a write error to the CPU.

Read To Non-Readable Address

One kind of error is “read to non-readable address”. Reads that are not decoded by the config register block return *sresp*=ERR response and *sdata*=0xdeadbeef to the OCP master (see [2]). If a MCD configuration is used, the read-error is generated by the pre-decoder module instead of the config register block.

Target Response Timeout

The other kind of error handling optionally implemented in the register-shell is “target response timeout”. This occurs if the OCP target interface block does not receive a *trans_done* signal in a reasonable time-frame. Reasons for this could be that e.g. a clock is switched off.

The timeout detection circuit and a dedicated, embedded control/status register is optional in the ocp_target module and is enabled via a Verilog instantiation parameter. The MIX software sets this parameter automatically dependent on the existence of the control/status register (reserved name *RS_CTLSTS*) in the register-master (see table 1–7). The timeout detection circuit *must* be inferred if there are registers in a PU clock-domain that are not controlled by the system, e.g. a video clock from an external interface that can be plugged in or out.

The control/status register is embedded in the OCP target and not part of the config register block, because it must always be accessible. If the timeout detection circuit is inferred, the OCP target resets its state-machine in case of time-out. Write data may or may not have reached its target register.

For reads, this error is fed back to the transaction originator via *sresp*=ERR*/sdata=0xdeadbeef* or the *sinterrupt* signal (optionally software configurable). The timeout-value is also optionally software configurable. **Note** that the read-data of the first successful transfer after resolving the blocking at the config register block may be invalid. This depends on the nature and exact point-of-time for the reason that the config register is blocked. For writes, the only method of error-signalling is the *sinterrupt* signal.

Table 1–7: Control/Status Register: RS_CTLSTS Bitfields

Bit	Name	R/W	Function
[19:6]	rs_to_addr[13:0]	R	**Timeout Address** Indicates the address that the last timeout occurred. Will only be updated if rs_to_ists is cleared.
[5:3]	rs_to_val[2:0]	RW or R	**Timeout Value** Start-value of timeout-counter in terms of OCP bus cycles 3'b000 = timeout detection disabled 3'b001 = 16 3'b010 = 32 3'b011 = 64 3'b100 = 128 3'b101 = 256 3'b110 = 512 3'b111 = 1024
[2]	rs_to_rerr_en	RW or R	**Timeout Read-Error Enable** 1 = Timeout is signalled to transaction originator as read-error
[1]	rs_to_ien	RW or R	**Timeout Interrupt Enable** 1 = Timeout is signalled to transaction originator via OCP "SInterrupt"
[0]	rs_to_ists	RW	**Tiimeout Interrupt Status**

Note: if one or more of the fields *rs_to_val*, *rs_to_rerr_en* or *rs_to_ien* is/are set to R (read-only), all of them will be read-only

1.6.8. Read Multiplexer

In case of read, the read data from the PU or the read-shadow register is forwarded to the OCP master. In case of a large decoded address space, the acknowledge for read transactions is delayed by one or more cycles. If the read-path is implemented as a combinatorial mux, this results in a multi-cycle-delay path. The read-path can also be implemented with intermediate FF stages (which is preferred due to DfT reasons). **Note** that read-data is only buffered in the OCP target for the duration of the transfer.

