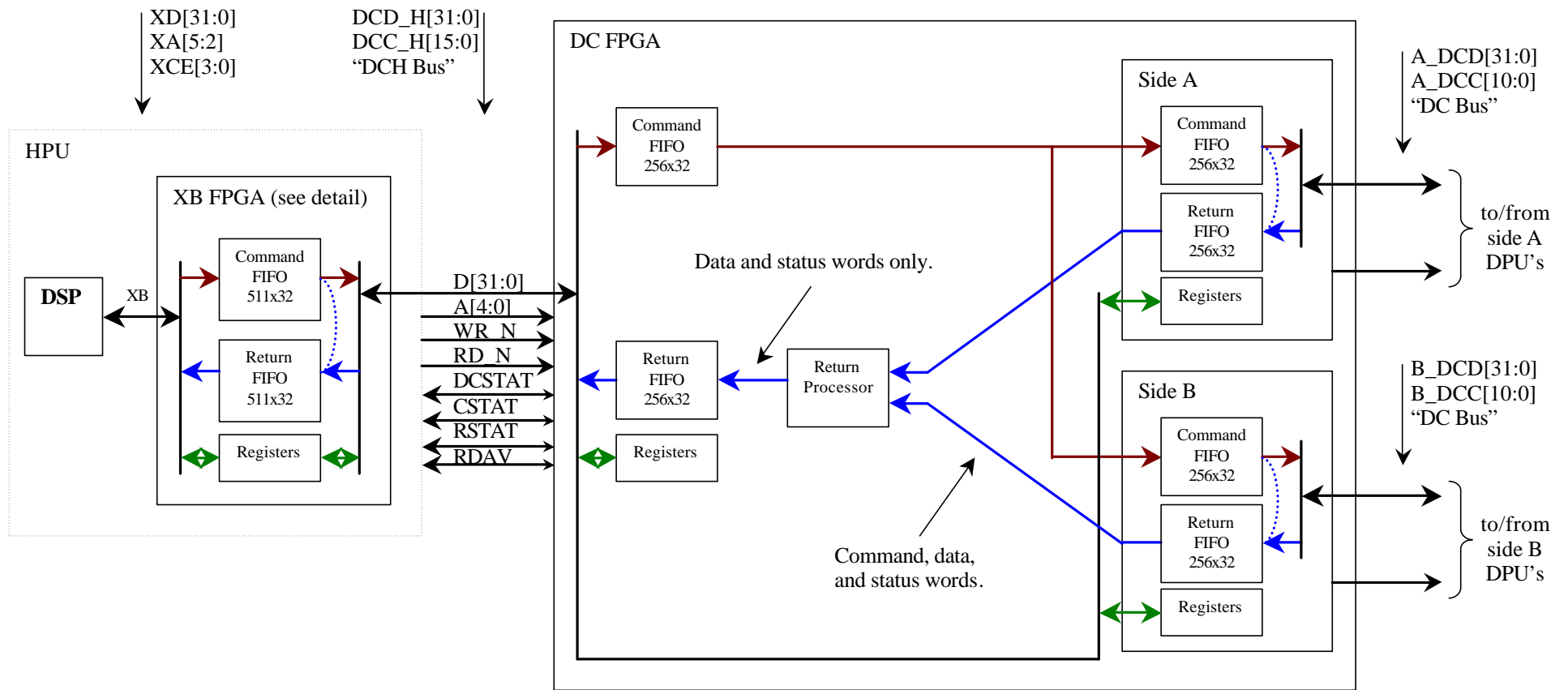



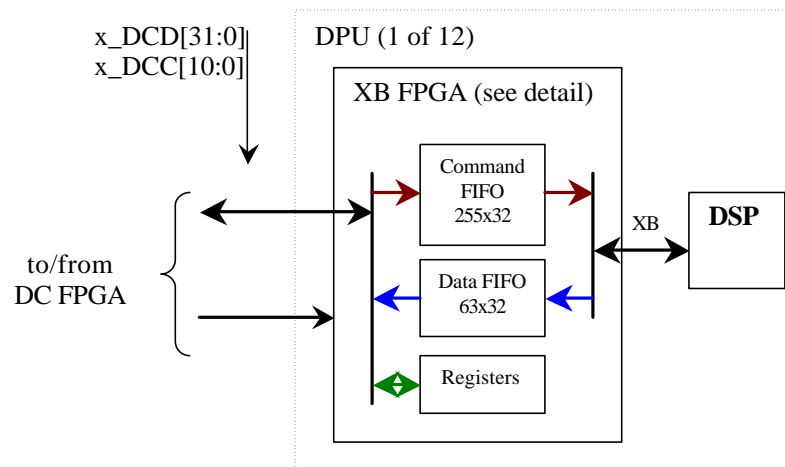
# DPU Control (DC) Overview



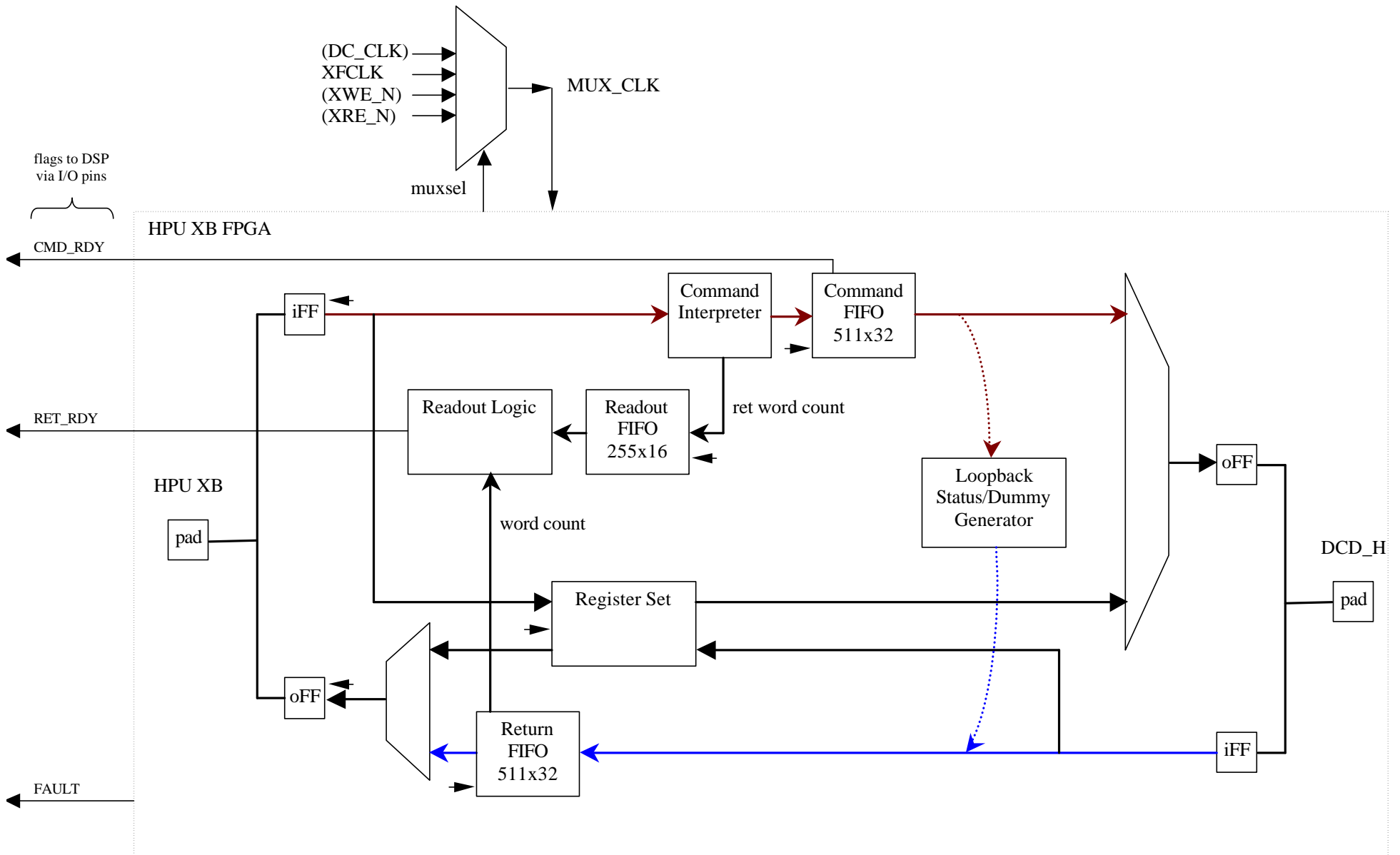
 = test path

“256x” indicates FIFO is fully synchronous to DC\_CLK.  
Other FIFO’s have asynchronous ports.

The DPU’s 63x32 Data FIFO is the minimum. See DPU XB FPGA detail.

