RTL Design Document

Asynchronous FIFO — Credit Pulse from Synced Pointer

`async_fifo.sv`

Module	`async_fifo`
Version	2.0
Language	SystemVerilog (IEEE 1800-2017)
Target	ASIC / FPGA
Reset	Active-low async (rd domain primary, CDC to wr)
Date	February 20, 2026

Senior RTL Architect

Contents

1 Overview

async_fifo_credit transfers data between two independent clock domains using Gray-coded pointers and 2-FF synchronizers. The write side receives a credit pulse every time a read completes — the pulse is generated directly from the synchronized read pointer, not from a separate toggle signal.

- Gray-coded PTR_W-bit pointers — one bit changes per step
- 2-FF synchronizer on every CDC path — no combinational cross-domain logic
- wr_credit_pulse generated from rd_ptr_gray_sync_wr changes — one pulse per completed read
- Reset originates in rd domain, synchronized into wr domain via 2-FF sync
- Independent clocks, any frequency ratio
- Standard valid/ready handshake on read side

2 Architecture

2.1 Block Diagram

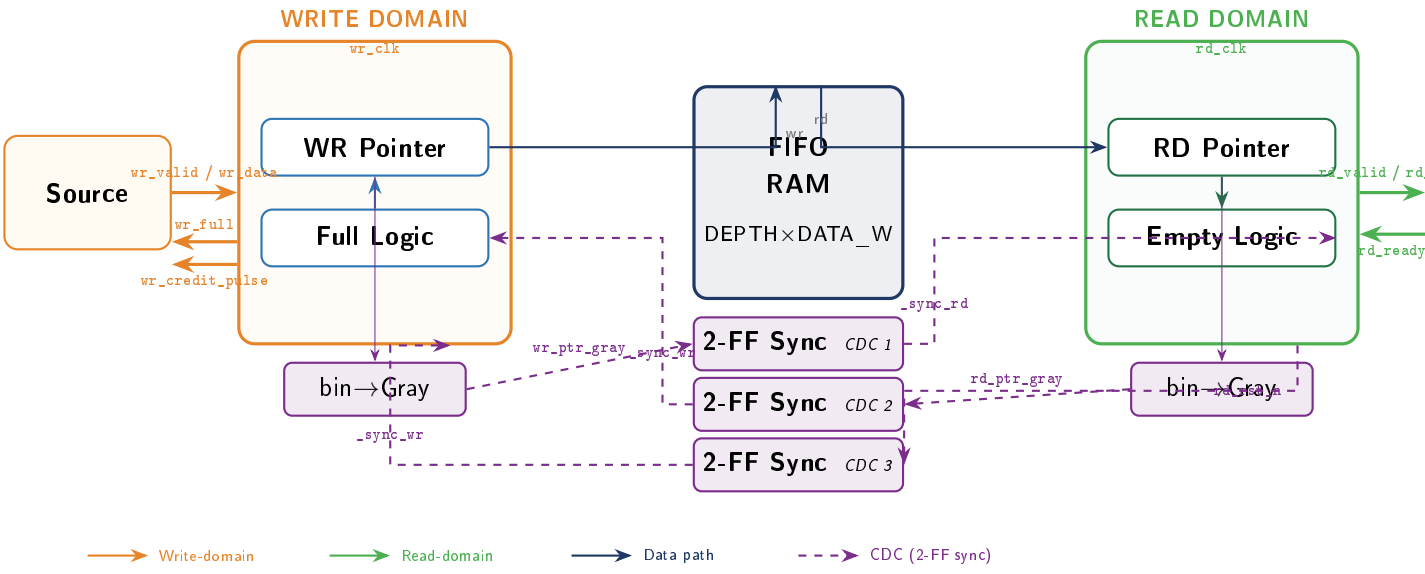

Figure 1: Pointer-sync-only architecture. Three CDC paths: two Gray pointers + reset.

2.2 Module Hierarchy

Module	Instances	Description
cdc_sync	u_sync_wr_ptr u_sync_rd_ptr u_sync_reset	2-FF metastability synchronizer, parameterized width.
async_fifo	top-level	RAM, pointers, full/empty logic, CDC instantiation.

