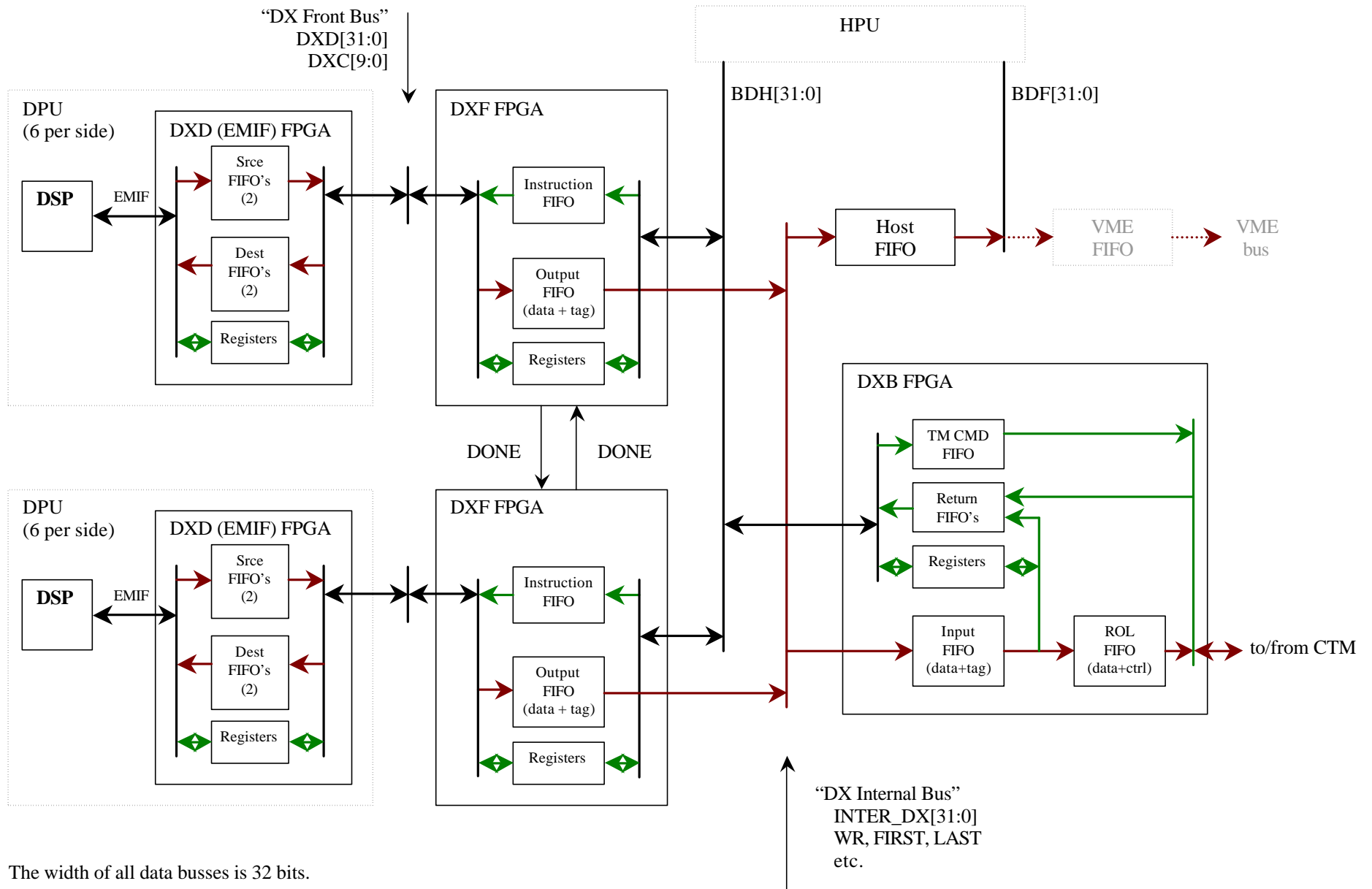
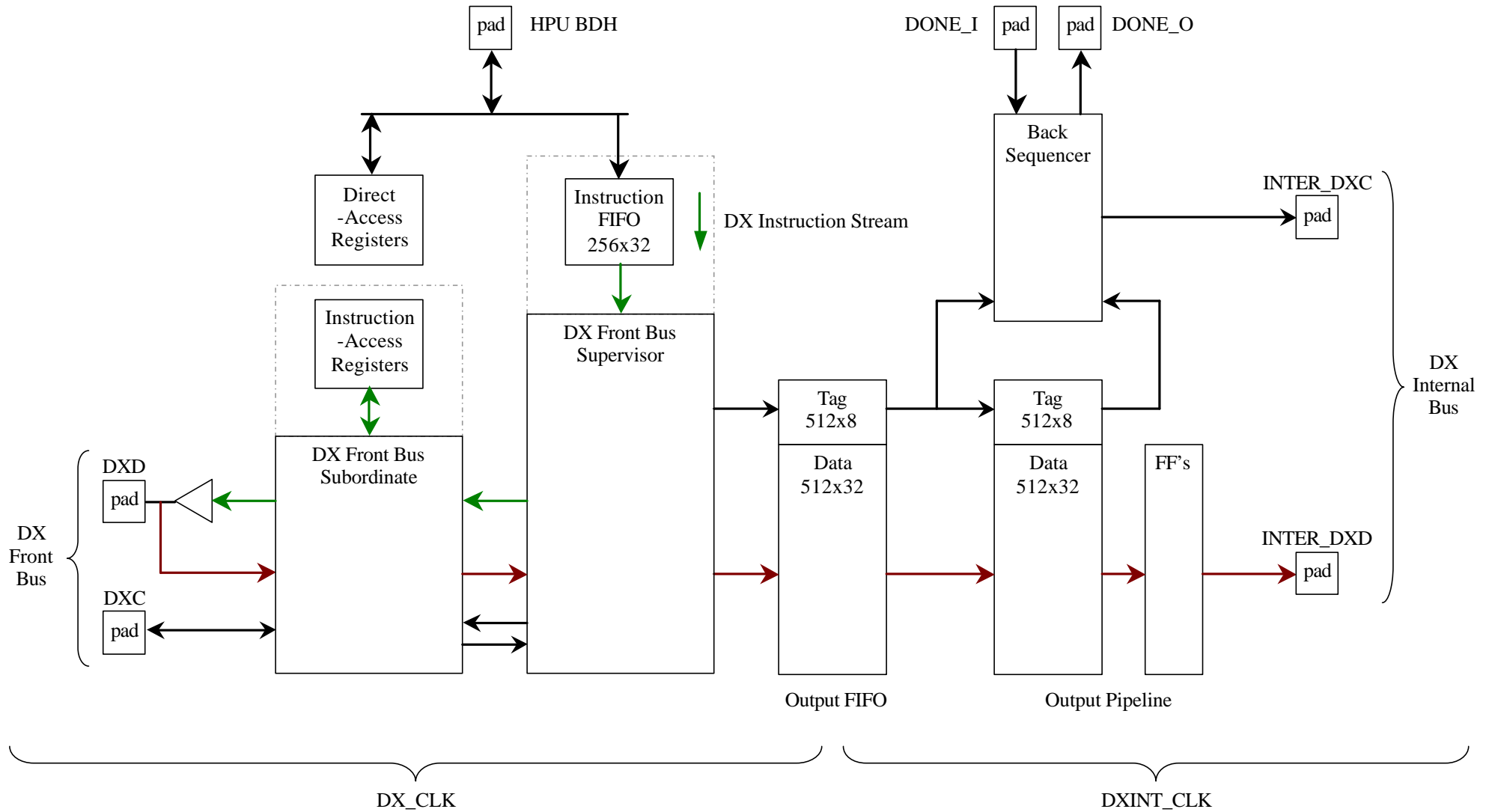


