Cortex-A57 Synchronizer Flops

Processor Division

Document number:	ARM-EPM-022126 2.0
Date of Issue:	26 September 2013
Author:	ARM

Abstract

This document describes and lists the synchronizer modules in Cortex-A57, where the synchronizer modules can be located in the Cortex-A57 RTL bundle, and how to modify the number of synchronizating stages from the default 2-stage.

Contents

1 INTRODUCTION

This document provides a description of the synchronizing flops used in the Cortex-A57 RTL Verilog and lists the instances and filenames.

The control path synchronization flops use a default two-stage synchronization, as shown in Figure 1 below.

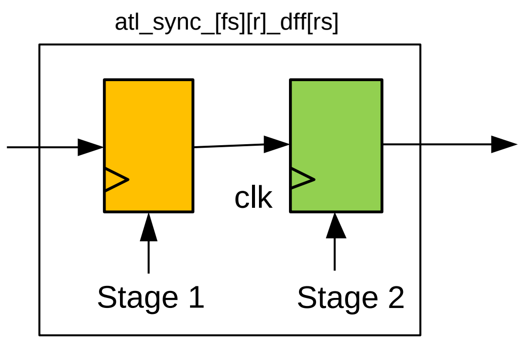

Figure 1: Cortex-A57 default 2-stage synchronization

The data path synchronization flops use a single-stage synchronization, as shown in Figure 2 below. Synchronized control handshake is used to enable data capture for these flops.

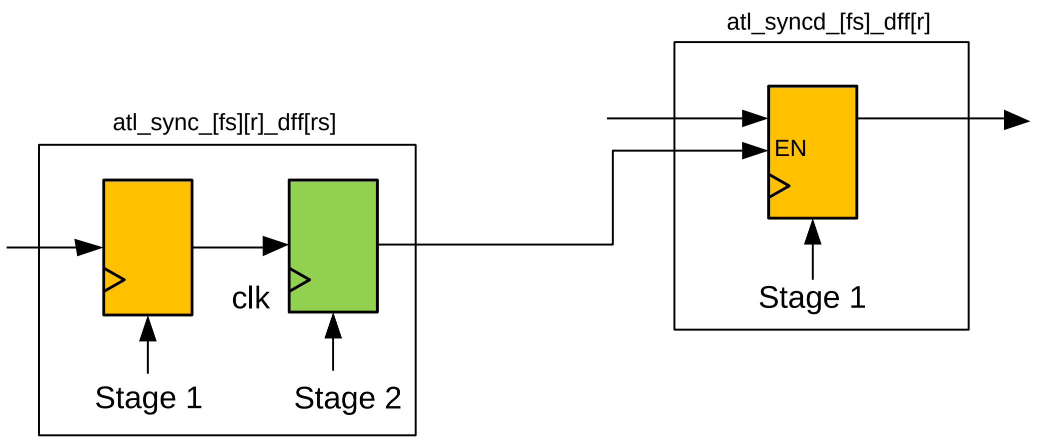

Figure 2: Cortex-A57 1-stage synchronization of data along with 2 stage synchronization of control handshake.

The following sections describe the various synchronizers in the Cortex-A57 RTL, where the synchronizer modules can be located in the Cortex-A57 RTL bundle, and how to modify the number of synchronizating stages from the default 2-stage.

2 SYNCHRONIZER VARIANTS

Most of the synchronizers in Cortex-A57 have the following naming convention:

atl_sync[d]_[fs][r]_dff[rs]

where:

f	– denotes fast (CLK) domain
s	– denotes slow (PCLKDBG) domain
dff[rs]	– denotes resettable/settable flop
[fs]r	– denotes synchronizer used for reset pipeline/generation
syncd	– denotes the asynchronous boundary is protected by control handshake that guarantees stable data with setup and hold in the receiving clock domain

The only exception to this naming convention is the CoreSight Cross-Trigger Interface (CTI) IP synchronizers.

Most of the synchronizer modules live in the **logical/models/cell/generic** directory in the Cortex-A57 RTL bundle. The CTI synchronizers lives in the **logical/cscti/verilog** directory.

Number of flop stages inside synchronizers is controlled by the parameter "LEVELS". Each instance of the synchronizer uses either **`ATL_F_SYNC_LEVELS** or **`ATL_S_SYNC_LEVELS** tickdefines to define "LEVELS" parameter. Default value for these tickdefines is 2.