3 Parameters

Parameter	Default	Derivec	Description
DEPTH	16	No	FIFO depth. Power-of-2 required.
ADDR_W	4	Yes	$clog2(DEPTH). RAM address bits.
PTR_W	5	Yes	ADDR_W+1. MSB is wrap bit.
DATA_W	32	No	Data bus width in bits.

4 Port List

4.1 Write Domain

Port	Dir	Width	Description
wr_clk	in	1	Source clock.
wr_valid	in	1	Source has data. Write accepted on wr_valid && !wr_full.
wr_data	in	DATA_W	Write data bus.
wr_credit_pulse	out	1	One 1-cycle pulse per completed read. Derived from rd_ptr_gray_sync_wr.
wr_full	out	1	FIFO full. Source must not write when asserted.

4.2 Read Domain

Port	Dir	Width	Description
rd_clk	in	1	Receiver clock.
rd_rst_n	in	1	Active-low async reset (primary). Synchronized into wr domain.
rd_valid	out	1	FIFO non-empty.
rd_ready	in	1	Consumer accepts. Handshake fires on rd_valid && rd_ready.
rd_data	out	DATA_W	Read data, driven combinationally from RAM.

5 CDC Paths

5.1 Summary

#	Signal	Width	Direction	Sync clock	Purpose
1	wr_ptr_gray	PTR_W	wr→rd	rd_clk	Empty detect
2	rd_ptr_gray	PTR_W	rd→wr	wr_clk	Full detect
3	rd_rst_n	1	rd→wr	wr_clk	Reset sync

5.2 Path 1 — Write Pointer to Read Domain

wr_ptr_bin is Gray-encoded before leaving the write domain. After 2-FF sync into rd_clk:

$$\texttt{rd\_empty} = (\texttt{rd\_ptr\_gray} == \texttt{wr\_ptr\_gray\_sync\_rd})$$

5.3 Path 2 — Read Pointer to Write Domain

After sync into wr_clk, Gray is decoded to binary for full comparison:

$$\texttt{wr\_full} = (\text{addr bits equal}) \wedge (\text{wrap bits differ})$$

The synchronized read pointer rd_ptr_gray_sync_wr is also used to generate wr_credit_pulse. A shadow register holds the previous value; when the two differ, one read has completed and crossed the CDC boundary:

$$\texttt{wr\_credit\_pulse} = (\texttt{rd\_ptr\_gray\_sync\_wr} \neq \texttt{rd\_ptr\_gray\_prev\_wr})$$

This fires exactly one 1-cycle pulse per completed read, with no separate toggle signal required.

5.4 Path 3 — Reset Synchronization

rd_rst_n is the primary reset (asserted by the receiver). It is synchronized into the write domain via a dedicated 2-FF sync:

$$\texttt{wr\_rst\_n\_sync} = \texttt{cdc\_sync}(\texttt{rd\_rst\_n}, \texttt{wr\_clk})$$

Both domains see reset when rd_rst_n is low. The write domain sees it 2 wr_clk cycles later due to synchronizer latency.

Why synchronize reset?

Independent resets in each domain can cause a race: if wr_rst_n deasserts before rd_rst_n, the write domain may start writing while the read domain is still resetting. Synchronizing rd_rst_n into the write domain ensures both sides come out of reset in a safe order.

5.5 Why Gray Code is Safe

Binary 0111→1000 changes four bits. A synchronizer sampling mid-transition can land on any of 16 states. Gray code changes exactly one bit per step — the worst outcome is the old or new valid value, never a phantom.

6 Functional Blocks

6.1 RAM

Listing 1: Storage and async read

```
// storage
logic [DATA_W-1:0] mem [0:DEPTH-1];

// Write Domain
always_ff @(posedge wr_clk)
    if (wr_valid && !wr_full)
        mem[wr_ptr_bin[ADDR_W-1:0]] <= wr_data;

assign rd_data = mem[rd_ptr_bin[ADDR_W-1:0]];  // async read
```

6.2 Write Pointer

Listing 2: Write pointer and bin → Gray

```
always_ff @(posedge wr_clk or negedge wr_rst_n_sync)
    if (!wr_rst_n_sync) wr_ptr_bin <= '0;
    else if (wr_valid && !wr_full) wr_ptr_bin <= wr_ptr_bin + 1'b1;

assign wr_ptr_gray = wr_ptr_bin ^ (wr_ptr_bin >> 1); // bin -> Gray
```