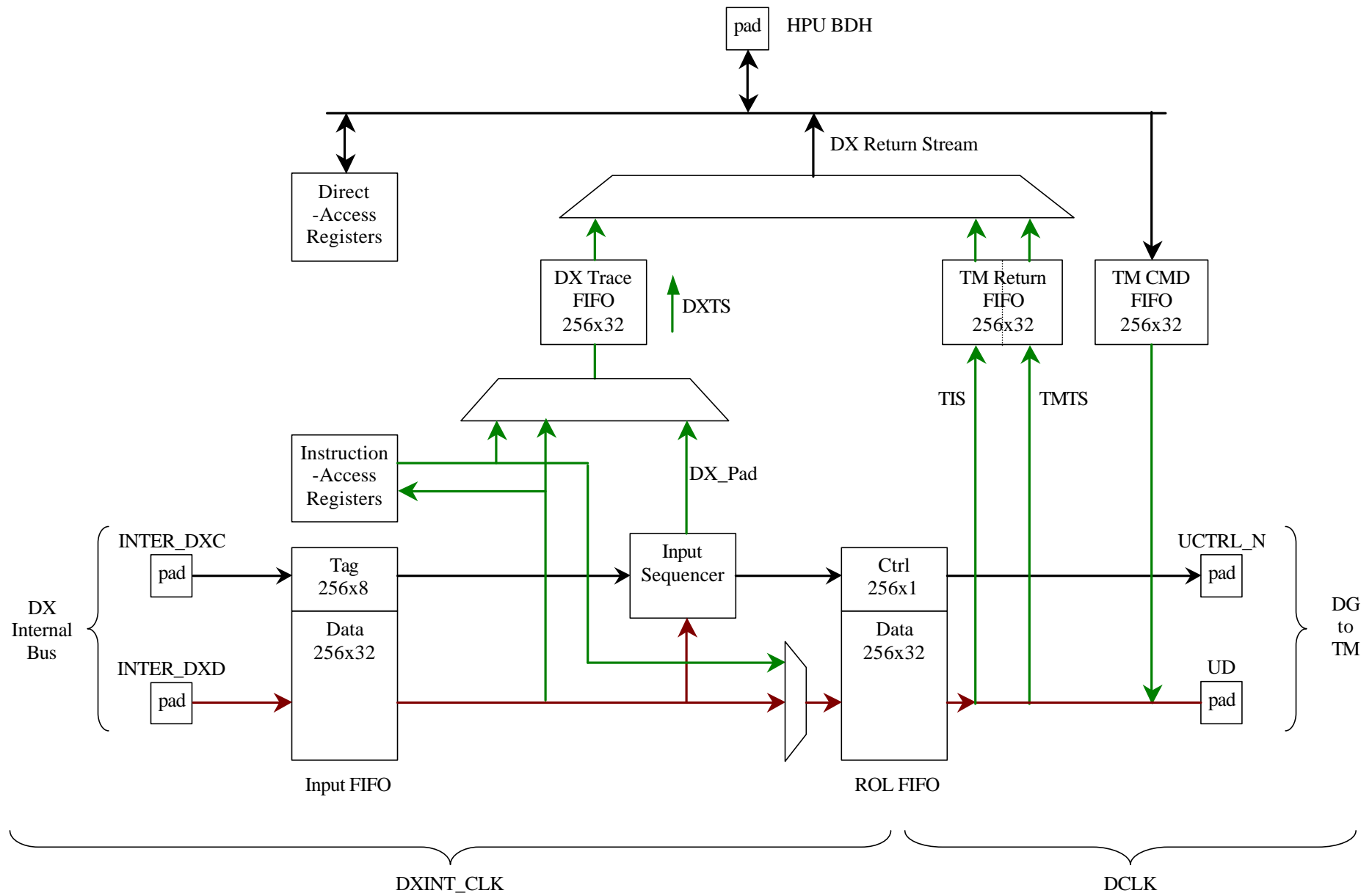
Data Exchange (DX) Overview



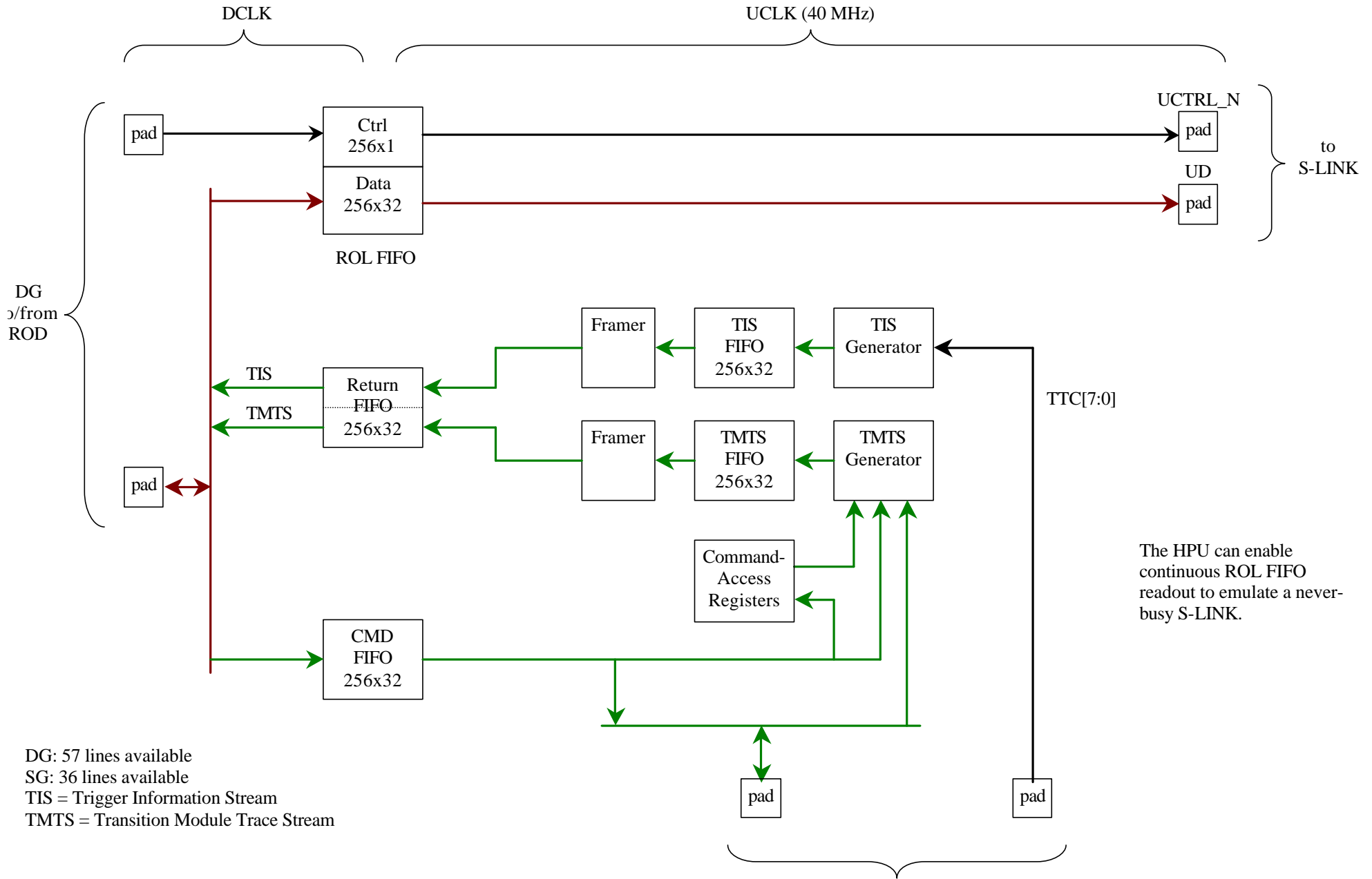
DXF FPGA Detail



Not shown: All I/O except HPU I/O is registered.



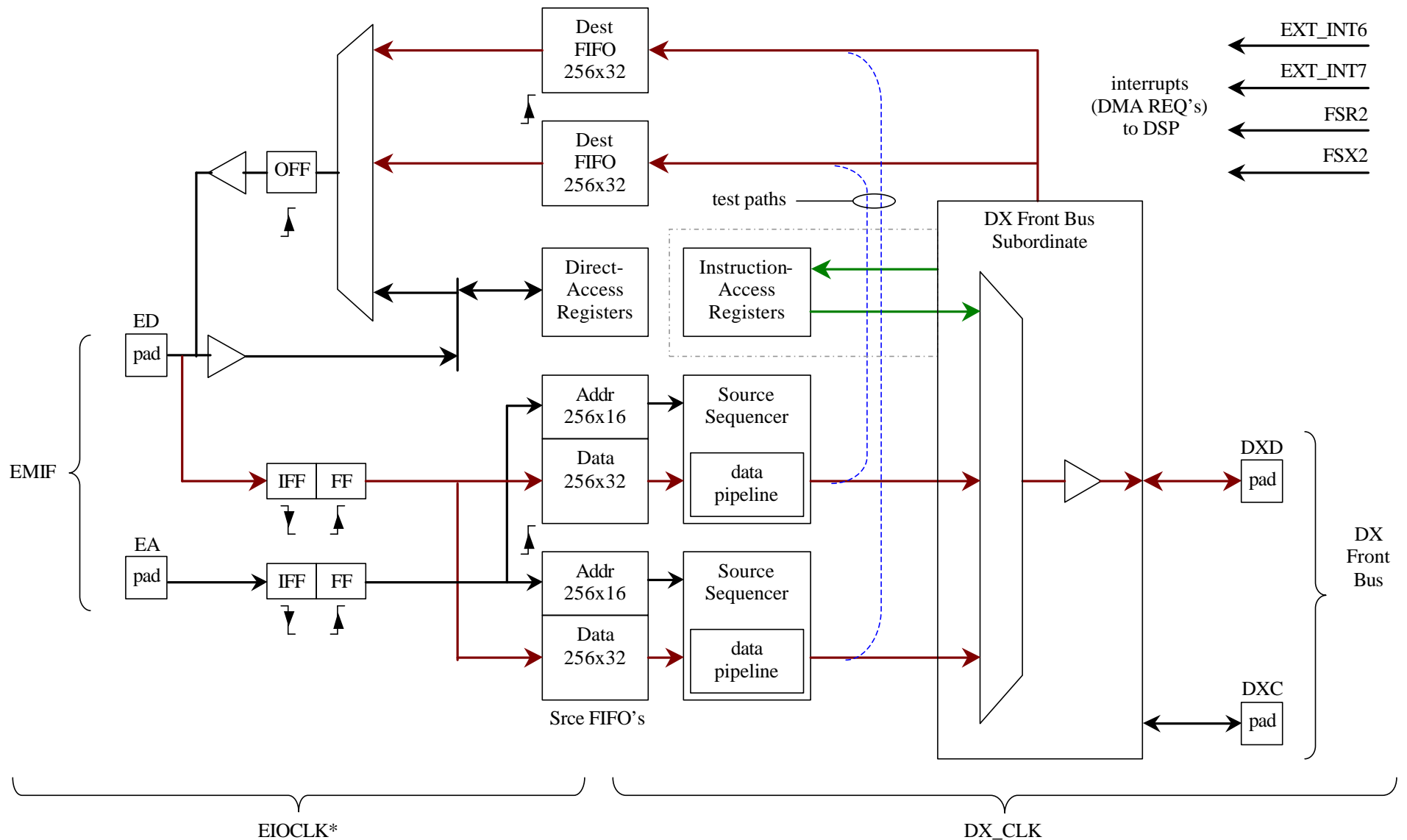
SL FPGA (on CTM) Detail



The HPU can enable continuous ROL FIFO readout to emulate a never-busy S-LINK.

