offload

Address width: 32

Data width: 32

Base address: 0x20001000

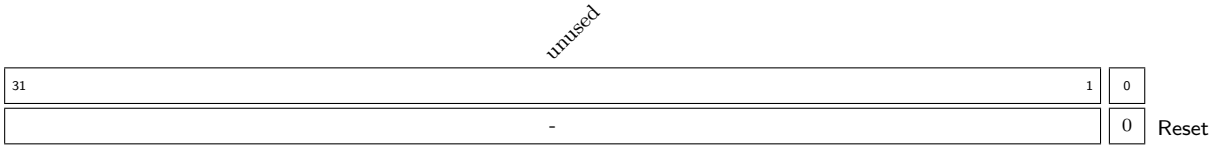
Offloading data from alpide_data modules, and stores them in AXI Stream FIFO for offloading. The module selects a alpide_data module for offloading based on the number of words stored in the alpide_data buffer FIFO. The module supports two modes: frame-based, and non-frame based. Frame-based mode will read an entire frame from an alpide_data module before evaluating to read from another, even though not a full frame is yet store in the selected alpide_data module. This may be less efficient (as the ALPIDE data stream might contain gaps), but the resulting data do not mix between frames from different ALPIDE chips. The non-frame mode will read a constant number of words from the alpide_data module based. Tlast functionality is added to support DMA usage.

1 Register List

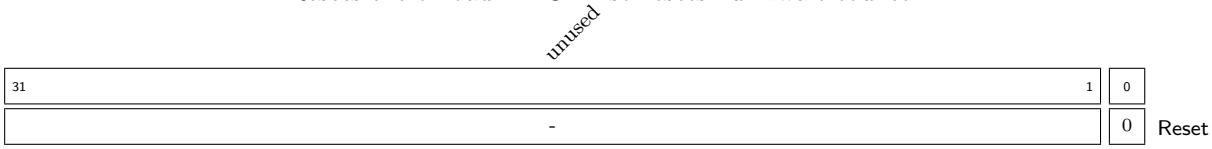
#	Name	Mode	Address	Type	Length	Reset
0	enable_offload	RW	0x00000000	SL	1	0x0
1	reset_offload_fifo	PULSE	0x00000004	SL	1	0x0
2	frame_based_offload	RW	0x00000008	SL	1	0x1
3	idle	RO	0x0000000C	SL	1	0x0
4	pDTP_TX_status	RO	0x00000010	FIELDS	28	0x0
5	test_mode	RW	0x00000014	FIELDS	32	0x0
6	num_test_words	RW	0x00000018	DEFAULT	32	0x400
7	num_wait	RO	0x0000001C	DEFAULT	32	0x0
8	tlast_threshold	RW	0x00000020	DEFAULT	32	0xFFFF
9	assert_tlast_when_empty	RW	0x00000024	SL	1	0x0
10	flush_buffer	PULSE	0x00000028	SL	1	0x0

2 Registers

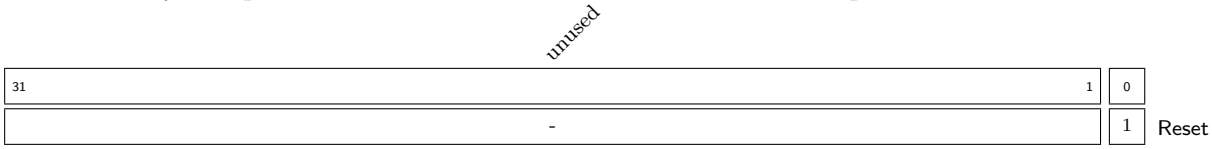
Register 2.1: ENABLE_OFFLOAD - RW (0x00000000)
Enable offload state machine. If unasserted, no data is read out from ALPIDE_DATA buffer FIFOs.