`ATL_F_SYNC_LEVELS and `ATL_S_SYNC_LEVELS are defined in **logical/shared/verilog/atl_header.v.**

Table 1 below lists the synchronizers that are present in Cortex-A57 RTL.

Table 1: Cortex-A57 synchronizer modules

Module name	Type	Parameter used for number of Levels	Reset present	Comment
atl_sync_fr_dffr.v	Synchronizer, 2 stages default	`ATL_F_SYNC_LEVELS	Yes	Synchronizer used for reset pipeline/generation in CLK domain
atl_sync_sr_dffr.v	Synchronizer, 2 stages default	`ATL_S_SYNC_LEVELS	Yes	Synchronizer used for reset pipeline/generation in PCLKDBG domain
atl_sync_f_dff.v	Synchronizer, 2 stages default	`ATL_F_SYNC_LEVELS	No	Synchronizer used for non-reset related signals in CLK domain
atl_sync_f_dffr.v	Synchronizer, 2 stages default	`ATL_F_SYNC_LEVELS	Yes	Synchronizer used for non-reset related signals in CLK domain
atl_sync_f_dffs.v	Synchronizer, 2 stages default	`ATL_F_SYNC_LEVELS	Yes	Synchronizer used for non-reset related signals in CLK domain
atl_sync_s_dff.v	Synchronizer, 2 stages	`ATL_S_SYNC_LEVELS	No	Synchronizer used for non-reset related signals in

	default			PCLKDBG domain
atl_sync_s_dffr.v	Synchronizer, 2 stages default	`ATL_S_SYNC_LEVELS	Yes	Synchronizer used for non-reset related signals in PCLKDBG domain
atl_syncd_f_dffr.v	Synchronizer, single stage	N/A	Yes	Used when the asynchronous boundary is protected by control handshake that guarantees stable data with setup and hold in the receiving CLK domain
atl_syncd_s_dff.v	Synchronizer, single stage	N/A	No	Used when the asynchronous boundary is protected by control handshake that guarantees stable data with setup and hold in the receiving PCLKDBG domain
atl_syncd_s_dffr.v	Synchronizer, single stage	N/A	Yes	Used when the asynchronous boundary is protected by control handshake that guarantees stable data with setup and hold in the receiving PCLKDBG domain
cxcti_en_sync_atl.v	Synchronizer, 2 stages	N/A	Yes	Synchronizer instantiated by Core Sight CTI block in PCLKDBG domain.

NOTE:

The atl_reset_rep2, atl_reset_rep3, and atl_reset_sync modules are NOT synchronizing flops and should not be changed.

The atl_syncd_[fs]_dff[r] modules do NOT require a 2nd-stage synchronizing flop as they are only used when the asynchronous boundary is protected by control handshake that guarantees stable data with setup and hold in the receiving clock domain.

3 SYNCHRONIZER LIST

Synchronizers inside Cortex-A57 are grouped into following 4 groups.

1. Top-level asynchronous resets synchronized in CLK domain.
2. Top- level asynchronous resets synchronized in PCLKDBG domain.
3. Top-level input ports (other than reset ports) and internal signals from PCLKDBG domain synchronized in CLK domain.
4. Top-level input ports (other than reset ports) and internal signals from CLK domain synchronized in PCLKDBG domain.

In Cortex-A57, each of the 4 groups has a different synchronizer wrapper module, but for all 4 groups there is only a single module for both stage 1 and stage 2 of the synchronizer. Having a single wrapper module for both stages of the synchronizer allows for easier replacement of the synchronizer, and having different wrapper modules for each of the 4 groups provides flexibility for Customers who may only want to replace some of the synchronizers.

- Group 1 is atl_sync_fr_dffr.v
- Group 2 is atl_sync_sr_dffr.v
- Group 3 is atl_sync_f_dff[rs].v
- Group 4 is atl_sync_s_dff[r].v and cxcti_en_sync_atl.v

The synchronizers belonging to Groups 1 and 2 are used for reset generation/pipelining. It is recommended that if Customers do have specific requirements for ‘N’ stages of synchronizer flops in Groups 1 and/or 2, that the appropriate atl_sync_[fs]r_dffr modules or ATL_*_SYNC_LEVELS tickdefines be changed for synchronizers belonging to Groups 1 and/or 2. If Customers do change synchronizer modules or LEVELS tickdefines in Group1, then they must make sure that all instances of synchronizer modules in Group 1 use the same number of ‘N’ stages through out the design inside Cortex-A57. Similary all instances of synchronizer modules in Group 2 need to have the same number of ‘N’ stages through out Cortex-A57.