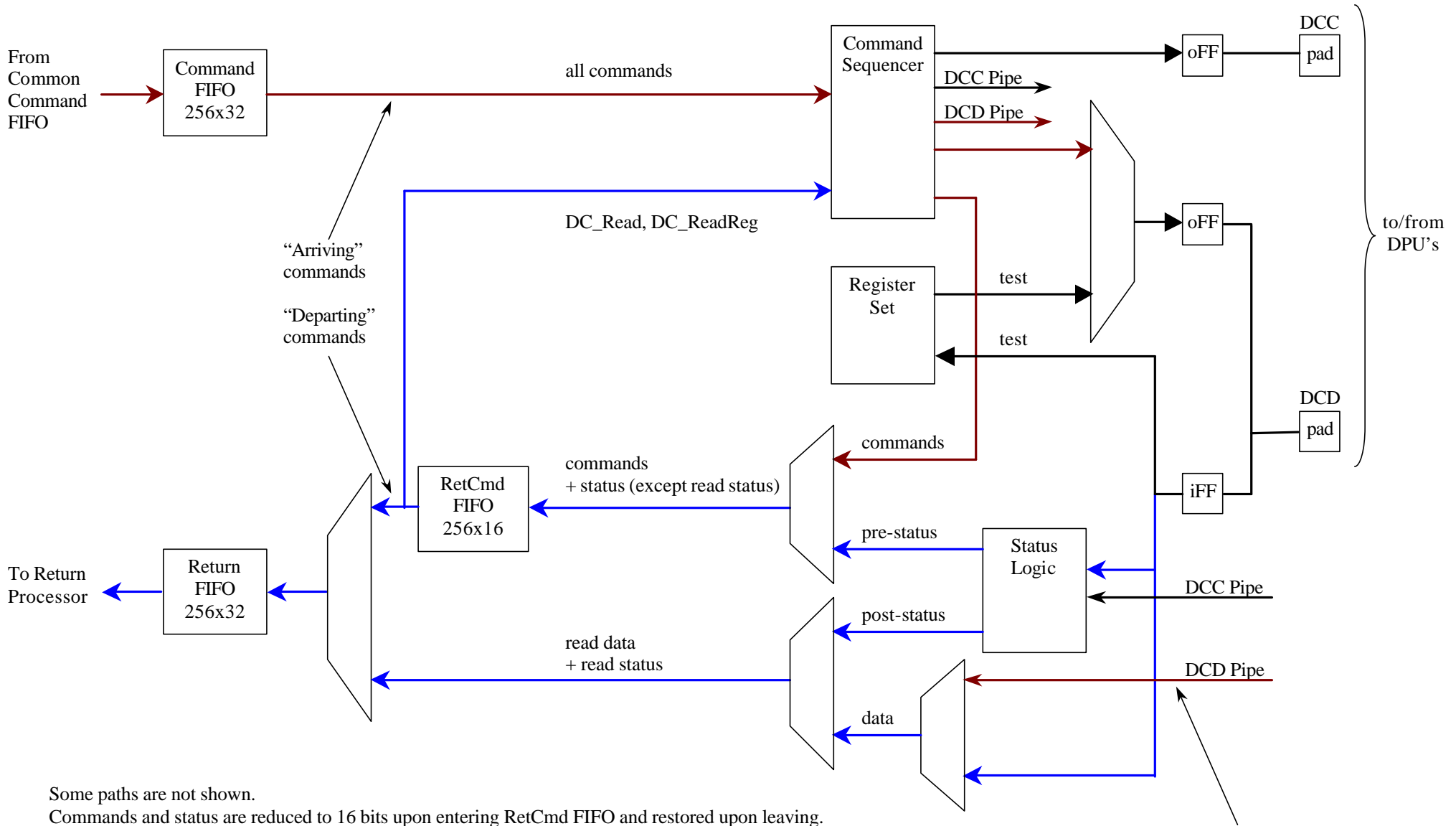


# HPU XB FPGA Detail



Logic is clocked by DC\_CLK (via DLL) unless noted otherwise.  
 ➔ indicates clocking by MUX\_CLK (via external mux and DLL)

# Side (A and B) Detail



Some paths are not shown.  