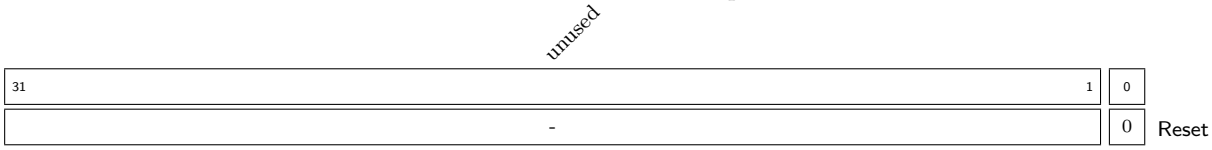
Register 2.2: RESET_OFFLOAD_FIFO - PULSE FOR 10 CYCLES - (0x00000004)
Resets the offload FIFO. Also resets num_wait counter.



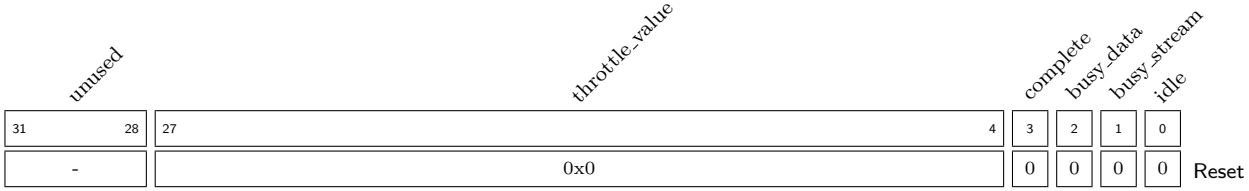
Register 2.3: FRAME_BASED_OFFLOAD - RW (0x00000008)
If enabled, the offload module will read a complete frame from an ALPIDE_DATA module buffer FIFO before reading from another ALPIDE_DATA module. Somewhat reduced efficiency (depending on gaps in ALPIDE chip data stream), but ensures that no data is mixed between ALPIDE chips. However, mixed frames may be separated from each other in software, because of the pRU data format structure.



Register 2.4: IDLE - RO (0x0000000C)
Is low whenever there are more data to transmit from alpide_data modules to offload buffer.

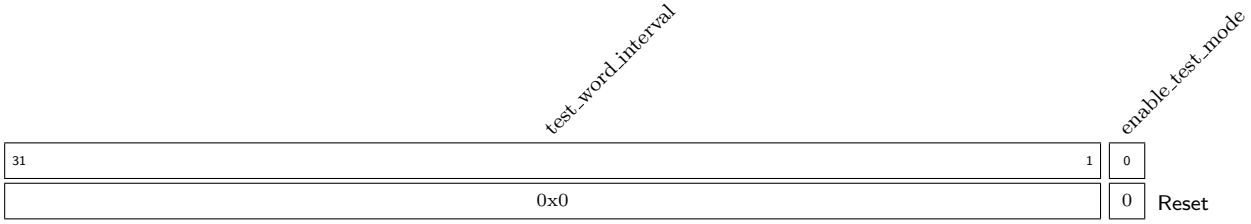


Register 2.5: PDTP_TX_STATUS - RO (0x00000010)
Test offload by filling FIFO with counter value.



idle	pDTP TX Module is idle.
busy_stream	pDTP TX Module is transmitting stream.
busy_data	pDTP TX Module is transmitting data.
complete	pDTP TX Module has completed previous task.
throttle_value	Current throttle value.

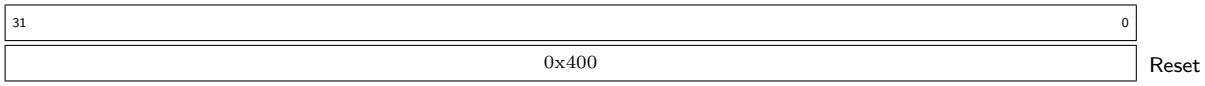
Register 2.6: TEST_MODE - RW (0x00000014)
Test offload by filling FIFO with counter value.



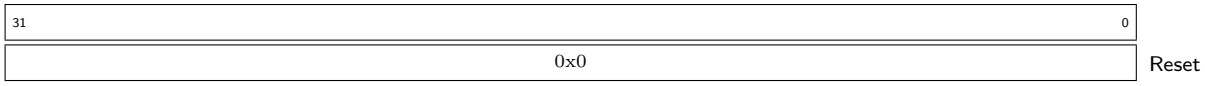
enable_test_mode Enable test mode.

test_word_interval The clock cycle interval between each test word. 0 gives 128bit x 120MHz, 1 gives 128bit x 60MHz, 2 gives 128 x 40 MHz, etc.

