dma_core_wrap Product Guide

// This version is untested.

Module Information

- The dma_core_wrap consists of:
  - axi_to_reg: AXI interface to register interface
  - idma_reg64_frontend: DMA frontend with 64bit-register interface
  - idma_reg64_2d_frontend (IsTwoD): DMA frontend with 2-dimension 64bit-register interface
  - idma_nd_midend (IsTwoD): DMA midend which can convert 2D transfer to 1D transfer
  - stream_fifo: Fifo of request from frontend
  - idma_backend: DMA backend with AXI interface output

Port Description

Signal Name	Interface	Signal Type
axi_mst_req_* / axi_mst_rsp_*	AXI Master	O/I
axi_slv_req_* / axi_slv_rsp_*	AXI Slave	I/O

Parameter Description

Parameter Name	Description	Default Value
AxiAddrWidth	Axi Address Width	64d
AxiDataWidth	Axi Data Width	64d
AxiIdWidth	Axi Id Width	01d
AxiUserWidth	Axi User Width	01d
AxiSlvIdWidth	Axi Slave Id Width	01d
NumAxInFlight	Number of transaction that can be in-flight concurrently	03d
MemSysDepth	The depth of the memory system the backend is attached to	00d
JobFifoDepth	Depth of the stream_fifo	00d
RawCouplingAvail	Should the `R - AW` coupling hardware be present? (recommended)	01d
IsTwoD	Use 2D frontend if 1	00d

Register Address Map

Address Space Offset	Name	Description	Effective Length	Type
00h	IDMA_REG64_FRONTEND_SRC_ADDR_OFFSET	source address	64d	Write

Address Space Offset	Name	Description	Effective Length	Type
08h	IDMA_REG64_FRONTEND_DST_ADDR_OFFSET	destination address	64d	Write
10h	IDMA_REG64_FRONTEND_NUM_BYTES_OFFSET	burst length	64d	Write
18h	IDMA_REG64_FRONTEND_CONF_OFFSET	configuration[1]	03d	Write
20h	IDMA_REG64_FRONTEND_STATUS_OFFSET	status	01d	Read
28h	IDMA_REG64_FRONTEND_NEXT_ID_OFFSET	next id	64d	Read
30h	IDMA_REG64_FRONTEND_DONE_OFFSET	done	64d	Read

Reset Value	Name
00h	IDMA_REG64_FRONTEND_STATUS_RESVAL
00h	IDMA_REG64_FRONTEND_NEXT_ID_RESVAL
00h	IDMA_REG64_FRONTEND_DONE_RESVAL

Configuration Register Details

Configuration Name	Index	Description
decouple	0	Couples the `R` to the `AW` channel by keeping writes back until the correspondingreads arrive at the DMA. This reduces the congestion in the memory system. Decouples the `R` and `W` channels completely can cause deadlocks. decouple_aw is always 0 and decouple_rw is controlled by this register. decouple_rw be 0 is preferred.
deburst	1	This register controls src_reduce_len and dst_reduce_len, which means should bursts be reduced in length? The transfers split into smaller chunks than 4KB if 1.
serialize	2	Not supported

Programming Sequence

1. Verify Status register = 0, i.e. backend is idle.
2. Write the Configuration register as Configuration Register Details Table if needed.
3. Write the desired transfer source address to the Source Address (SA) register. The transfer data at the source address must be valid and ready for transfer.
4. Write the desired transfer destination address to the Destination Address (DA) register.
5. Write the number of bytes to transfer to the CDMA Bytes to Num Bytes register. Writing to the Num Bytes register also starts the transfer.
6. The Done register presents transfer completed.
7. Ready for another transfer. Go back to step 1.

1. Configuration includes decouple and deburst (serialize is no longer supported), detailed in Configuration Register Details Table. ↩