DG: 57 lines available
 SG: 36 lines available
 TIS = Trigger Information Stream
 TMTS = Transition Module Trace Stream

DXD (DPU EMIF) FPGA Detail

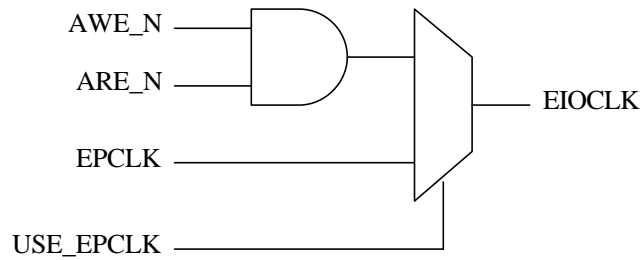


Not shown: All I/O except DPU input is registered.

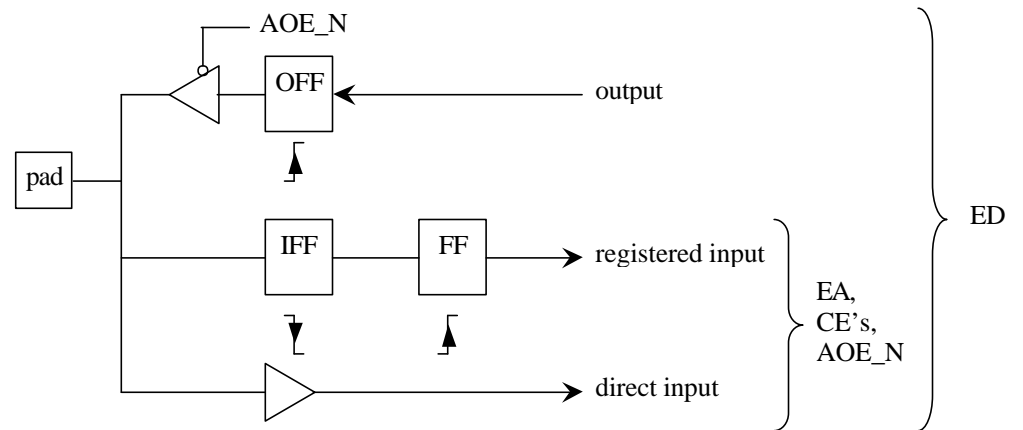
*Actual clocking is more complex, see next page.

DPU EMIF Bus Clocking

EMIF I/O Clock



EMIF I/O Connections



Sometimes additional EIOCLK pulses are needed to move data through logic clocked by EIOCLK.

The following algorithm guarantees glitch-free clocking. It is implemented in a state machine clocked by the free-running clock DX_CLK.

clear ARDY	force EMIF to wait after asserting ARE_N or AWE_N
wait two clocks	wait so that, by the time EIOCLK is forced low, any ARE_N or AWE_N pulse will be held active by ARDY
assert USE_EPCLK	EPCLK is initially low, EIOCLK may already be low
assert FORCE_EPCLK_HI	create initial rising edge
assert FORCE_EPCLK_DX for N clocks	create N more rising edges using DX_CLK
wait one clock	avoid runt/glitch
release USE_EPCLK	go back to using EMIF strobes, this results in a falling edge if a bus cycle is in progress
release FORCE_EPCLK_HI, and set ARDY	return to initial state, but only after EPCLK is no longer factored into EIOCLK resume normal EMIF operation

DPU EMIF FIFO Behavior

General Properties

	Source FIFO's	Destination FIFO's
EMIF Direction	DSP → Source FIFO	DSP ← Destination FIFO
Data Structure	blocks	no structure other than fixed frame size
DMA Frame Size	16 or few words per frame	user selectable—see DXF direct-access register: DXD_DSP_Control
Special addresses	Discard 2—word written to FIFO but discarded at FIFO output Discard 1, 0—no word written to FIFO	none—address LSB's are ignored

The DPU EMIF FPGA requests DMA on a per-frame basis as FIFO occupancy permits.

Discard 2 allows blocks with zero user words.

Discard 1 and Discard 0 simplify the hardware by pushing the previous word (Discard2) from the FPGA's input flip-flops and input pipeline to the source FIFO.

The discard addresses were chosen for convenient DMA. A block consists of zero or more user data words followed by Discard 2, Discard 1 and Discard 0:

data	word address
...	
user word	4 (typically—any address other than discard addresses is ok)
user word	3 (typically—any address other than discard addresses is ok)
ignored	2 (Discard 2)
ignored	1 (Discard 1)
ignored	0 (Discard 0)

Blocks that are longer than the maximum allowed DMA size must be divided into multiple DMA frames. The Discard words should be in the same frame. This will naturally be the case, because the shortest frame of a multiple-frame DMA transfer is always the first frame.

DPU EMIF FPGA FIFO Priorities

Yielding (Veto) Conditions

Condition	Comment
DEST FIFO empty	no more INT's for this FIFO
SRCE FIFO full	no more INT's for this FIFO
SRCE FIFO disabled	no INT's for this FIFO (dynamically controllable by DX(?))

ED is primed if it is not already primed, if a complete frame is present in either DEST FIFO, and if the EMIF has been idle for > 1 us. It will also be primed as a side effect of EMIF activity. In the table below, all DEST conditions (except DX needs DEST) assume that ED has been primed.

Priority Conditions: Left to Right → Highest to Lowest Priority *1

Condition	DX needs SRCE	DX needs DEST	SRCE INCMPLT	DEST pres	SRCE LHF
DX needs SRCE		not possible			
DX needs DEST	not possible				
SRCE INCMPLT	SRCE is empty	inhibit		inhibit after SRCE is HF	
DEST pres	inhibit *2	DEST is AF	inhibit until SRCE is HF *2		
SRCE LHF	SRCE is empty	inhibit	SRCE is INCMPLT	inhibit unless SRCE is AE	
other SRCE INCMPLT	inhibit *2	n/a	inhibit until SRCE is HF *2	n/a	
other SRCE LHF	inhibit *2	n/a	inhibit until SRCE is HF *2	n/a	ok
other DEST pres	n/a	inhibit *3	n/a	inhibit *4	n/a

Except for shaded cells, the text in a cell controls the interrupt source associated with the condition in the left column.

*1. All interrupts are inhibited while an EP (extra EIOCLK pulse) request is pending. This prevents collision of SRCE/DEST writes/reads and EP clocks.

*2. Inhibit unless EMIF is idle for 1 us. This should not occur for SRCE INCMPLT, because the DMA controller typically transfers an entire block autonomously.

*3. DEST interrupts are issued only after the first word of the corresponding frame has been loaded into the ED output flip-flops.

*4. DEST FIFO's: continue with current DEST FIFO until empty unless it is not AF and other DEST FIFO is AF

Abbreviation	Meaning/Comment
DX needs SRCE	DX is stalled because the SRCE FIFO is empty.
DX needs DEST	DX is stalled because the DEST FIFO is full.
SRCE INCMPLT	A SRCE FIFO has been sent some but not all of a block. The remainder of the block should be immediately available, because the DMA controller typically transfers an entire block autonomously.
DEST pres	A frame is present in the DEST FIFO and the ED output flip-flops are primed with DEST data.
SRCE LHF	A SRCE FIFO is less than half full and contains no incomplete blocks. See SRCE INCMPLT.
AE, AF, HF	Almost Empty (e.g., less than ¼ full), Almost Full (cannot take much more input), Half Full.
other	The other FIFO, i.e., not the one referred to in the column heading.