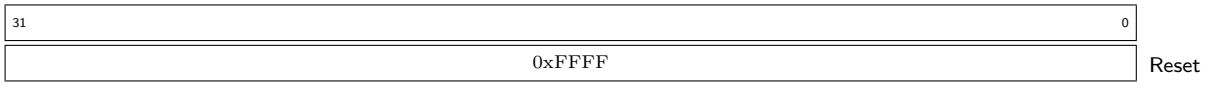
Register 2.7: NUM_TEST_WORDS - RW (0x00000018)
Number of words to be stored in offload FIFO in test mode. Last word is written with tlast.



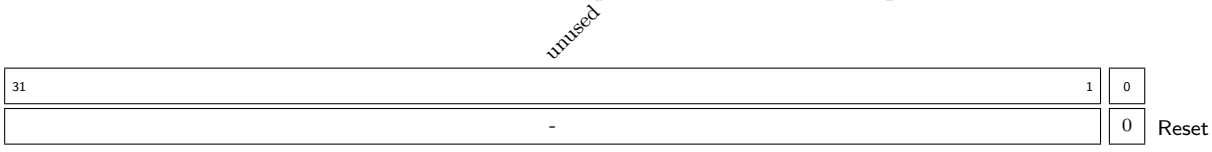
Register 2.8: NUM_WAIT - RO (0x0000001C)
Number of clock cycles waiting for FIFO to have space for writing.



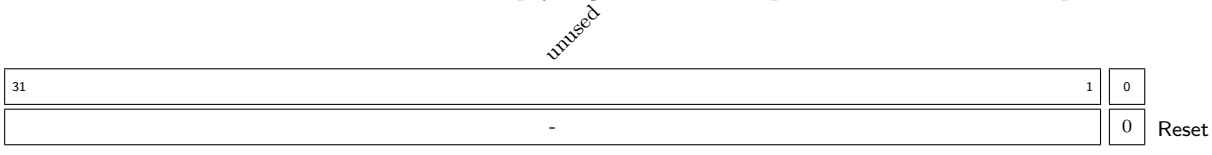
Register 2.9: TLAST_THRESHOLD - RW (0x00000020)
Note: deprecated for all HW except PTB. Threshold for number of 128 bit words written to FIFO before tlast is asserted. This number must be smaller than the DMA length register value to avoid DMA error.



Register 2.10: ASSERT_TLAST_WHEN_EMPTY - RW (0x00000024)
Note: deprecated for all HW except PTB. Assert TLAST when the last word is read out of the FIFO. This causes the DMA to assert an IRQ, and ensures that no data is lost in the buffer. Can be replaced by flush_buffer assertion. Note: deprecated for all HW except PTB.



Register 2.11: FLUSH_BUFFER - PULSE FOR 1 CYCLES - (0x00000028)
Note: deprecated for all HW except PTB. Causes a delimiter word (all ones) to be written to the output buffer FIFO with TLAST asserted. Will provoke the DMA to assert an IRQ so the transfer is fulfilled. Can be used instead of assert_tlast_when_empty register. Note: deprecated for all HW except PTB.



3 Example VHDL Register Access

All registers are bundled in records based on their mode. E.g. all RW registers are accessed through the record *bustype_rw_regs*. Access is also dependent on the type of register. All register of type SL, SLV and DEFAULT are all directly accessed by just specifying the mode record signal. E.g. the RW register *reg0* can be assigned a value like this (assuming AXI-bus):

```
axi_rw_regs.reg0 <= (others => '0');
```

Registers of type FIELD cannot be directly accessed without specification of a certain field. This is because the registers are implemented as a record in VHDL (thus a record of records). E.g. if the RO register *reg1* contains the field *field3* it can be accessed like this (assuming AXI-bus):

```
axi_ro_regs.reg1.field3 <= (others => '0');
```