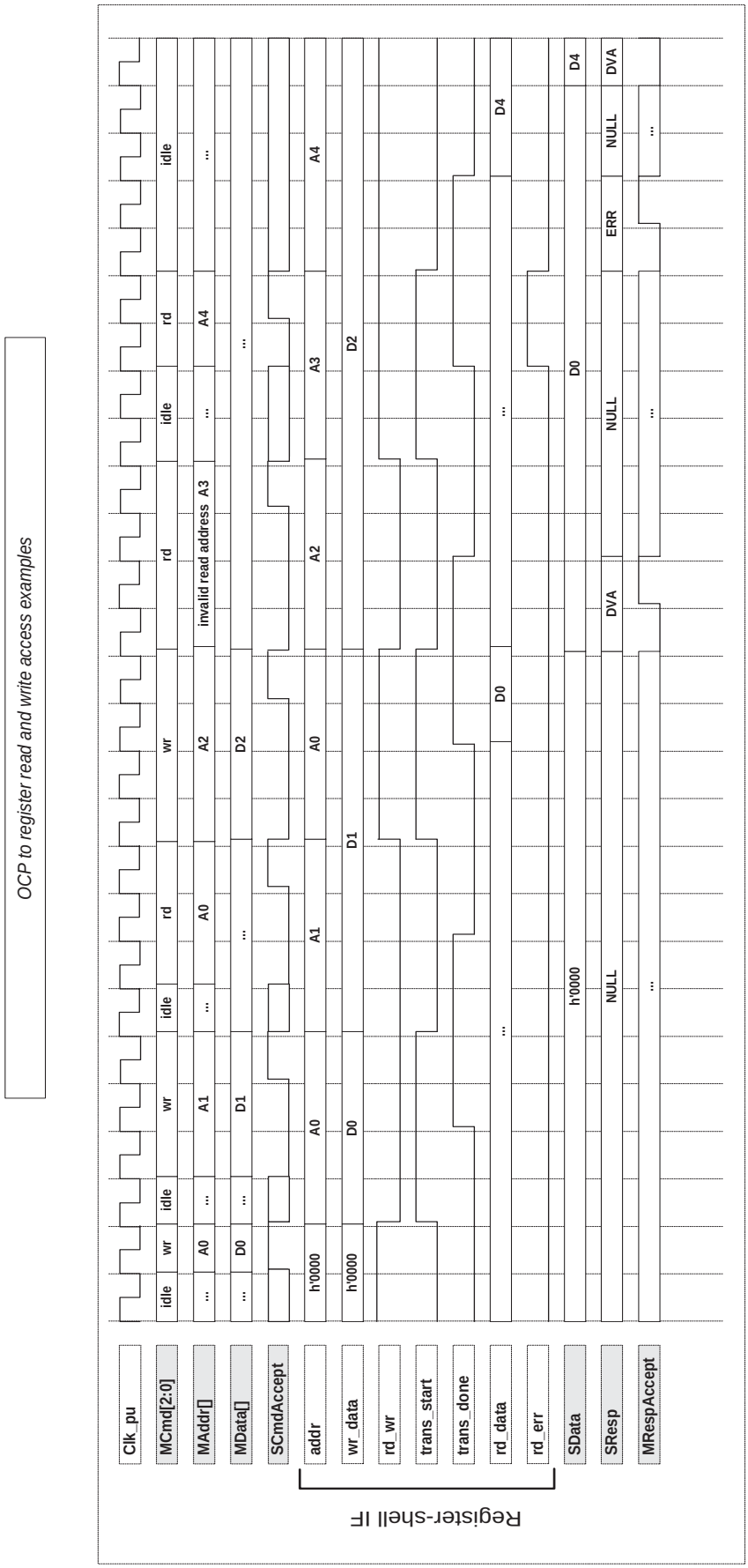
The number of pipeline stages or the insertion of a multi-cycle read-mux is controlled by configuration parameters in the MIX software.

1.6.9. DfT

The generated top-level file of the register-shell has two test inputs. See (see table 1–2) for details.

1.7. Bus Timing Diagrams

Figure 1–6: Example Timings (i/f of OCP target and i/f between OCP target and register block with toggle hs)



1.8. Appendix

1.8.1. Document Reference List

[1] VGCH SCI - SOC infrastructure specification

[2] Open Core Protocol Specification, Release 1.0

[3] Mix documentation in /tools/mix/<release>/doc (Unix) or \\galaxy\development\PROJECTS\MIX (Win)

1.8.2. Terms of Interest and Abbreviations.

	Description
PU	Processing unit, refers to attached IP block without config register shell. Can be anything that needs config registers.
OCP	Open Core Protocol
SCI	System-On-Chip Configuration Interface
field	bit-field or bit-slice of a register: field size greater than register size is not supported
register	physical storage element of fixed size (usually constrained by OCP bus size); contains one or more fields which are the logical elements of a register
domain	set of registers/fields that are addressed by one register-shell
SVA	SystemVerilog assertions
SCD	single-clock-domain configuration (-sync or -async)
MCD	multi-clock-domain configuration
CDC	clock-domain-crossing

Table 1–8: Terms of interest and abbreviations