DX Instruction Summary

DX Instruction Format (HPU to DXF FPGA's) :

Name	Front-Executed Instructions:	Ni	Nr	description
DX_RunSequence	0010 R00K BA00 0000 DFdd dddd 0000 QQQQ	1	F*Σ _{sw}	run front sequence *2
DX_WriteData data, data, ...	0100 000K BA00 0000 DFdd dddd cccc cccc	1+c	F*c	write c data/control words to destination *1, *2, *3
DX_WriteReg value	0110 0000 BA00 0000 0Fdd dddd 00ii iiiii	2	0	write one word to register i
DX_ReadReg	1000 0HLK BA00 0000 0Fss ssss 00ii iiiii	1	F+Σ _{s1}	read one word from register i to DXF Output FIFO—F indicates DXF is a targetted source: HLK are independent of F *2
DXF Back-Executed Instructions:				
DX_SetHostFIFO	0000 0001 0000 0000 0000 0000 0000 0SVV	1	0	set Host FIFO snoop mode and Host FIFO control signals, VV=MRS_N:LD_N
DX_SetFirstSide	0000 0010 0000 0000 0000 0000 0000 000V	1	0	set back-end output order for DX_RunSequence with BA = 11: V = 0 → A then B, else B then A
DXB Back-Executed Instructions:				
DX_ReadRegDXB	1010 00LK 0000 0000 0000 0000 0000 iiiii	1	1	read one word from DXB register i to DX Trace FIFO and/or ROL FIFO, depending on trace mode and L
DX_WriteRegDXB value	0000 0011 0000 0000 0000 0000 0000 iiiii	2	0	write one word to DXB register i
Instructions executed at more than one location:				
DX_SetBackDest	0000 0100 0000 0000 0000 0000 0000 00HL	1	0	set back-end destination(s) for DX_RunSequence and DX_WriteData—applicable only when instruction's F bit is set
DX_SetTraceMode	0000 0101 0000 0000 0000 TTTT TTTT TTTT	1	0	set trace mode
DX_TraceNext	0000 0110 0000 0000 0000 TTTT TTTT TTTT	1	0	override current trace mode for next instruction
Other Instructions:				
DX_Nop	0000 0000 0000 0000 UUUU UUUU UUUU UUUU	1	0	no operation, U is any user value
Trace FIFO Special Fields:				
	NM nnnn			see next page

*1. DX_WriteData transports data via the instruction stream. The hardware does not interpret the data words, i.e., it does not consider the first word to be a header word.

*2. Only the ROL differentiates data and control words.

*3. When DX_WriteData has F set and at least one bit of BA set, both sides write the parameter words to their output FIFO's, but only side A outputs the parameter words to the back end.

DX Instruction Summary (continued)

Name	Trace FIFO Special Fields/Instructions:	Ni	Nr	description
	nnnn			n—number of words of data/count that follow **
	M			M—flag indicating more frames for this instruction follow
	N			N—flag indicating that count of data rather than data follows
DX_Pad	1110 0000 0000 0000 0000 0000 0000 0000	1	0	pad—this and all remaining words in frame may be ignored
DX_TraceTimeout	1110 0001 0000 0000 0000 0000 0000 0000	1	0	timeout—input to Trace FIFO once it has an input timeout
DX_Fault	1110 1111 0000 0000 0000 0000 0000 0000	1	0	

** n is for the current frame and current instruction only. If the instruction's M flag is set, then the instruction continues in the next frame: the instruction word will be repeated (with a new value of n) and will be followed by additional data/parameter words. If the DX Trace FIFO times out, then a frame may begin with a DX_TraceTimeout instruction even though the M bit was set in the previous frame.

Former instructions, now to be handled by register writes:

Name	Front-Executed Instructions:	Ni	Nr	
DX_VerifyFront	0000 0000 BA00 0000 VVVV VVVV VVVV VVVV	1	0	verify front counter LSB's equal V, flag verification error
DX_VerifySequence	0000 0000 BA00 0000 VVVV VVVV VVVV VVVV	1	0	verify last front sequence counter LSB's equal V, count verification error
DX_SetJamValueN jam value	0000 0J00 BA00 0000 nnnn nnnn nnnn nnnn	2	0	set jam value and N
DX_SetJamN	0000 0J00 BA00 0000 nnnn nnnn nnnn nnnn	1	0	set jam N
	0000 0000 BA00 0000 0000 0000 0000 0000	1	0	notify front (send pulse to HPU when executed at front)
	0000 000b BA00 0000 00dd dddd 00ss ssss	1	0	force down (b = force entire bus down)
	0000 0000 BA00 0000 0000 0000 0000 0000	1	0	reset front counter (or just allow write to front counter???)
	0000 0000 BA00 0000 0000 0000 0000 0000	1	0	reset bus—holds all DPU FPGA's on DX bus in reset
	0000 0000 BA00 0000 0000 0000 0000 0000	1	0	unreset bus—releases bus from reset

Name	DXB Back-Executed Instructions:			
DX_VerifyBack	0000 0000 0100 0000 VVVV VVVV VVVV VVVV	1	0	verify that back counter LSB's equal V
	0000 0000 0100 0000 0000 0000 0000 0000	1	0	notify back (send pulse to HPU when executed at back)
	0000 0000 0100 0000 0000 0000 0000 0000	1	0	reset Trace FIFO
	0000 0000 0100 0000 0000 0000 0000 0000	1	0	set reset ROL (?)
	0000 0000 0100 0000 0000 0000 0000 0000	1	0	reset ROL information registers
	0000 0000 0100 0000 0000 0000 0000 0000	1	0	update ROL information registers

DX Instruction Summary Notes

- BA = target of instruction:
- | BA | targetted DX side(s) |
|----|----------------------|
| 01 | A only |
| 10 | B only |
| 11 | A and B |
| 00 | neither |
- c = count of data/control words that immediately follow the instruction word
- F = DXF is a target:
DXF Output FIFO is target (destination) for DX_RunSequence, DX_WriteData
DXF register is target for DX_WriteReg, DX_ReadReg
- D = channel (FIFO) for destination—applicable only to DPU's (the DXF FPGA's only destination is the Output FIFO)
- d = destination target(s)
- s = source target(s)
- i = register index
- F*Σsw = 0 if DXF Output FIFO is not a destination, else the total number of words read from all sources
- F*c = 0 if DXF Output FIFO is not a destination, else the number of data words from the HPU
- F+Σs1 = the number of sources (one word per source)
- K = control bit, sent along with data to back end—typically used with S-Link to mark event start and end
- HL = destination FIFO enables (Host FIFO : ROL FIFO)—applicable for DX_RunSequence and DX_WriteData if F bit is set
- Ni = number of words in the instruction
- Nr = number of words read by the instruction – see Nr note
- Q = sequence number—index in sequence register file
- R = header word is removed at destination (mechanism not currently implemented)
- S = 1 enables Host FIFO snoop mode, in which each word transferred from DXF's to DXB is also written to the Host FIFO
- U = user value, has no effect on hardware
- V = value to compare against register LSB's
- T = trace control bits—see details below

Nr note: Nr changes depending on the trace mode

The **sequence register file** contains:

- | | |
|-----------|-----------------------------------------------------------------------------------|
| sources | up to 8 sources, one source per nybble, LS nybble is the first source – see below |
| jam value | any 32-bit value |
| jam N | 0-255 |

- source nybbles have format: Sddd
- | | |
|-----|-------------------------|
| S | = source channel (FIFO) |
| ddd | = source DPU # (0-5) |

If a sequence has fewer than 8 sources, the unused MS nybbles must be set to 0xf.

DX Front Bus Control Signals

master	command	DXC[9:7]	DXD	
DXF	write_inst_first	001	instruction	write first instruction stream word to all
DXF	write_inst	010	instruction	write other instruction word
any	nop	111	undefined	no operation
DXD	nop_end	100	undefined	no operation + end-of-source indicator *
DXD	nop_last	000	undefined	no operation + last cycle indicator
DXD	write	011	data	write data from source to destination
DXD	write_end	101	data	write + end-of-source indicator (last command issued 3 cycles later)
DXD	write_last	110	data	write + last cycle indicator

* **nop_end** is also used to mark the first word of an instruction that is not targetting this DXF bus (a silent **write_inst_first**). With this information the subordinate can keep a complete count of instructions in its InstructionCount register.

The DXC bit values are chosen such that an idle DXF bus followed by a DX_WriteReg instruction exercises all 3 bits (causes **dxc_nop**, **dxc_write_inst_first**, **dxc_write_inst**). The subordinates contain a lock mechanism that prohibits the subordinate from driving DXD or DXC unless this sequence is initially seen. See “Subordinate Unlock Mechanism”, below.

To detect missing/down DPU’s during DX_RunSequence:

the first command for a source must be one of:	nop , nop_end	write , write_end
the last command for a source must be one of:	nop_last	write_last
write_inst_first must be followed by two cycles of	nop_last	

Nop’s are used when:

- there is nothing to do
- the destination is busy
- instructions that subordinate devices are to ignore are written to the DXF bus
- DXF reads register of subordinate device

DXC[5:0] = ssssss = serial status from each DXF device. Each s bit has serial form:

01FBdd:

- F = summary of self-detected faults, e.g., destination FIFO overrun
- B = DXF bus fault (bad unlock sequence or DXC/DXD readback error)
- dd = destination FIFO AF flags (for destinations 1:0)

All DXF devices must have the same value for timeout registers → all devices detect timeout simultaneously.

All devices contain two source down lists (one per source FIFO) and two destination down lists (one per destination FIFO). Each list contains one bit per DPU. The DXF FPGA never times out and never goes down. DPU’s with corrupt serial status are added to the down list as soon as the bad serial protocol is detected.

Additional DXF FPGA serial status code:

11111 Synchronous reset. First zero bit marks end of reset.

Down destinations are ignored.

EMIF FPGA’s whose own destination down flag is set inhibit all further activity by their destination FIFO’s.

A destination is added to the down list if it times out while a destination of the current sequence.

A source is added to the down list if it times out while it is the current source.

The source timeout interval is always greater than the destination timeout interval such that a busy destination will not cause a source to timeout.