 Commands and status are reduced to 16 bits upon entering RetCmd FIFO and restored upon leaving.

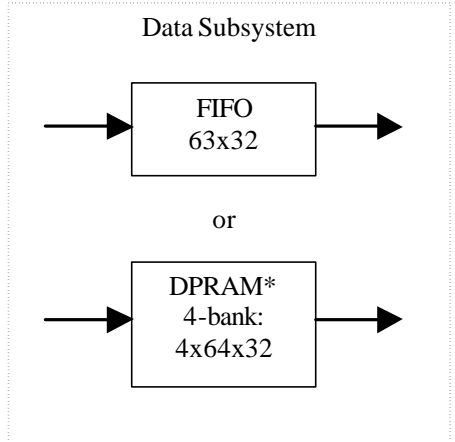
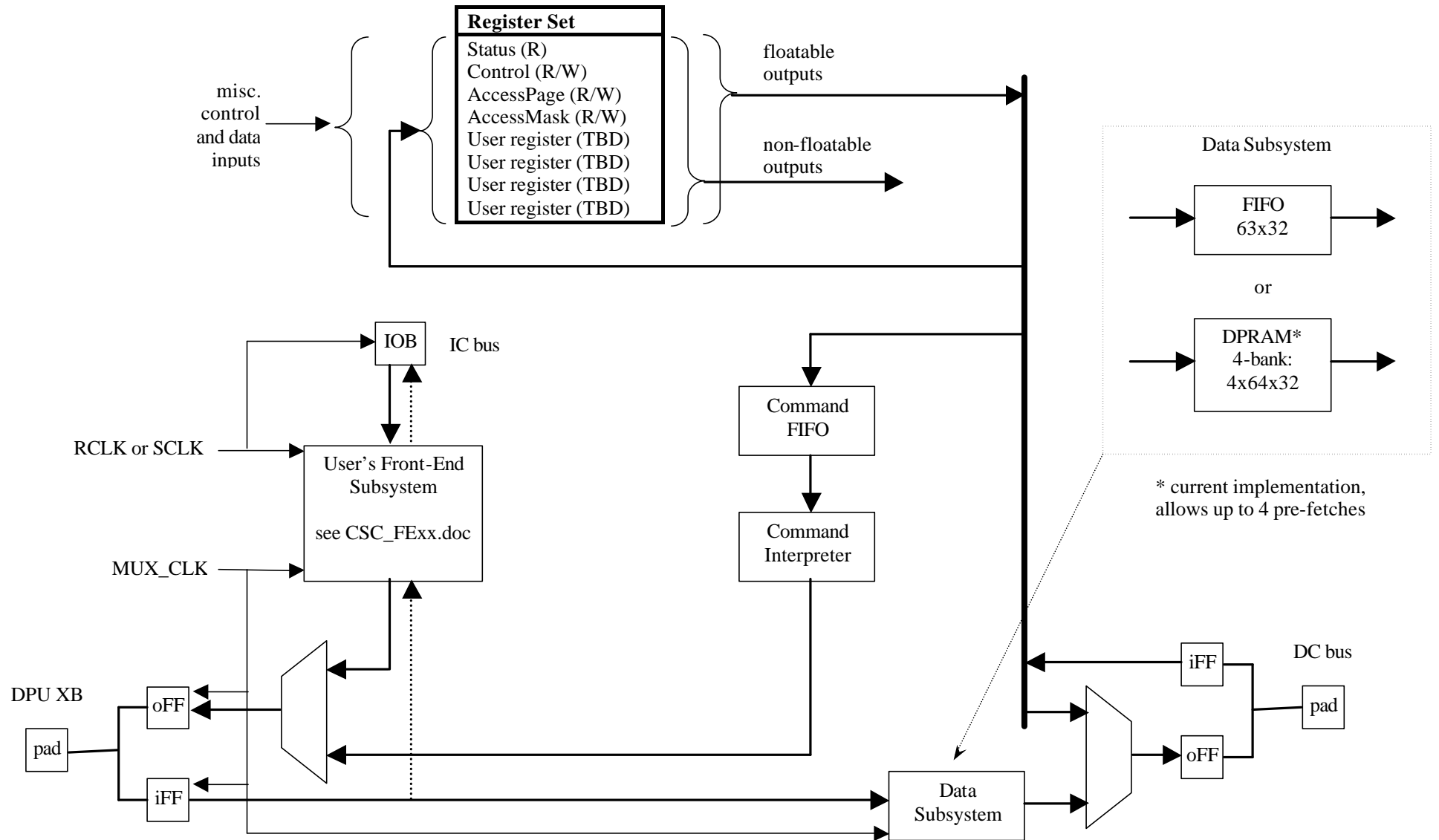
Simulated Data (loopback mode)  
 or  
 FailureValue (down DPU)

All logic is clocked by DC\_CLK (via DLL).