The synchronizers belonging to Groups 3 and 4 are used for synchronizing signals that cross the asynchronous boundary to/from CLK and PCLKDBG domains. It is recommended that if Customers do have specific requirements for ‘N’ stages of synchronizer flops in Groups 3 and/or 4, that the appropriate atl_sync_[fs]_dff[rs] or ATL_*_SYNC_LEVELS tickdefines or cxcti_en_sync_atl modules be changed for synchronizers belonging to Groups 3 and/or 4.

You can do a Unix grep to get the full list of all the synchronizers in the RTL.
E.g. egrep “atl_sync_f | atl_sync_s” logical/verilog/*.v

The following lists the instances (and filenames) in each of the 4 groups mentioned above for the 4 cpu configuration. Please note that this list is from the **Cortex-A57 r0p0-00lac0** snapshot of the database, and the exact instances may change slightly in a future revision of the database.

Group 1: Following lists the instances of atl_sync_fr_dffr

uck_reset_b_l2cpu_n	(in logical/atl_clock/verilog/atl_ck_cpu_logic.v)
uck_reset_b_l2dt_n	(in logical/atl_clock/verilog/atl_ck_cpu_logic.v)
uck_reset_b_n	(in logical/atl_clock/verilog/atl_ck_cpu_reset.v)
uck_reset_b_dt_n	(in logical/atl_clock/verilog/atl_ck_cpu_reset.v)
uck_reset_b_por_n	(in logical/atl_clock/verilog/atl_ck_cpu_reset.v)
uck_reset_b_cpupor_n	(in logical/atl_clock/verilog/atl_ck_dt_logic.v)
uck_treset_b_dt_n	(in logical/atl_dbgtrace/verilog/atl_dt_sb_ctl.v)
uck_reset_b_l2_n	(in logical/atl_level2/verilog/atl_l2_clken_ctl.v)

Group 2: Following lists the instance of atl_sync_sr_dffr

uck_reset_b_dbg_pclk_n_q	(in logical/atl_dbgtrace/verilog/atl_dt_presetgen.v)

Group 3: Following lists the instances of atl_sync_f_dff[rs]

ur_req_c2	(in logical/atl_clock/verilog/atl_ck_cpu_logic.v)
uck_dt_wfx_wakeup_c2	(in logical/atl_clock/verilog/atl_ck_cpu_logic.v)
uck_dt_apb_active_c2	(in logical/atl_clock/verilog/atl_ck_cpu_logic.v)
uck_dt_noclkstop_dly_q	(in logical/atl_clock/verilog/atl_ck_cpu_reset.v)
uporeset_status_ack_q	(in logical/atl_clock/verilog/atl_ck_dt_logic.v)
ucti_et_extout_ack_sync_q_3_0	(in logical/atl_dbgtrace/verilog/atl_dt_ct.v)
ucti_pmuirq_ack_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_ct.v)
ucti_dbg_trigger_ack_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_ct.v)
ucti_et_extin_sync_q_3_0	(in logical/atl_dbgtrace/verilog/atl_dt_ct.v)
ucti_dbgrestart_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_ct.v)
ucti_edbgrq_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_ct.v)
uedbgrq_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_ct.v)
udbif_req_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_db_main.v)
uedecr_osuce_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_db_main.v)
uedacr_frc_idleack_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_db_main.v)
uedprcr_corepurq_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_db_main.v)
uedecr_rce_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_db_main.v)
uedecr_ss_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_db_main.v)
upmusnapshot_req_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pm.v)
usystem_nsei_q	(in logical/atl_intctrl/verilog/atl_ic_interface.v)
usystem_nrei_q	(in logical/atl_intctrl/verilog/atl_ic_interface.v)