When the current source is down, all DXF devices do the following:

- set both AF flags
- push jam values into the destination FIFO (if any) that is a current destination
- return both AF flags to normal operation
- wait until no destination of the current sequence is AF
- advance to the next source in the sequence

When a device is both source and destination, it uses data from its output tracking register rather than data from the DXF bus.

When a front sequence is traced with data counting and the DXF FPGA is not a destination, the counting is performed before input to the DXF Output FIFO. I.e., only the count is written to the DXF Output FIFO, not the data words. This minimizes the effects of the trace on DX activity.

Subordinate Unlock Mechanism

Following reset or DX_RESET_N, subordinates are prohibited from driving DXD and DXC. Logic in each subordinate checks that DXC values follow the correct startup sequence. If the correct sequence is seen, subordinate output is enabled (once the ENABLE_DRIVERS bit is set via a DX_WriteReg instruction). If an incorrect sequence is seen, the unlock mechanism enters a fault state until the next reset or DX_RESET_N. The mechanism is intended to prevent subordinates from corrupting DXF bus activity, e.g., if DXC lines are shorted or open.

To unlock subordinates, HPU software must:

1. configure/reset all DX FPGA's (non-configured DXD FPGA's are legal)
2. (the DXF FPGA's instruction FIFO must be empty)
3. clear then set the DX_RESET_N bit in the DXF direct-access control register (DXFR_Control)
4. send a DX_WriteRegister instruction (ok if no targets)
5. wait while the unlock mechanism enables subordinate output, e.g., send a few DX_Nop or DX_WriteReg instructions before reading from subordinates

Typically, step 4 is used to set the CONTROL register, including the ENABLE_DRIVERS bit.

Typically, steps 4 and 5 are used to initialize all instruction-access registers and register files that must be identical for all DXF/DXD FPGA's.

DX Front Bus Protocol: DX_RunSequence

Master	SEQS	DXC	DXD	Comment
				this sequence assumes flags always show destinations ready
DXF	first	write_inst	iword	write sequence instruction to all
DXF	alast	nop	-	
DXF	last	nop_last	-	nop_last is required in last cycle
S1	first	nop_end	-	source nops when fewer than three words in block
S1	alast	write	D ₁	
S1	last	write_last	D ₁	write_last (or nop_last) is required in last cycle
S2	first	write	D ₂	
S2	other	write_end	D ₂	
S2	alast	write	D ₂	
S2	last	write_last	D ₂	
S3	first	write	D ₃	
S3	other	write	D ₃	
S3	other	write	D ₃	
S3	other	write	D ₃	
S3	other	write	D ₃	
S3	other	write	D ₃	
S3	other	write	D ₃	
S3	other	write_end	D ₃	
S3	alast	write	D ₃	
S3	last	write_last	D ₃	
S4	first	nop	-	S4 nops because its own source FIFO is not ready
S4	other	nop	-	
S4	other	write_end	D ₄	
S4	alast	write	D ₄	
S4	last	write_last	D ₄	
DXF	first	write_inst	iword	e.g., write DX_WriteData instruction to all
DXF	first	write_inst	iword	
DXF	first	write_inst	iword	
DXF	first	nop	-	nop if a destination is almost full

Transfer strategy:

1. The DXF FPGA sends the instruction word as soon as it has it. The instr_write is followed by two nops to allow for pipeline delay and to set the DXC lines to nop in case of non-present DPU. Keepers on the DXC lines hold the nop value.
2. Only DPU's can be sources for a DX_RunSequence.
3. A source issues write/write_end/write_last only if it knows what commands it must issue during the following two clock cycles. Otherwise it nops, e.g., until all words to be transferred are in its output pipeline.
4. The last command issued by a source must have the _last suffix (write_last or nop_last). The first command issued by a source must not have the _last suffix (write, write_end, nop, nop_end). This allows detection of missing sources, because keepers hold DXC at its current value when no source is driving the bus.

DX Internal Bus Data and Control Signals

Signal Logical Name	Signal Net Name	Driver	Receiver
HF_PAF_N	HFIFO_PAF_N	Host FIFO	DXF FPGA's
HF_MRS_N	HFIFO_MRS_N	DXF FPGA A	Host FIFO
HF_LD_N	HFIFO_LD_N	DXF FPGA A	Host FIFO
HF_WEN_N	HFIFO_WEN_N	current driver	Host FIFO
D	INTER_DX[31:0]	current driver	DXB FPGA and Host FIFO *1
WR_N	INTER_DX[32]	current driver	DXB FPGA *1
LAST_N	INTER_DX[33]	current driver	DXB FPGA and DXF FPGA's
FIRST_N (optional)	INTER_DX[34] *2	DXF FPGA	DXB FPGA *1
DXB_AF	INTER_DX[35] *2	DXB FPGA	DXF FPGA's
A_ICOUNT_N	INTER_DXF[0]	DXF FPGA A	DXF FPGA B
B_ICOUNT_N	INTER_DXF[1]	DXF FPGA B	DXF FPGA A
A_DONE_N*3	INTER_DXF[2]	DXF FPGA A	DXF FPGA B
B_DONE_N*3	INTER_DXF[3]	DXF FPGA B	DXF FPGA A
A_IFIFO_READY *4	HDX_DXF_HG[0]	DXF FPGA A	DXF FPGA B and HPU EMIF FPGA (EGEN)
B_IFIFO_READY *4	HDX_DXF_HG[1]	DXF FPGA B	DXF FPGA A and HPU EMIF FPGA (EGEN)
unused	HDX_DXF_HG[2]		
SYNC_HFIFO_AF	HDX_DXF_HG[3]	DXF FPGA A	DXF FPGA B and HPU EMIF FPGA (EGEN)

*1. The DXF FPGA's may receive these signals solely for the purpose of detecting faults.

*2. INTER_DX[35:34] are point-to-point connections from the DXB FPGA to each DXF FPGA. The actual net names are INTER_DXA[35:34] and INTER_DXB[35:34].

*3. DONE is used only during DX_RunSequence. It indicates that the side's final output will be in two clock cycles.

*4. IFIFO_READY: 1 → ok to DMA at least one frame to Instruction FIFO

Lines to/from HPU EMIF FPGA (EGEN's):

DXF_HG[3:0] // bus to both DXF FPGA's, pullups are installed for DXF_HG[1:0] only

DXB_HG[3:0] // point-to-point lines to DXB FPGA, no pullups

DXF Output FIFO

The DXF Output FIFO contains 32 bits of data and 8 bits of tag.

The tag contains: Lrrr FHTT

L = last word associated with an instruction

F = word is instruction for fixed-length DX_RunSequence

H = word is to be written to the Host FIFO

TT = tag type:

0 = ttInstruction // first word of instruction

1 = ttParameter // parameter word, including parameters of DX_WriteData

2 = ttCount // count of data words from DPU's (under some trace conditions, DX_RunSequence only)

3 = ttData // data word (DX_RunSequence or DX_ReadReg)

DX Internal Bus Protocol

Driver	DONE	WR	FIRST	LAST	D	Comment
A_DXF		nop			x	
A_DXF		nop			x	
A_DXF		write	1	1	I _A	instruction word, DX_Nop
A_DXF		write	1		I _A	instruction word, DX_WriteReg
A_DXF		write		1	P _A	parameter word
A_DXF	1	write	1		I _A	instruction word, DX_RunSequence
A_DXF		write			D _A	
A_DXF		write		1	D _A	last data word from side A
B_DXF	1	write	1		I _B	instruction word, same as side A instruction word
B_DXF		write		1	D _B	last data word for instruction
B_DXF		nop			x	
A_DXF						instruction ok here

Transfer strategy:

- Only side A drives Host FIFO control signals MRS_N and LD_N.
- Both sides implement the same algorithm for output to the DX internal bus, but each side drives the bus only when the algorithm indicates that it is time for it to do so.
- Output is stalled if either the DXB Input FIFO or the Host FIFO is almost full.
- Each DXF FPGA asserts its inter-DXF ICOUNT_N line under either of the following conditions:
 - an instruction word and all or several data words arrive in the output pipeline (DX_RunSequence only)
 - a complete instruction arrives in the output pipeline (all other instructions)
- ICOUNT_N indicates the side's ability to begin output (if any) for an instruction.
- Except in the case of DX_RunSequence:
 - only side A outputs the instruction word and parameter words
 - the number of words of output for each instruction is determined solely by the instruction word
 - output for an instruction begins only after all of its words are present in both sides's Output Pipeline
 - LAST is activated once per instruction
- In the case of DX_RunSequence:
 - each targetted side outputs the instruction word and its data words
 - each targetted side asserts the DONE line two cycles before its final DX internal bus cycle
 - each targetted side asserts the LAST line when its writes its last word (instruction or data)
 - the rules above do not apply for some combinations of F bit and trace mode in which instruction length is fixed
- When DX_RunSequence targets both sides, the DXB FPGA ignores the first LAST and the second instruction word.
- The DXB FPGA uses the instruction stream to determine:
 - the type of most words, i.e., data, parameter, data count
 - the ROL control bit (K)
 - trace behavior

DX_RunSequence Cases:

transfer order	A has 0 words	B has 0 words	DX internal bus activity
none	always	always	A outputs instruction word only.
A only	sometimes	always	A outputs instruction and any data.
B only	always	sometimes	B outputs instruction and any data.
A then B or B then A	sometimes	sometimes	Each side outputs the instruction word and data, if any. LAST is activated once per side.