# DPU XB FPGA Detail

Logic is clocked by DC\_CLK (via DLL) unless noted otherwise.  
 MUX\_CLK can be either DC\_CLK or XRE\* (via external mux and no DLL).

DC\_CLK (with DLL) to muxsel out: 3.3 ns  
 max mux enable/disable time: 5.5 ns → 56.8 MHz max DC\_CLK  
 total: 8.8 ns



\* current implementation, allows up to 4 pre-fetches

# DC Registers

Register Name	R/W/S	Address	Contents	Description
<b>In HPU XB FPGA:</b>				
<b>XCE0 -- resets</b>				
DCR_DLL_RESET	W	5	rrrrrrrr rrrrrrrr rrrrrrrr rrrrrrrr	resets the DLL's, which causes a soft reset until the DLL's lock
DCR_SOFT_RESET	W	A	rrrrrrrr rrrrrrrr rrrrrrrr rrrrrrrr	resets most of the chip
<b>XCE1 -- user registers</b>				
DCR_HPU_Control	R/W	1	rrrrrrrr rrrrrrrr rrrrrrrr rrrrrRLL R = reset DC FPGA LL = loopback mode, see note 1	
DCR_HPU_Status	R	2	see note 2	status
DCR_RegWrite	R/W	3	VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV	data to write to register in DC FPGA
DCR_RegIndex	R/W	4	rrrrrrrr rrrr00RW 00000000 000iiii R = read from register in DC FPGA W = write to register in DC FPGA if both R and W are set, write is performed then read i = index of register in DC FPGA rrrrrrrr rrrr1000 000vvvvv vvvvvvvv test mode: v is written to DCC lines (DCH bus), and DC FPGA copies these lines to the DCC_TEST register	index of register in DC FPGA to be read/written or value to be verified on DCC lines  this register is zeroed after test, read and/or write completes
DCR_RegRead	R	5	VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV	data read from register in DC FPGA
DCR_LED_Control	R/W	6	rrrrrrrr UFrrrrRC rrrrrrrr DDDDDDDD U = LED on unconditionally F = LED on when fault R = LED on when Return FIFO not empty C = LED on when Command FIFO not empty D = minimum LED on time, units ~= ms	control of HPU's green LED  C and R bits are ignored if FPGA enters fault state
DCR_HPU_XB_StdTest	R/W	F	see note 3	standard test register
<b>XCE2: FIFO's</b>				
DC_FIFOS	W	r	VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV	DC command stream via synchronous FIFO interface
DC_FIFOS	R	r	VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV	DC return stream via synchronous FIFO interface

Notes:

### 1. Loopback mode in DCR\_HPU\_Control:

- 00 // normal operation (no loopback)
- 01 // direct loopback -- all values written to Command FIFO (including DC\_Pad) are looped back to Return FIFO
- 10 // like direct loopback, except Command FIFO output proceeds to the DC FPGA (which can be set up to loop it back to the HPU XB FPGA's Return FIFO)
- 11 // simulated normal operation

### 2. DCR\_HPU\_STATUS: for latest bit assignments, search for "// >>> DCR\_HPU\_STATUS" in hpu\_xb.v:

```
DCR_HPU_STATUS <= { 12'b0,
                    3'b0,
                    1'b0, RET_OVERREAD_FAULT, RET_CAPACITY_FAULT, DC_FPGA_FAULT,
                    2'b0, RET_FIFO_RD_FAULT, RET_FIFO_WR_FAULT, USER_REG_RD_FAULT,
                    2'b0, REO_FIFO_RD_FAULT, REO_FIFO_WR_FAULT, RET_FIFO_RD_FAULT,
                    1'b0, CMD_INVALID_FAULT, CMD_FIFO_RD_FAULT, CMD_FIFO_WR_FAULT
                };
```

### 3. Standard Test Register: Any value may be written to the standard test register. The value is incremented after each read of the register.

