Contents

Stream Index

Generated: 2025-11-16

STREAM Specification Index

Version: 0.26 **Date:** 2025-10-17 **Purpose:** Complete technical specification for STREAM subsystem

---

Document Organization

Note: All chapters linked below for automated document generation.

Chapter 1: Overview

- Clocks and Reset

Chapter 2: Functional Blocks

- Descriptor Engine
- Scheduler
- AXI Read Engine
- AXI Write Engine
- SRAM Controller
- Simple SRAM
- Channel Arbiter
- APB Config
- MonBus AXIL Group
- Top-Level Integration
- Performance Profiler

Quick Reference

Functional Unit Blocks (FUB)

Module	File	Purpose	Li
descriptor_engine	`stream_fub/descriptor_engine.sv`	Descriptor fetch/parse (256-bit)	~3
scheduler	`stream_fub/scheduler.sv`	Transfer coordinator	~4
axi_read_engine	`stream_fub/axi_read_engine.sv`	AXI read master	~3
axi_write_engine	`stream_fub/axi_write_engine.sv`	AXI write master	~4
sram_controller	`stream_fub/sram_controller.sv`	Per-channel buffer management	~3
simple_sram	`stream_fub/simple_sram.sv`	Dual-port SRAM primitive	~1
perf_profiler	`stream_fub/perf_profiler.sv`	Channel performance profiling	~4

Integration Blocks (MAC)

Module	File	Purpose	Lines	Stat
channel_arbiter	`stream_macro/channel_arbiter.sv`	Priority arbitration	~200	[Pen
apb_config	`regs/stream_regs.rdl` + wrapper	Config registers	~350	[Fut

Module	File	Purpose	Lines	Stat
monbus_axil_group	stream_macro/monbus_axil_group.sv	MonBus + AXIL	~800	[Dor
stream_top	stream_macro/stream_top.sv	Top-level	~500	[Pen

Performance Modes (AXI Engines)

STREAM AXI engines support three performance modes via compile-time parameters, offering area/performance trade-offs from tutorial simplicity to datacenter throughput.

Holistic Overview: V1/V2/V3 Working Together

Design Philosophy:

- **V1 (Low):** Tutorial-focused, simple architecture, minimal area
- **V2 (Medium):** Command pipelining hides memory latency, 6.7x throughput improvement
- **V3 (High):** Out-of-order completion maximizes memory controller efficiency, 7.0x throughput

Key Insight: The benefit scales with memory latency. For low-latency SRAM (2-3 cycles), V1 achieves 40% throughput. For high-latency DDR4 (70-100 cycles), V1 drops to 14% but V2/V3 maintain 94-98% throughput.

Parameterization Strategy:

- `ENABLE_CMD_PIPELINE = 0`: V1 (default, tutorial mode)
- `ENABLE_CMD_PIPELINE = 1`: V2 (command pipelined, best area efficiency)
- `ENABLE_CMD_PIPELINE = 1, ENABLE_OOO_DRAIN = 1` (write) or `ENABLE_OOO_READ = 1` (read): V3 (out-of-order, maximum throughput)

V1 - Low Performance (Single Outstanding Transaction)

Architecture: Single-burst-per-request, blocking on completion

- **Area:** ~1,250 LUTs per engine
- **Throughput:** 0.14 beats/cycle (DDR4), 0.40 beats/cycle (SRAM)
- **Outstanding Txns:** 1
- **Use Case:** Tutorial examples, embedded systems, low-latency SRAM
- **Key Feature:** Zero-bubble streaming pipeline (NO FSM!)

Blocking Behavior:

- Write: Blocks on B response before accepting next request
- Read: Blocks on R last before accepting next request
- Simple control: 3 flags (`r_ar_inflight`, `r_ar_valid`, completion tracking)

V2 - Medium Performance (Command Pipelined)

Architecture: Command queue decouples command acceptance from data completion

- **Area:** ~2,000 LUTs per engine (1.6x increase)
- **Throughput:** 0.94 beats/cycle (DDR4), 0.85 beats/cycle (SRAM)
- **Outstanding Txns:** 4-8 (command queue depth)
- **Use Case:** General-purpose FPGA, DDR3/DDR4, balanced area/performance
- **Key Feature:** Hides memory latency via pipelining (6.7x improvement)

Command Pipelining:

- Write: AW queue + W drain FSM + B scoreboard (async response tracking)
- Read: AR queue + R in-order reception (simpler than write, no B channel)
- Per-command SRAM pointers enable multiple bursts from same channel

Best Area Efficiency: 4.19x throughput per unit area (6.7x throughput / 1.6x area)

V3 - High Performance (Out-of-Order Completion)

Architecture: OOO command selection maximizes memory controller efficiency

- **Area:** ~3,500-4,000 LUTs per engine (2.8-3.2x increase)
- **Throughput:** 0.98 beats/cycle (DDR4), 0.92 beats/cycle (SRAM)
- **Outstanding Txns:** 8-16 (larger command queue)
- **Use Case:** Datacenter, ASIC, HBM2, high-performance memory controllers
- **Key Feature:** Flexible drain order based on SRAM data availability (7.0x improvement)