The table illustrates why the DX_RunSequence protocol must be different from that of other instructions.

DX Trace

The DX trace mechanism can be used for debugging or for monitoring DX operation, e.g., reading out registers via the DX instruction stream. The instruction stream is always sent, along with any data read, to the DXB FPGA. Depending on the trace mode, portions of the instruction and data stream may be copied to the DX Trace FIFO.

The DX_SetTraceMode instruction sets the current trace mode while the DX_TraceNext instruction overrides this mode for one instruction. The T field of these instructions is detailed below:

T = trace mode (twelve bits)—controls input to the DX Trace FIFO:

	T	T	T	T	TT	TT	TT	TT
meaning of field	N	W	R	B	mm	mm	mm	mm
source of data words					H	H	D	D
data words destined for DXF Output FIFO?					n	y	n	y

H = HPU is source of data (DX_WriteData)

D = DPU is source of data (DX_RunSequence)

n = DX_WriteData or DX_RunSequence specified that words are **not destined** for the DXF Output FIFO (F bit is zero)

y = DX_WriteData or DX_RunSequence specified that words are **destined** for the DXF Output FIFO (F bit is one)

N = trace enable for instructions whose trace is not enabled by W, R, B and mm fields

W = trace enable for register write instructions (DX_WriteReg and DX_WriteRegDXB)

R = trace enable for DX_ReadReg instruction

B = trace enable for DX_ReadRegDXB instruction

mm = trace mode for instructions with data words sourced by the HPU or DPU(s) (see H and D):

00 = trace disabled

01 = trace returns instruction word only (DX_WriteData)

01 = trace returns instruction word and a one-word count of data words (DX_RunSequence)

10 = trace returns instruction word and all parameter words (DX_WriteData)

10 = trace returns instruction word and all data words (DX_RunSequence)

11 = reserved

#	DXTS words *
1	I
2	I + parameter word
1+	I + data words (14 max)
2	I + data word
1	I
2	I + count word
1+	I + parameter words
1+	I + data words

= count of DXTS words: When # is 1+, the instruction and its data/parameter words may span multiple frames. When # is 1 or 2, the instruction and its associated data/parameter/count word (if any) will be contained in a single frame.

* Words contributed to the DX Trace Stream, excluding any words added by the DXB FPGA to format the stream into frames (e.g., padding words and duplicate instruction words). I is the instruction word. N is a count of the DXTS words. 1+ means one plus zero or more data/parameter words.

Regardless of the trace mode, the following will never be sent to the Host FIFO or the ROL FIFO:

HPU-sourced data words **not destined** for the DXF Output FIFO

DPU-sourced data words **not destined** for the DXF Output FIFO

Instruction words

Count words

Words written to registers

Words read from registers unless one of the bits in the instruction's HL field is set

To prevent overflow of the DXF Output FIFO due to trace, the DPU's always consider its AF bit regardless of whether the DXF Output FIFO is a destination.

DX Trace Timeouts and Frame Padding

	Frame Timeout	FIFO Timeout
Serious?	No.	Yes.
Purpose is...	to release an incomplete frame (padded to make it legal) for readout by HPU.	to suspend input to the DX Trace FIFO so that the DX Trace FIFO does not block data flow to the ROL FIFO.
Can be disabled?	Yes, by setting timeout interval to zero.	Yes, by setting timeout interval to zero.
Occurs when the associated timeout interval expires and...	the DX Trace FIFO contains no complete frames but one incomplete frame containing only complete instructions.	the DX Trace FIFO is almost full (≥ 12 frames).
Implications to HPU software:	Software sees legal frame with padding (hardware sometimes adds padding for other reasons).	See notes below.

DX Trace FIFO Timeout Details:

When the DX Trace FIFO times out, its current input frame is overwritten with a timeout frame (first word = `DX_TraceTimeout`, remain words undefined) and subsequent input is inhibited until the FIFO is reset. When input is inhibited, instructions flow through the DXB FPGA as though no tracing is enabled.

If HPU software encounters a `DX_TraceTimeout` instruction at the beginning of a continuation frame, it must disregard the instruction that was to be continued.

HPU software may resume Trace FIFO input by writing to the DXB's `DXBR_ResumeTraceFIFO` register—input resumes at the first word of the next instruction to be traced.

The DX Trace FIFO times out when it has remained almost full for the timeout interval. To avoid timeout, the average DX Trace FIFO readout rate must be greater than the average data rate of the trace stream, and the timeout interval must be long enough to cover the longest period of above average stream rate. E.g., assuming an average DX Trace FIFO readout rate of ~ 5 MW/s, the timeout interval must be set to 1 ms to accommodate an isolated burst of ~ 5000 trace stream words.

Padded Frames:

Padded frames are created in either of the following cases:

1. when the last word of an instruction is written to a frame's 14th word
2. when there is a frame timeout

Case 1 was introduced to simplify FPGA logic. It can cause a minor inefficiency, because a single-word instruction might have been written to the frame's 15th word.

DX Return Stream

The DX Return Stream is the result of merging three unrelated tributary streams. The streams are merged so that a single DMA channel can efficiently transfer all three streams from the DXB FPGA to HPU internal memory. The DMA frame size is always 16 words. The three tributary streams are:

type	tributary stream	ultimate source	real-time priority
0	TIS Trigger Information Stream	CTM	high
1	TMTS Transition Module Trace Stream	CTM	low
2	DXTS DX Trace Stream	ROD DX subsystem	low

TIS contains L1 trigger information, i.e., L1ID, BCID, orbit count, etc. Its source is the CTM.

TMTS contains trace information for commands sent by the HPU to the CTM, including values read from registers on the CTM.

DXTS is the stream generated by the DX system's trace mechanism.

Each 16-word DMA frame has format:

type	1 word
data	15 words

Each of the three types of tributary streams has its own format for data words. The formats must be compatible with a stream of fixed-length packets, e.g., by providing for padding. When appropriate to do so, the hardware adds padding to partial frames and releases them for HPU readout.

Because the CTM Return FIFO conveys the time-critical Trigger Information Stream (TIS), the DX Trace FIFO is only read out when the CTM Return FIFO is empty. (This behavior depends on the values of the ..._PRIOR_LOAD fields in the DXBR_Control register.) The HPU always reads the DX Return Stream from the same address. Logic in the DXB FPGA automatically provides the HPU with data from the correct FIFO. The HPU task that manages the TIS should process the DX Return Stream as follows:

- assume that the stream typically contains mostly TIS frames

- if a frame's type is TIS, then process the frame or a portion of the frame

- if a frame's type is not TIS, then move all of the frame's data words to the appropriate buffer: TMTS buffer or DXTS buffer

- assume that other tasks will interpret the contents of the TMTS and DXTS buffers

DX Front Bus Parcel Format and Functionality

Implementation Note: The current implementation ignores all parcel content. I.e., much of the functionality on this page is not implemented.

header word rounding/checking/removal: not implemented
DMA frame size test: not implemented

Words are transferred over the DX Front Bus in parcels. When the HPU or a DPU is the source of a parcel, it must provide every word in the parcel, including any necessary padding words. Each parcel has format:

header (N)	1 word
body payload padding	zero or more words zero or more words

The header word, N, is required at the source, but the hardware can optionally remove it at the destination.
 The hardware never removes padding words.

The parcel “source size” is by definition the total number of words in the parcel as it is transmitted by the source.
 The parcel “destination size” is by definition the total number of words in the parcel after optional removal of the header word.
 When the header word is retained at the destination, destination size == source size.
 When the header word is removed at the destination, destination size == source size - 1.

The hardware allows two options for the value of N:

- a. N = source size: 1 + the number of words in the body
- b. N = source size - the number of padding words: 1 + the number of words in the payload

For option (b), the hardware will use the RoundingMask register to calculate parcel source size

“DMA frame size” refers to the size of DMA frames used by a DPU to read out its DX destination FIFO(s).

remove header?	round N?	source size	destination size	DMA frame size
no	no	N	N	unrestricted
no	yes	N rounded up	N rounded up	must be power of 2
yes	no	N	N - 1	
yes	yes	N rounded up	N rounded up - 1	must be power of 2 - 1

Rounding is performed at source and destination by using the value in the RoundingMask register:
 rounded value = (value + RoundingMask) & ~RoundingMask

Example for RoundingMask = 0xf and value = 0x33:

$$0x40 = (0x33+0xf) \& 0xfffffff0$$

The value in the RoundingMask register must always be a power of two minus 1.

N is restricted to 1-65535 in order to enforce a practical limit on transfer lengths.

A mechanism is provided to verify that each parcel’s destination size (see above) is a multiple of the DMA frame size (e.g. 16 words). This rule is enforced at each destination. Enforcement of the rule can be disabled on a per-destination basis.

The DX_WriteData instruction transports data via the instruction stream. The hardware does not interpret the data words, i.e., it does not consider the first word to be a header word.

Implementation Note: The current implementation ignores all parcel content. I.e., much of the functionality on this page is not implemented. See implementation table at the top of this page.