## DC Registers (continued)

<i>In DC FPGA:</i>		A[3:0]	A[4] = 1	
DCR_Control	R/W	3	rrrrrrrr AAAAPPPP TTTTTTTT TTTTTTTT A = RETCMD FIFO almost full threshold units: 16 words suggested default value: 4 → 64 words P = RETCMD FIFO DC_Read priority threshold units: DC_Read's in RETCMD FIFO suggested default value: 8 T = timeout interval in DC_CLK cycles	control register see note 4
DCR_FailureValue	R/W	4	VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV	value to return instead of data when a DPU read fails Also used for DCR_TestOut (value to drive onto DCD or DCC during test)
DCR_RP_CmdCount	R	B	VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV	value of return processor command counter – incremented when the command processor processes the command word from side A, i.e., when the command processor begins processing a command
DCR_Status	R	C	rrrrrrrI rrrr0MCS rrrrrrrB rrrrrrrA I = invalid command word at common Command FIFO output 0MCS = Return Processor faults: 0 = zero M = command mismatch between sides C = command synchronization S = status synchronization B = Side B severe fault A = Side A severe fault	status register  synchronization faults occur when a command or status word was not properly formatted  DCR_SideStatus contains details
DCR_DCC_Test	R	D	rrrrrrrr rrrrrrrr rrrVVVVV VVVVVVVV V = value of DCC lines (DCH bus)	value captured on DCC lines as a result of a DCC test (see DCR_RegIndex)
DC_StdTest	R/W	F	see note 3	standard test register
<b>for each side:</b>		<b>A[2:0]</b>	<b>A[4] = 0, A[3] = side</b>	
DCR_SideControl	R/W	0	MMMMRRRR rrrrrrrr rrDDDDDD rrTTTTTT M = mode (see DC Control Modes table below) R = register for some tests D = user-specified down DPU's T = target DPU(s) for some tests, three LSB's are TTT	control register
DCR_TestIn	R	3	VVVVVVVV VVVVVVVV VVVVVVVV VVVVVVVV	value read from DCD or DCC during test
DCR_SideStatus	R/S	4	rrrrrrrr rrrrrrrr rrrrrrbB rrDDDDDD b = DCC line bit error (details in DCR_Fault_DCD) B = DCD line bit error (details in DCR_Fault_DCD) D = DPU is down (details in DCR_DownStatus)	DC status for one side, including summary of down status.
DCR_DownStatus	R	5	rrRRRRRR rrWWWWWW rrMMMMMM rrFFFFFF R = read timeout W = write timeout M = DPU Data FIFO misread fault F = DPU other fault	DPU down status: Each 1 bit indicates a down DPU. If both M and F are set, the DPU is absent or not responding.
DCR_Fault_DCD	R	6	ffffffff ffffffff ffffffff ffffffff f = DCD line experienced a bit error	DCD bus bit error faults
DCR_Fault_DCC	R	7	rrrrrrrr rrrrrrrr rrrrrfff ffffffff f = DCC line experienced a bit error	DCC bus bit error faults

Notes:

**4. DCR\_Control:** comments from dc\_side.v:

```
A: // 1/16 * number of words RetCmd FIFO may contain before being considered almost full
    // a smaller A value tends to reduce latency but also reduces throughput
    // suggested initial A value: 4 (-->64 words or 32 commands in RetCmd FIFO)
P: // number of DC_Reads that can be outstanding before arriving/departing priority rules change
    // once this threshold is crossed, the next departing DC_Read gets priority if all of its targets are ready for readout
    // a smaller P value tends to reduce latency -- too small of a value (e.g., 1 or 2) can substantially reduce throughput
    // suggested initial P value: 8
T: // timeout interval in DC_CLK cycles
```

## DC Registers (continued)

In DPU XB FPGA:		rrrr		
DCR_DPU_Fifos		0		do not access – address is reserved for Command and Data FIFO's
DCR_DPU_Control	R/W	1	EEEEEEEE LLLLLLLL rrrrrrrC PPPPfrBD E = LED enables: FFFrTWRM FFF = front-end T = fault W = reg or FIFO write R = reg or FIFO read M = manual (under control of HPU software) L = LED minimum duration, unit ~ms C = check DC/DSP address P = priority postpone DC is allowed P*16 clock cycles before it must surrender to the higher-priority front end F = DPU XB FPGA reset B = DSP big endian D = DSP run (else reset)	
DCR_DPU_Status	R	2	rrrrrrUV rrrrARFO rrrrIIII rrXXXXXX U = user's front end subsystem detected a fault this fault is not fatal to DC V = DC_Verify detected a verification error A = command FIFO is almost full R = ready for readout (of queued data for DC_Read) F = any DC fault except readout over/under read fault O = readout over/under read fault I = command tracker faults: inv ovf unf sum inv = invalid command at Command FIFO input ovf = command FIFO overflow unf = command FIFO underflow (overread) sum = summary of command tracker faults X = command executor faults: din inv wra rda ali sum din = data subsystem input fault inv = invalid command at Command FIFO output wra = DSP write address out of range rda = DSP read address out of range ali = DSP address is not word aligned sum = summary of command executor faults	status of DPU XB FPGA
DCR_DPU_WR_Limit_Lo	R/W	3	AAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA A = address in DPU	constrains writes to DPU memory
DCR_DPU_WR_Limit_Hi	R/W	4	AAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA A = address in DPU	constrains writes to DPU memory
DCR_DPU_RD_Limit	R/W	5	HHHHHHHH HHHHHHHH LLLLLLLL LLLLLLLL H = MS halfword of high address limit e.g., 8000 → maximum legal address of 7FFF FFFC L = MS halfword of low address limit e.g., 4000 → minimum legal address of 4000 0000	constrains reads of DPU memory
DCR_DPU_Capture	R	7	rrrrrrrr rrrrrrrr rrrrrCCC CCCCCCCC C = value of DCC lines AAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA A = DSP address that caused error after DC_Verify AAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA A = last targetted address upon command executor fault	value captured from DCC lines in response to capture_next_dcc, or address of verification error, or  address at time of command executor fault
DCR_DPU_CLR_Verify	W	7	rrrrrrrr rrrrrrrr rrrrrrrr rrrrrrrr	clears the Verify flag

r = reserved (host software should set to zero)

S = value can also be read by GetDC\_Status

## DC Control Modes

---

MMMM	Mode	Target	DCC	DCD	DCR_TestIn	typical DPU XB FPGA state
0	Idle		drive nop	float	no change	any
1	Float		float	float	no change	not configured
2	RunLoopback		drive	i / o (i is ignored)	no change	any
3	RunNormal		drive	i / o	no change	any
	test modes:					
8	DriveDCC	TTT	pulse capture_next_dcc pulse DCR_FailureValue	float	capture DCC	any
9	DriveDCD		drive nop	pulse DCR_FailureValue	capture DCD	not configured
10	DriveDPU_Flags	all	pulse nop_default pulse drive_flags	zero each DPU drives flag bits	capture DCD	configured
11	WriteDPU_Register	TTTTTT	pulse write_register	pulse DCR_FailureValue	no change	configured
12	ReadDPU_Register	TTT	pulse read_register	driven by target DPU	register value	configured

### Notes:

1. “capture DCC” and “capture DCD” capture the DCC or DCD bus state at the end of the clock cycle during which the bus is pulsed.
2. DriveDCC will cause the targeted DPU XB FPGA to capture the DCC bus value (DCR\_TestOut value) in its DCR\_DPU\_Capture register.