6.3 Read Pointer and Empty

Listing 3: Read pointer, empty flag, bin → Gray

```
assign rd_empty = (rd_ptr_gray == wr_ptr_gray_sync_rd);
assign rd_valid = !rd_empty;
assign rd_data  = mem[rd_ptr_bin[ADDR_W-1:0]];

always_ff @(posedge rd_clk or negedge rd_rst_n)
    if (!rd_rst_n) rd_ptr_bin <= '0;
    else if (rd_valid && rd_ready) rd_ptr_bin <= rd_ptr_bin + 1'b1;

assign rd_ptr_gray = rd_ptr_bin ^ (rd_ptr_bin >> 1); // bin -> Gray
```

6.4 Full Flag

Listing 4: Gray-to-binary and full comparison

```
always_comb begin : gray2bin_rd_in_wr
    rd_ptr_bin_wr[PTR_W-1] = rd_ptr_gray_sync_wr[PTR_W-1];
    for (int i = PTR_W-2; i >= 0; i--)
        rd_ptr_bin_wr[i] = rd_ptr_bin_wr[i+1] ^ rd_ptr_gray_sync_wr[i];
end

assign wr_full = (wr_ptr_bin[ADDR_W-1:0] == rd_ptr_bin_wr[ADDR_W-1:0])
   &&
                 (wr_ptr_bin[ADDR_W]      != rd_ptr_bin_wr[ADDR_W]);
```

6.5 Reset Synchronization

Listing 5: Reset CDC from rd to wr domain

```
logic wr_rst_n_sync;

cdc_sync #(.W(1)) u_sync_reset (
    .clk   (wr_clk),
    .rst_n (1'b1),           // this sync never resets (always enabled)
    .d     (rd_rst_n),
    .q     (wr_rst_n_sync)
);
```

6.6 Credit Pulse Generation

Listing 6: Credit pulse from synced read pointer

```
// shadow register holds previous value of synced rd pointer
always_ff @(posedge wr_clk or negedge wr_rst_n_sync)
    if (!wr_rst_n_sync) rd_ptr_gray_prev_wr <= '0;
    else                rd_ptr_gray_prev_wr <= rd_ptr_gray_sync_wr;

// pulse fires when pointer changes (one read completed and crossed CDC)
assign wr_credit_pulse = (rd_ptr_gray_sync_wr != rd_ptr_gray_prev_wr);
```

7 Timing and Reset

7.1 Reset Strategy

`rd_rst_n` is the primary reset source. It directly resets the read domain and is synchronized into the write domain to produce `wr_rst_n_sync`. Both domains reset when `rd_rst_n` is asserted low; the write domain sees it 2 `wr_clk` cycles later.

The reset synchronizer itself (`u_sync_reset`) never resets — its `rst_n` pin is tied to logic `1'b1` so it is always enabled and can safely propagate `rd_rst_n` into the write domain even during reset assertion.

7.2 STA Constraints

Listing 7: SDC / XDC false-path constraints

```
# CDC Path 1: wr_ptr_gray -> rd_clk
set_false_path -from [get_cells {wr_ptr_gray_reg[*]}] \
               -to   [get_cells {u_sync_wr_ptr/s1_reg[*]}]

# CDC Path 2: rd_ptr_gray -> wr_clk
set_false_path -from [get_cells {rd_ptr_gray_reg[*]}] \
               -to   [get_cells {u_sync_rd_ptr/s1_reg[*]}]

# CDC Path 3: rd_rst_n -> wr_clk
set_false_path -from [get_ports {rd_rst_n}] \
               -to   [get_cells {u_sync_reset/s1_reg}]
```

8 Synthesis Notes

FPGA — `mem` infers block RAM in SDP mode. `(* ASYNC_REG = "TRUE" *)` co-locates `s1`/`s2` in the same slice and adds them to Vivado's CDC report automatically. Apply `set_false_path` on all three CDC paths in XDC.

ASIC — Replace each `cdc_sync` instance with the foundry-characterised `sync_cell` hard macro. Replace `mem` with a RAM macro for area/power. No latches inferred.