DXF and DXD Instruction-Access Registers

In DXF and DXD FPGA's	R/W	R	r		MM ?	
DXFI_Sequence	R/W	0	Q	SsssSsss SsssSsss SsssSsss SsssSsss S = source FIFO# s = source DPU# (0-5) Ssss = 1111 → end of sequence	Sequence register file. This register lists the sources for DX_RunSequence. The rightmost nybble is the first source.	yes
DXFI_JamData	R/W	1	Q	VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV	Jam data register file. This value is written to a targeted destination FIFO when a source is down. The value is written JamCount times.	no
DXFI_JamCount	R/W	2	Q	00000000 00000000 00000000 VVVVVVVV	Jam count register file. Zero is legal.	no
DXFI_Control	R/W	3	0	EDSSrrrr rrrrrrrr rrrrrrrr MMMMMMMM E = enable DXD and DXC output drivers D = enable destination FIFO interrupts to DPU S = enable source FIFO interrupts to DPU M = 1 MHz divisor, e.g., 50 for 50 MHz DX_CLK	Control register. The E, D and S bits are applicable only to DPU's. Writing this register resets the 1 MHz divider. Frequent writes will affect the DPU's 1 MHz timebase.	yes
DXFI_Timeout	R/W	3	1	DDDDDDDD DDDDDDDD IIIIIIII IIIIIIII D = timeout divisor I = timeout interval	Timeout register. A timeout occurs when the DX Front bus is stalled for D*I microseconds. E.g., if D is set to 1 and I is set to 1000, then a timeout occurs after 1 ms. For best accuracy, avoid small I.	yes
DXFI_InstructionCount	R/W	3	7	cccccccc cccccccc cccccccc cccccccc c = number of instructions seen by subordinate. Instructions are counted regardless of which side is targeted, whether the subordinate interprets them, etc. Counting stops when the subordinate's instruction sequencer faults. Other faults do not affect counting.	Instruction count register. The user may write any value to this register but typically writes zero.	no
DXFI_Status	R	3	3	ffffffff fffffffF FFFFFFFF SSDDIiii f = status from DXF/DXD-specific logic * F = status from subordinate ** S = source FIFO empty flags D = destination FIFO empty flags I = side (A = 0, B = 1) i = ID strap, e.g., 0 for DPU 0	Status register. For DXF FPGA's, i is hardwired to 6, and S and D are hardwired to 0.	n/a
DXFI_Test	R/W	3	F	see standard test register note	standard test register	no

MM = must match: a single DX_WriteReg must be used to set the same value in a side's DXF FPGA and all of its DXD FPGA's. The two sides of the ROD may have different values.

* DXD-specific status has form: rFVP rSrs DDDD dddd

- F = DSP read from wrong destination FIFO
- V = DSP read bad value from destination FIFO
- P = destination FIFO pipeline fault (indicates Verilog bug)
- S/s = source FIFO 1/0 overflow
- D/d = destination FIFO 1/0 fault: { RD_OVR, WR_OVR, FR_OVR, FR_UND }
- RD_OVR = read overflow (DSP side)
- WR_OVR = write overflow (DX side)
- FR_OVR = frame counter overflow (DX side)
- FR_UND = frame counter underflow (DSP side)

* DXF-specific status has form: Iibr SSSS BBBB BBBB

- I = IFIFO_READY—instruction FIFO can accept at least one frame of DMA
- i = IFIFO_EF—1 → instruction FIFO is empty
- S = SUPR_FAULTS—supervisor fault code (0 = no fault, else see Verilog)
- b = BACK_EF—1 → back end FIFO's and pipeline are empty
- B = BACK_FAULTS—back end faults + fault code (0 = no fault, else see Verilog)

** Status from subordinate has form: 00 BU IIII

- B = FAULT_DXF_BUS - 1 → readback error detected on DX bus lines: DXC[9:7] or DXD
- U = FAULT_UNLOCKED - 1 → unlock sequence failed
- I = FAULT_ISS - 0 → no fault, else see fiss... 4-bit fault codes in dxs_subordinate.v

DXF Instruction-Access Registers

<i>In DXF FPGA only:</i>	R/W	R r		
DXFI_DownList	R	3 b	rrSSSSSS rrssssss rrDDDDDD rrddddd	List of down sources and destinations. S/s = source 1/0 down list D/d = destination 1/0 down list
DXFI_SerialStatus	R	3 c	rrrrrrrr rrrrrrrr rBBBBBBB rFFFFFFF	Serial status register. B and F are independent. B = serial status bus fault bits (1 DXF, 6 DXD) F = serial status fault summary bits (1 DXF, 6 DXD)
			timeouts: Host FIFO	A Host FIFO timeout is not implemented.

DXD Instruction-Access Registers

<i>In DXD FPGA only:</i>	R/W	R r		
DXFI_SourceCount	R/W	3 9	cccccccc cccccccc cccccccc cccccccc	Source word count register. The user may write any value to this register but typically writes zero. It is incremented upon output of a source word to the DXF bus. c = count of words sourced by either source FIFO
DXFI_DestinationCount	R/W	3 a	cccccccc cccccccc cccccccc cccccccc	Destination word count register. The user may write any value to this register but typically writes zero. It is incremented when DX writes a word to either destination FIFO. c = count of words written to destination FIFO
DXFI_LED	R/W	3 b	rrrrrrrr MMMMMMMM rrrrrrrr EEEEEEEE	LED control register. Controls the DPU's red LED. M = minimum flash duration in ms E = LED enables *1
DXFI_Temperature	R	3 c	nnnnnnnn nnnnnnnn nnnnnnnn TTTTTTTT	Temperature + serial number MSB register. n = serial number MSB's T = DPU temperature, signed, degree C
DXFI_SerialNumber	R	3 d	nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn	Serial number LSB register. n = serial number LSB's

*1: The LED enable bits are:

- 7: DSP_LED_LEVEL, // level control from DSP (LED on while DXD_LED_Level is 1)
- 6: DSP_LED_PULSE, // pulse control from DSP (one flash per DSP access to DXD_LED_Pulse register)
- 5: DSP_LED_SRCE, // any DSP write to source FIFO's
- 4: DSP_LED_DEST, // any DSP read from destination FIFO's
- 3: DX_LED_FAULT, // DX fatal fault
- 2: DX_LED_WRITE, // DX write that targets this DPU
- 1: DX_LED_READ, // DX read that targets this DPU
- 0: ON // constant ON source → HPU can force LED on

The DPU's LED will be lit when an enable bit is set and its corresponding signal is active. It is legal to set more than one enable bit. "DSP" is the DPU's DSP. Logic in the DPU's EMIF FPGA forces the LED to remain lit for at least M ms.

DXF Direct-Access Registers (accessible to HPU)

<i>In DXF FPGA:</i>	R/W	EA[5:2]		
DXFR_DLL_Reset	W	0	Writing any value resets the DLL, which causes an asynchronous reset of the FPGA until the DLL's lock.	DLL reset location.
DXFR_InstructionFIFO	W	1	vvvvvvvv vvvvvvvv vvvvvvvv vvvvvvvv Typically, DMA is used to write to the instruction FIFO of both DXF FPGA's simultaneously. The HPU EMIF FPGA generates a DMAREQ when both FPGA's can accept a frame of instructions.	Input to instruction FIFO.
DXFR_Status	R	0	same value as DXFI_Status	Status register
DXFR_SerialStatus	R	2	same value as DXFI_SerialStatus	Serial status register.
DXFR_DownList	R	3	same value as DXFI_DownList	List of down sources and destinations.
DXFR_Control	R/W	7	rrrrrrrr rrrrrrrr rrrrrrrr rrrrrrrR R = DX_RESET_N (must be set low then high to reset)	Control register.
DXFR_Test	R/W	f	see standard test register note	standard test register

DXD Direct-Access Registers (accessible to DPU)

<i>In DXD FPGA:</i>	R/W	EA[7:4]		
DXD_LED_Level	W	0	1 --> EMIF (red) LED on	Sets the DPU's EMIF (red) on if enabled by DX.
DXD_SDCKE	W	1	1 --> SDRAM clock is enabled	
DXD_TINP0_Period	W	3	rrrrrrrr rrrrrrrU pppppppp pppppppp U = unit (0 → DX_CLK cycles, 1 → microseconds) p = TINP0 period in the specified units	The default value, 0x10001 results in a TINP0 period of 1 microsecond.
DXD_DLL_Reset	W	5	Writing any value resets the DLL, which causes an asynchronous reset of the FPGA until the DLL's lock.	DLL reset location.
DXD_DSP_Control	R/W	6	rrrrrrrr rrrrrrrr rrrrrrLL rFFFFFFF F = destination FIFO DMA frame size in words L = loopback enable for each srce/dest pair of FIFO's	
DXD_EP_IDLE_T	W	7	rrrrrrrr rrrrrrrr rrrrrrrr VVVVVVVV	The number of DX_CLK cycles EMIF must be idle before EP request is granted. The default value of 50 is appropriate for most applications.
DXD_LED_Pulse	W	8		Writing to this register pulses the DPU's EMIF (red) LED, if enabled by DX.
DXD_ID_Strap	W	9	rrrrrrrr rrrrrrrr rrrrrrrr rrrrIiii I = side (A = 0, B = 1) i = ID strap, e.g., 0 for DPU 0	DPU's ID#. Initial value of i is 7, so DPU does not react to DX activity.
DXD_Status	R	1	same value as DXFI_Status	Status register.
DXD_Temperature	R	2	same value as DXFI_Temperature	
DXD_SerialNumber	R	3	same value as DXFI_SerialNumber	