### Example Tests of DC Wiring:

#### Test for shorted lines:

drive DCC, or  
drive DCD, or  
write DPU register (e.g., test register) + read DPU register

#### Test for bad DPU ID straps/non-configured DPU XB FPGA:

drive DPU flags  
to test for the unlikely case that two DPU’s have swapped ID straps, each DPU XB FPGA can be deconfigured in turn

#### Test for opens at DC FPGA:

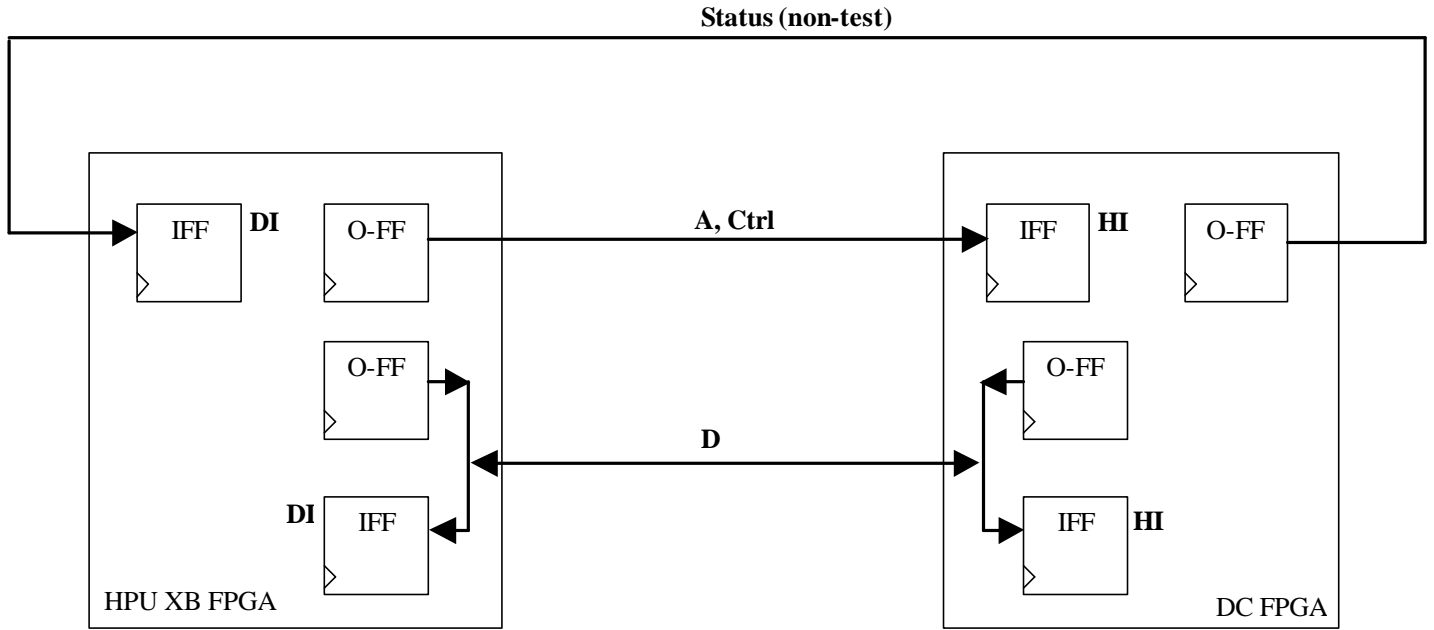
no direct test—must rely on open tests at DPU’s

#### Test for DCD lines open at DPU:

write DPU register (e.g., test register) + drive DCD (e.g., to zero DCD bus) + read DPU register

#### Test for DCC lines open at DPU:

capture DPU DCC + read DPU register (read DCC\_Capture)



WR_N, RD_N >>	A >>	status flags <>	RDAV_N <>	DCD_H <>	Comment
nop	>>				
write	write addr >>				
write	write addr >>				
read	read addr >>	valid <<	1	float	DCD_H is floated by both FPGA's when not in use
read	read addr >>	valid <<	1	write data >>	
test	test addr >>	valid <<	1	write data >>	
		valid <<	1	read data <<	invalid data is driven by DC FPGA
		valid <<	0	read data <<	valid data is driven by DC FPGA
		test >>	test >>	float	HPU XB FPGA drives test value onto most DCC lines

>> = to DC FPGA  
 << = to HPU XB FPGA  
 <> = direction not fixed

## Critical Path Timing Tables

Generic DC (fully pipelined):

description	component	delay	
clock-to-output delay	DPU or Host FPGA	7.6	(slow, 8mA, with DLL, worst case: CLK to lo-Z)
wiring + settling delay	wire + load	10.0	
clock skew	clock driver(s) + wires + loads	1.7	ns (driver → .2 ns, 40 pF * 25 ohm → 1 ns)
input setup	Host FPGA or DPU	1.9	IFF
<i>total</i>		21.2	ns

XB FIFO Timing Data and Control Signals Setup at XB FPGA:

description	component	delay	
clock-to-output delay	DSP	6.0	ns (worst case: XD)
wiring + settling delay	wire + load	0.5	ns
input setup	XB FPGA	1.9	ns (-5, with DLL)
clock (XFCLK) delay	mux	0.0	ns (credit, conservative)
<i>total</i>		8.4	ns (must be less than XFCLK period, e.g., 13.3 ns)