ulegacy_nirq_q	(in logical/atl_intctrl/verilog/atl_ic_interface.v)
ulegacy_nfiq_q	(in logical/atl_intctrl/verilog/atl_ic_interface.v)
uexternal_nvirq_q	(in logical/atl_intctrl/verilog/atl_ic_vinterface.v)
uexternal_nvfiq_q	(in logical/atl_intctrl/verilog/atl_ic_vinterface.v)
uexternal_nvsei_q	(in logical/atl_intctrl/verilog/atl_ic_vinterface.v)
uhwflushreq_q	(in logical/atl_level2/verilog/atl_l2_arbitration.v)
ul2qreq_q	(in logical/atl_level2/verilog/atl_l2_clken_ctl.v)
udt_wfx_wakeup_reg_o	(in logical/atl_reg_rep/verilog/atl_dsncpu_reg_rep.v)
ueventi_c1	(in logical/atlas/verilog/atl_cpu_io.v)
uclrexmonreq_c1	(in logical/atlas/verilog/atl_cpu_io.v)
udbgen_cpu0_o	(in logical/atlas/verilog/atl_cpu_io.v)
uniden_cpu0_o	(in logical/atlas/verilog/atl_cpu_io.v)
uspiden_cpu0_o	(in logical/atlas/verilog/atl_cpu_io.v)
uspniden_cpu0_o	(in logical/atlas/verilog/atl_cpu_io.v)
udbgen_cpu1_o	(in logical/atlas/verilog/atl_cpu_io.v)
uniden_cpu1_o	(in logical/atlas/verilog/atl_cpu_io.v)
uspiden_cpu1_o	(in logical/atlas/verilog/atl_cpu_io.v)
uspniden_cpu1_o	(in logical/atlas/verilog/atl_cpu_io.v)
udbgen_cpu2_o	(in logical/atlas/verilog/atl_cpu_io.v)
uniden_cpu2_o	(in logical/atlas/verilog/atl_cpu_io.v)
uspiden_cpu2_o	(in logical/atlas/verilog/atl_cpu_io.v)
uspniden_cpu2_o	(in logical/atlas/verilog/atl_cpu_io.v)
udbgen_cpu3_o	(in logical/atlas/verilog/atl_cpu_io.v)
uniden_cpu3_o	(in logical/atlas/verilog/atl_cpu_io.v)
uspiden_cpu3_o	(in logical/atlas/verilog/atl_cpu_io.v)
uspniden_cpu3_o	(in logical/atlas/verilog/atl_cpu_io.v)

Group 4: Following lists the instances of atl_sync_s_dff[r]

uctmchin_sync_q_3	(in logical/atl_dbgtrace/verilog/atl_dt_pct_ctm.v)
uctmchin_sync_q_2	(in logical/atl_dbgtrace/verilog/atl_dt_pct_ctm.v)
uctmchin_sync_q_1	(in logical/atl_dbgtrace/verilog/atl_dt_pct_ctm.v)
uctmchin_sync_q_0	(in logical/atl_dbgtrace/verilog/atl_dt_pct_ctm.v)
uctmchoutack_sync_q_3	(in logical/atl_dbgtrace/verilog/atl_dt_pct_ctm.v)
uctmchoutack_sync_q_2	(in logical/atl_dbgtrace/verilog/atl_dt_pct_ctm.v)
uctmchoutack_sync_q_1	(in logical/atl_dbgtrace/verilog/atl_dt_pct_ctm.v)
uctmchoutack_sync_q_0	(in logical/atl_dbgtrace/verilog/atl_dt_pct_ctm.v)

udbgen_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uniden_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uspiden_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uspniden_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uedbgrq_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
ucryptodisable_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
ugiccdisable_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uck_dt_standbywfx_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uck_dt_wfx_ack_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uck_poreset_status_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
ucoredbg_in_reset_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uhalt_ack_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uos_double_lock_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uwfx_dbg_req_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uhlt_dbgevt_ok_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uet_oslock_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
upmusnapshot_ack_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
udbif_ack_sync_q	(in logical/atl_dbgtrace/verilog/atl_dt_pdb_main.v)
uclusteridaff1_sync_q_7_0	(in logical/atl_dbgtrace/verilog/atl_dt_presetgen.v)
uclusteridaff2_sync_q_7_0	(in logical/atl_dbgtrace/verilog/atl_dt_presetgen.v)