DXB Instruction-Access Registers

<i>In DXB FPGA:</i>	R/W	R r		
DXBI_Control	R/W	x 0	rrrrrrrr rrrrrrrr rrrrrrrr rrrrrrrr	Control register. Not currently implemented.
DXBI_Timeout	R/W	x 1	FFFFFFFF FFFFFFFF ffffffff ffffffff F = Trace FIFO timeout interval f = Trace frame timeout interval	Timeout register. Setting an interval to zero disables the timeout.
DXBI_Status	R	x 2	see DXB status note below	Status of DXB FPGA.
DXBI_TM_Status	R	x 3	see SL FPGA documentation	Status from TM's SL FPGA.
DXBI_ResumeTraceFIFO	W	x 2	rrrrrrrr rrrrrrrr rrrrrrrr rrrrrrrr	writing any value re-enables DX Trace FIFO input, i.e., recovers from timeout
DXBI_ReleaseTraceFrame	W	x 3	rrrrrrrr rrrrrrrr rrrrrrrr rrrrrrrr	writing any value allows the current trace frame to be released for HPU readout, effectively reducing the frame's timeout interval to zero
DXBI_InstructionCount	R/W	x 7	cccccccc cccccccc cccccccc cccccccc c = count of instructions	Instruction count register. *1
DXBI_HostFIFO_Count0 DXBI_HostFIFO_Count1 DXBI_ROL_FIFO_Count0 DXBI_ROL_FIFO_Count1	R/W	x 8 x 9 x a x b	cccccccc cccccccc cccccccc cccccccc c = count of words written to FIFO	Each FIFO has two count registers, e.g., one for per-run accounting and one for per-event counting. *1
DXFI_Test	R/W	x F	see standard test register note	standard test register

*1: HPU software can set count registers to any value via the `DX_WriteRegDXB` instruction. The count value can be read only via a `DX_ReadRegDXB`. The value read reflects all activity at the Input FIFO output up to but excluding the `DX_ReadRegDXB` instruction that returned the value.

DXB Direct-Access Registers (accessible to HPU)

<i>In DXB FPGA:</i>	R/W	EA[5:2]		
DXBR_DLL_Reset	W	0	Writing any value resets the DLL, which causes an asynchronous reset of the FPGA until the DLL's lock.	DLL reset location.
DXBR_Status	R	0	see DXB status note below	Status register.
DXBR_TM_Status	R	1	see SL FPGA documentation	Status from TM's SL FPGA.
DXBR_TM_CommandFIFO	W	8	32-bit value	Input to TM command FIFO. Input is in words.
DXBR_DX_ReturnFIFO	R	8	32-bit value	Combined output of DX Trace FIFO and TM Return FIFO. Output is in 16-word frames.
DXBR_Control	R/W	7	see DXBR_Control note	Control register.
DXBR_Test	R/W	f	see standard test register note	standard test register

See notes on next page.

DXBR_Control Note:

The format for DXBR_Control is:

31	TM_DG_OUT_PHASE	1 → delay TM DG bus synchronous outputs to DXB by 1/2 DCLK period
30	TM_AAL_LOCK	rising edge triggers TM's DG bus auto-align sequence
29	TM_AAL_MANUAL_PHASE	phase to be used when TM_AAL_MANUAL is 1
28	TM_AAL_MANUAL	1 → TM uses manual DG bus phase rather than phase determined by auto-alignment
27	TM_RESET_N	reset to TM's SL FPGA (active low)
26	TM_ENABLE	0 → ignore inputs from TM, e.g., when TM is not powered up
25:24	reserved	
23	DG_OUT_PHASE	1 → delay DXB DG bus synchronous outputs to TM by 1/2 DCLK period
22	AAL_LOCK	rising edge triggers DXB's DG bus auto-align sequence
21	AAL_MANUAL_PHASE	phase to be used when AAL_MANUAL is 1
20	AAL_MANUAL	1 → DXB uses manual DG bus phase rather than phase determined by auto-alignment
19:16	TM_PRIOR_LOAD	reload values for DX Return Stream priority counters *
15:12	DT_PRIOR_LOAD	for Transition Module Trace Stream (suggested value: 15)
11:8	TI_PRIOR_LOAD	for DX Trace Stream (suggested value: 15)
7:0	DXINT_CLK_UDIV	for Trigger Information Stream (suggested value: 1)
		set value to DXINT_CLK rate in MHz, e.g., 50 for 50 MHz DXINT_CLK

Bits 31:27 are output to the TM's SL FPGA on dedicated DG lines. They are not used by logic within the DXB FPGA.

* These values determine the relative priority of the three sources of DX Return Stream frames. Smaller values yield higher priority. See dxb.v for details.

DXB Status Note:

The format for both DXBI_Status and DXBR_Status is the same:

31	FAULT_TRFIFO_IFRAME	1 → transition module changed destination FIFO (TIS/TMTS) before frame was complete
30:29	2'b0	reserved
28	FAULT_INFIFO_OVF	1 → input FIFO overflowed
27	FAULT_ROFIFO_OVF	1 → ROL FIFO overflowed
26	FAULT_TCFIFO_OVF	1 → TM Command FIFO overflowed
25	FAULT_TIFIFO_OVF	1 → TIS FIFO overflowed
24	FAULT_TMFIFO_OVF	1 → TMTS FIFO overflowed
23:22	2'b0	reserved
21	FAULT_TIFIFO_UNF	1 → TIS FIFO underflowed
20	FAULT_TMFIFO_UNF	1 → TMTS FIFO underflowed
19	FAULT_INFIFO_FIRST_MTCH	1 → the FIRST from the two DXF FPGA's did not match
18	FAULT_INFIFO_FIRST_LAST	1 → FIRST and LAST from the DXF FPGA's were inconsistent
17	FAULT_INFIFO_INST	1 → an illegal instruction word was encountered
16	FAULT_INFIFO_INST_DUPL	1 → the instruction words from the DXF FPGA's did not match (some cases of DX_RunSequence only)
15:12	4'b0	reserved
11:10	2'b0	auto-alignment status for DG bus
9	AAL_PHASE	
8	AAL_LOCKED	
7:0	AAL_HIST	

Standard Test Register Note:

Any value may be written. The value is post-incremented after each read.

Failures Most Likely Caused by Hardware:

Failure	Fatal to DPU?	Fatal to DX?	How Detected?	Reaction
Two DPU's have same ID strap value	n/a	yes	pre-operation check of test register (deconfiguring DXD FPGA's can help identify faulty DPU)	HPU does not use DX and reports DX failure.
Shorted DXC or DXD line	n/a	yes	software pre-operation scan (before DPU FPGA's are configured?)	HPU does not use DX and reports DX failure.
Open DXC or DXD line	yes	yes	software verification of DXD FPGA test register accesses, one DPU at a time	HPU does not use DX and reports DX failure.
Data read back on DXD/DXC does not match data driven onto DXD/DXC.	yes	no*	DXD and DXF FPGA's monitor.	If detected by a DXD FPGA, the DXD FPGA goes down. If detected by the DXF FPGA, DX fails.

*unless it was the DXF FPGA that detected the failure.

Failures Most Likely Caused by Software:

Fault	Fatal to DPU	Fatal to DX?	Where Detected	Reaction
illegal register address: DX_WriteReg DX_WriteRegDXB	no	no	not detected	write is ignored
illegal register address: DX_ReadReg DX_ReadRegDXB	no	no	not detected	read returns undefined data

Critical Path Timing Tables

Generic DX (fully pipelined):

description	component	delay	
clock-to-output delay	DPU or DXF FPGA	6.2	(-6, slow, 8mA, with DLL, worst case: CLK to lo-Z)
wiring + settling delay	wire + load	10.0	
clock skew	clock driver(s) + wires + loads	1.0	ns (driver → .2 ns, 20 pF * 25 ohm → .5 ns)
input setup	DXF FPGA or DPU	1.7	IFF (-6 with DLL)
<i>total</i>		18.9	ns (minimum period for DX_CLK)

DXF FPGA Data and Control Signals Setup at Host FIFO:

description	component	delay	
clock-to-output delay	DXF FPGA	6.2	(-6, slow, 8mA, with DLL, worst case: CLK to lo-Z)
wiring + settling delay	wire + load	2.5	ns
input setup	IDT72V3690L10PF	3.5	ns
clock skew	clock driver(s) + wires + loads	0.5	ns (driver → .2 ns, 10 pF * 25 ohm → .25 ns)
<i>total</i>		12.7	ns (minimum period for DXINT_CLK)

DXF FPGA Bussed Signal on DXF_HG (to HPU), e.g. Host FIFO sync'd AF output by one DXF FPGA:

description	component	delay	
clock-to-output delay	DXF FPGA	5.8	(-6, slow, 8mA, with DLL, CLK to out)
wiring + settling delay	wire + load	4.7	ns
input setup	DXF FPGA	1.7	IFF (-6 with DLL)
clock skew	clock driver(s) + wires + loads	0.5	ns (driver → .2 ns, 10 pF * 25 ohm → .25 ns)
<i>total</i>		12.7	ns (minimum period for DXINT_CLK)