XB FIFO Timing DSP Data and Control Signals Hold at XB FPGA:

description	component	delay	
clock-to-output delay	DSP	1.5	ns (for control signals – no specification for XD)
wiring + settling delay	wire + load	0.0	ns
clock (XFCLK) delay	mux	-0.5	ns (debit, conservative)
input hold	XB FPGA	0.0	ns (debit)
<i>total</i>		1	ns (must be nonnegative)

XB FPGA Data Setup at DSP's XD:

description	component	delay	
clock (XFCLK) delay	mux	0.5	ns (conservative)
clock-to-output delay	FPGA	6.8	ns (max, slow, 8mA, -5, with DLL)
wiring + settling delay	wire + load	0.5	ns
input setup	DSP	3.0	ns
<i>total</i>		10.8	ns (must be less than XFCLK period, e.g., 13.3 ns)

XB FPGA Data Hold at DSP's XD:

description	component	delay	
clock (XFCLK) delay	mux	0.0	ns (credit, conservative)
clock-to-output delay	FPGA	3.5	ns (min, slow, 8mA, -5, with DLL)
wiring + settling delay	wire + load	0.0	ns (conservative)
input hold	DSP	-2.5	ns (debit)
<i>total</i>		1	ns (must be nonnegative)

## Other Time Numbers

---

Time to fill empty front-end FIFO, no DMA draining FIFO:

8 time slice FIFO capacity, 4 time slices per trigger, 100 kHz trigger rate

→ 20 us

Time to drain full front-end FIFO, 75 MW/s DMA draining FIFO, data continues to fill FIFO at average rate:

8 time slices in FIFO, 100 W per time slice, 4 time slices per trigger, 100 kHz trigger rate

→ 40 MW/s incoming rate

→ 35 MW/s drain rate

→ 23 us

Safe DC XB utilization:

10.0 us of DC allows filling of ½ front-end FIFO

11.5 us of DMA drains ½ of front-end FIFO

→ 47% of XB time may be used for DC

Average Synchronous Host bandwidth:

50 MHz XB clock

50% overhead

→ 250 W/10 us of DC activity (25 MW/s)

→ 250 W/21.5 us

→ 11.6 MW/s

## Failures Most Likely Caused by Hardware:

Failure	Fatal to DPU?	Fatal to DC?	How Detected?	Reaction
Two DPU's have same ID strap value	n/a	yes	pre-operation check of flags (deconfiguring DPU XB FPGA's can help identify faulty DPU)	HPU does not use DC and reports DC failure.
Shorted DCC or DCD line	n/a	yes	software pre-operation scan via test registers (before DPU FPGA's are configured)	HPU does not use DC and reports DC failure.
Open DCC or DCD line	yes	yes	software verification of writes to DPU's, one DPU at a time	HPU does not use DC and reports DC failure.
Writing to one or more targets times out	yes	no	timer expires	DC FPGA examines DPU flag bits, updates its down DPU list, and resumes writing to remaining targets, if any.
Reading from a target times out	yes	no	timer expires	DC FPGA examines DPU flag bits, updates its down DPU list, and resumes reading from remaining targets, if any. Dummy data words (FailureValue) are supplied for down targets.
Data read back on DCD does not match data driven onto DCD.	no	no	DPU XB FPGA and DC FPGA monitor and DPU ??????	Non-fatal faults are recorded in the XB FPGA and/or one of the SIDE modules in the DC FPGA.
A readout is attempted with no associated fetch.	yes	no	DC FPGA observes the DPU's NR status code (nothing to read out).	The read operation proceeds after all DPU's with a read/fetch mismatch are added to the down list.
Readout word count does not match fetch word count.	yes	no	At end of readout, DPU XB FPGA detects that either too few or too many words were read out.	DC FPGA detects this error only after all DPU's have been read out. Data words for the faulty DPU are therefore undefined. The fault will be reflected in the status word for this command, and the DPU will be added to the down list.

To allow testing of failure recovery mechanisms, the DPU XB FPGA control register has bits to (1)disable writing to DPU RAM, (2) disable reading from DPU RAM, and (3) override flags.

## Failures Most Likely Caused by Software:

Fault	Fatal to DPU	Fatal to DC?	Where Detected	Reaction
The two LSB's of the DPU address are not zero.	yes	no	DPU XB FPGA	A fault flag is set, and the DPU XB FPGA enters the FAILED state.
The DPU address is out of bounds, i.e., outside of the page specified by DCR_DPU_Page and DCR_DPU_Mask.	yes	no	DPU XB FPGA	A fault flag is set, and the DPU XB FPGA enters the FAILED state.
DC_ReadRegister or DC_WriteRegister attempts to access an illegal location (e.g., register 0) .	no	no	DC FPGA	The command is considered to be legal, but it is ignored. No fault is recorded.