Following lists the instances of cxcti_en_sync_atl

u_ct_ack_sync	(in logical/cxcti/verilog/cxcti_apb_if_atl.v)
u_cxcti_ciin_sync_0	(in logical/cxcti/verilog/cxcti_ci_atl.v)
u_cxcti_ciin_sync_1	(in logical/cxcti/verilog/cxcti_ci_atl.v)
u_cxcti_ciin_sync_2	(in logical/cxcti/verilog/cxcti_ci_atl.v)
u_cxcti_ciin_sync_3	(in logical/cxcti/verilog/cxcti_ci_atl.v)
u_cxcti_choutack_sync_0	(in logical/cxcti/verilog/cxcti_ci_atl.v)
u_cxcti_choutack_sync_1	(in logical/cxcti/verilog/cxcti_ci_atl.v)
u_cxcti_choutack_sync_2	(in logical/cxcti/verilog/cxcti_ci_atl.v)
u_cxcti_choutack_sync_3	(in logical/cxcti/verilog/cxcti_ci_atl.v)
u_ct_req_sync	(in logical/cxcti/verilog/cxcti_reg_c_atl.v)
u_dbgen_sync	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_niden_sync	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tinsync_0	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tinsync_1	(in logical/cxcti/verilog/cxcti_ti_atl.v)

u_tinsync_2	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tinsync_3	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tinsync_4	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tinsync_5	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tinsync_6	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tinsync_7	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tioutsync_0	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tioutsync_1	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tioutsync_2	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tioutsync_3	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tioutsync_4	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tioutsync_5	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tioutsync_6	(in logical/cxcti/verilog/cxcti_ti_atl.v)
u_tioutsync_7	(in logical/cxcti/verilog/cxcti_ti_atl.v)

4 SYNCRHONIZER ENABLES

Most synchronizer instantiations in Cortex-A57 have the enables tied to 1’b1 like this:

E.g. atlas/verilog/atl_cpu_io.v:

atl_sync_f_dff #(.WIDTH(1), .LEVELS(`ATL_F_SYNC_LEVELS)) udbgen_cpu0_o (.sync_o (dbgen_cpu0_o), .sync_i (dbgen_i[0]), .clk (ck_gclkfr), .enable (**1'b1**));

The only synchronizers where that is not the case are the synchronizers in Group 1 in CLK domain and Groups 2 and 4 in PCLKDBG domain.

E.g. (in CLK domain) atl_dbgtrace/verilog/atl_dt_sb_ctl.v:

atl_sync_fr_dffr #(.WIDTH(1), .LEVELS(`ATL_F_SYNC_LEVELS)) uck_treset_b_dt_n (.sync_o (ck_treset_b_dt_n), .sync_i (1'b1), .clk (ck_gclkfr), .enable (**en_ck_treset_dt**), .reset (ck_treset_dt));

E.g. (in PCLKDBG domain) atl_dbgtrace/verilog/atl_dt_pdb_main.v:

atl_sync_s_dffr #(.WIDTH(1), .LEVELS(`ATL_S_SYNC_LEVELS)) udbgen_sync_q (.sync_o (dbgen_sync_q), .sync_i (DBGEN), .clk (PCLKDBG), .enable (**pclk_en**), .reset (dt_preset3));

For these synchronizing flops, which do not have the enable tied to 1’b1, the enable will be asserted valid long enough so that the synchronization can be achieved for the extended synchronization stage.

5 SYNCHRONIZER STAGES

Instead of replacing the synchronizer modules **inside logical/models/cells/generic** directory, Customers can alternatively update the tickdefines **`ATL_F_SYNC_LEVELS** and/or **`ATL_S_SYNC_LEVELS** to increase the number of stages from default 2-stage synchronization.

`ATL_F_SYNC_LEVELS and `ATL_S_SYNC_LEVELS are defined in **logical/shared/verilog/ atl_header.v.**

Note: `ATL_S_SYNC_LEVELS does not control the number of stages in synchronizer cells of group 4 instantiated by “cxcti_en_sync_atl” module. Customers will need to manually update module **logical/cscti/Verilog/cxcti_en_sync_atl.v**.

Example:

If Customers want to increase the number of stages in synchronizers of group 1 and group 3 to 3, then they will need to have the following code in atl_header.v file.

```
// Synchronization Flop Stages
// =========================
// Number of flop stages in Clock Domain Crossing (CDC) synchronizers.
// Separate defines for fast and slow clocks.
//
`define ATL_F_SYNC_LEVELS 3
`define ATL_S_SYNC_LEVELS 2
```