Out-of-Order Mechanisms:

- Write: OOO W drain (select command with SRAM data ready), AXI ID matching for B responses
- Read: OOO R reception (match m_axi_rid to queue entry), independent SRAM write per command
- Transaction ID structure: `{counter, channel_id}` preserves channel routing for MonBus

Why OOO is Naturally Supported:

1. Per-channel SRAM partitioning (no cross-channel hazards)
2. Per-command pointer tracking (no pointer collision)
3. Transaction ID matching (channel ID in lower bits for routing)

Performance Comparison Summary

Configuration	Area	DDR4 Throughput	SRAM Throughput	Area Efficiency	Use
V1	1.0x	0.14 beats/cycle (1.0x)	0.40 beats/cycle (1.0x)	1.00	Tut
V2	1.6x	0.94 beats/cycle (6.7x)	0.85 beats/cycle (2.1x)	4.19	Gen
V3	2.8x	0.98 beats/cycle (7.0x)	0.92 beats/cycle (2.3x)	2.50	Dat

Key Takeaway: V2 offers best area efficiency (4.19x) for typical FPGA use cases. V3 provides marginal throughput improvement (7.0x vs 6.7x) at higher area cost, justified only for high-performance memory controllers that support out-of-order responses.

Clock and Reset Summary

Clock Domains

Clock	Frequency	Usage
`aclk`	100-500 MHz	Primary - all STREAM logic, AXI/AXIL interfaces
`pclk`	50-200 MHz	APB configuration interface (may be async to aclk)

Reset Signals

Reset	Polarity	Type	Usage
`aresetn`	Active-low	Async assert, sync deassert	Primary - all STREAM logic
`presetn`	Active-low	Async assert, sync deassert	APB configuration interface

See: Clocks and Reset for complete timing specifications

Interface Summary

External Interfaces

Interface	Type	Width	Purpose
APB	Slave	32-bit	Configuration registers
AXI (Descriptor)	Master	256-bit	Descriptor fetch
AXI (Read)	Master	512-bit (param)	Source data read
AXI (Write)	Master	512-bit (param)	Destination data write
AXIL (Slave)	Slave	32-bit	Error/interrupt FIFO access
AXIL (Master)	Master	32-bit	MonBus packet logging to memory
IRQ	Output	1-bit	Error interrupt

Internal Buses

Interface	Width	Purpose
MonBus	64-bit	Internal monitoring bus (channels -> monbus_axil_group)

Area Estimates

By Performance Mode

Configuration	Total LUTs	SRAM	Use Case
Low (Tutorial)	~9,500	64 KB	Educational, area-constrained
Medium (Typical)	~11,200	64 KB	Balanced FPGA implementations
High (Performance)	~13,700	64 KB	High-throughput ASIC/FPGA

Breakdown (Low Performance)

Component	Instances	Area/Instance	Total
Descriptor Engine	8	~300 LUTs	~2,400 LUTs
Scheduler	8	~400 LUTs	~3,200 LUTs
AXI Read Engine (Low)	1	~250 LUTs	~250 LUTs
AXI Write Engine (Low)	1	~250 LUTs	~250 LUTs
SRAM Controller	1	~1,600 LUTs	~1,600 LUTs

Component	Instances	Area/Instance	Total
Simple SRAM (internal)	1-8	1024x64B total	64 KB
Channel Arbiter	3	~150 LUTs	~450 LUTs
APB Config	1	~350 LUTs	~350 LUTs
MonBus AXIL Group	1	~1,000 LUTs	~1,000 LUTs
Total	-	-	**~9,500 LUTs + 64KB**

Related Documentation

- **PRD.md** - Product requirements and overview
- **ARCHITECTURAL_NOTES.md** - Critical design decisions
- **CLAUDE.md** - AI development guide
- **Register Generation** - PeakRDL workflow

Specification Conventions

Signal Naming

- **Clock:** `aclk`, `pclk`
- **Reset:** `aresetn`, `presetn` (active-low)
- **Valid/Ready:** Standard AXI/custom handshake
- **Registers:** `r_` prefix (e.g., `r_state`, `r_counter`)
- **Wires:** `w_` prefix (e.g., `w_next_state`, `w_grant`)

Parameter Naming

- **Uppercase:** `NUM_CHANNELS`, `DATA_WIDTH`, `ADDR_WIDTH`
- **String parameters:** `PERFORMANCE` ("LOW", "MEDIUM", "HIGH")

State Machine Naming

```
typedef enum logic [3:0] {
    IDLE   = 4'h0,
    ACTIVE = 4'h1,
    // ...
} state_t;

state_t r_state, w_next_state;  // Current and next state
```

Last Updated: 2025-10-17 **Maintained By:** STREAM Architecture Team