## DC Command Stream Format (HPU → DC FPGA)

DC command stream format:

CCCC I III 00NN NNNN t tTT TTTT t tTT TTTT (Command)  
 [Address in DPU]  
 [Data]...[Data]

C = command

I = inverted command, used to verify valid command

0 = must be zero, used to verify valid command

N = data count (per target on read), also used to specify a register for DC\_WriteRegister and DC\_ReadRegister

T = target DPU(s) (side A targets are in byte 0, side B in byte 1)

t = target bits that are not used as targets but are used by DC\_MarkReturn

DC Command Streams:

C 4 bits CCCC	N/r 6 bits	tT 16 bits	Length	Side Return Count	P?	DC Command Stream (HPU → DC FPGA)	DC Instruction Sequence (DC FPGA → DPU FPGA)
0	ignored	ignored	1	N/A	N/A	<b>DC_Pad</b>	none
1	ignored	ret word count	1	N/A	N/A	<b>DC_MarkReturn</b>	none
2	ignored	S_target	1	1+1	Y	<b>DC_Nop</b>	none
3	ignored	S_target	1	1+1+1	N	<b>DC_GetDC_Status</b>	none
4	00rrrr	target	1	1+TC+1	N	<b>DC_ReadRegister</b>	read_register
5	00rrrr	target	2	1+1	N	<b>DC_WriteRegister</b> Data	write_register, Data
6	read word count (per target)	target	2	1+N*TC+1	Y	<b>DC_Read</b> Address in DPU	Q: DC_Read (fetch) Q: Address in DPU read_fifo, read_fifo,...
7	write word count	target	N+2	1+1	Y	<b>DC_Write</b> Address in DPU Data, Data, ...	Q: DC_Write Q: Address in DPU Q: Data, Data, ...
8	write word count	target	N+2	1+1	Y	<b>DC_Verify</b> Address in DPU Data, Data, ...	Q: DC_Verify Q: Address in DPU Q: Data, Data, ...
9	ignored	target	2	1+1	Y	<b>DC_Interrupt</b> Dummy	Q: DC_Interrupt Q: Dummy

See notes on next page.

1. **ret word count:** Number of return words to be read out by DMA. The max legal value is DC\_FIFO\_LIMIT\_RET (see hpu\_dpu\_control.h). ret word count is queued in a dedicated FIFO (the Readout FIFO) upon arrival in the HPU XB FPGA. It is subsequently used to generate the RET\_RDY signal.
2. **S\_target:** Target DPU(s) whose status is returned.
3. **Side Return Count:** Number of words that will be returned in the SIDE module's Return FIFO. These words include 1 command word, zero or more data words, and one status word. TC == target count for the side serviced by the SIDE module. See Verilog source code.
4. **Q:** The word is queued in the DPU FPGA's write queue. Execution of non-queued instructions begins immediately, regardless of write queue contents.
5. **XXXX:** register address (accesses to register 0000 are ignored)
6. **P?:** Can the command preempt a DC\_Read, i.e., can the DC FPGA execute the command's instructions (except read\_fifo's) before executing the deferred read\_fifo instructions of a DC\_Read?
7. The DC FPGA may defer the read\_fifo instructions associated with a DC\_Read, allowing subsequent commands to send instructions to DPU's. The DC FPGA preserves the effective order of command execution—only the readout of fetched data is delayed.
8. HPU software should place zero in ignored fields.

## DC FPGA Command and Return Streams

---

<b>Command Stream from HPU</b>	<b>Return Stream from Side A(B)</b>	<b>Return Stream to HPU</b>	<b>(DCC)</b>
<b>DC_Pad</b>	N/A	N/A	N/A
<b>DC_MarkReturn</b>	N/A	N/A	N/A
<b>DC_Nop</b>	<b>DC_Nop</b> pre-command-status	<b>DC_Nop</b> pre-command-status	not used
<b>DC_GetDC_Status</b>	<b>DC_GetDC_Status</b> dc status Side A(B)  pre-command-status	<b>DC_GetDC_Status</b> dc status Side A dc status Side B pre-command-status	not used
<b>DC_ReadRegister</b>	<b>DC_ReadRegister</b> data Side A(B)...  post-command-status	<b>DC_ReadRegister</b> data Side A... data Side B... post-command-status	read_register
<b>DC_WriteRegister</b> Data	<b>DC_WriteRegister</b> pre-command-status	<b>DC_WriteRegister</b> pre-command-status	write_register
<b>DC_Read</b> Address in DPU	<b>DC_Read</b> data Side A(B)...  post-command-status	<b>DC_Read</b> data Side A... data Side B... post-command-status	Q, read from data FIFO
<b>DC_Write</b> Address in DPU Data...	<b>DC_Write</b> pre-command-status	<b>DC_Write</b> pre-command-status	Q
<b>DC_Verify</b> Address in DPU Data...	<b>DC_Verify</b> pre-command-status	<b>DC_Verify</b> pre-command-status	Q
<b>DC_Interrupt</b> Dummy	<b>DC_Interrupt</b> pre-command-status	<b>DC_Interrupt</b> pre-command-status	Q

data Side X... = data words (if any) for targets (if any) on side X

dc status: rpvvStatus (16 bits) : DCR\_SideStatus (16 LSB's)  
pre-command-status: status of targets after any DPU's are added to the down list but before command is executed  
post-command-status: status of targets after command is executed  
Q: queue command in the target Command FIFO(s) using write\_fifo

## DPU Bootloading Sequence

---

<b>DC Command Stream</b>	
DC_WriteRegister Data	Cause the DPU FPGA to activate the DSP's RESET line.
(wait 1 ms)	Waiting may be necessary because: 1. The DSP's PLL may need to stabilize (e.g., if the clock rate has been changed recently), 2. The DSP's XD bus must be allowed to settle if pullups/pulldowns are used to set the boot configuration.
DC_WriteRegister Data	Cause the DPU FPGA to deactivate the DSP's RESET line. (The DSP's CPU remains in reset while the remainder of the device is released from reset.)
DC_Write Address in DPU Data, Data, ...	Download code and/or data to the DPU's memory. (Multiple DC_Write commands will be necessary.)
DC_Read Address in DPU	Optionally read back and verify the DPU's memory contents.
DC_ReadRegister	Make sure there were no faults during the download process.
DC_Interrupt	Wake the DSP's CPU from reset.
DC_ReadRegister	Make sure there were no faults during the download process.
DC_Read Address in DPU	Perform any checks necessary to confirm proper initialization of DPU software. (Multiple DC_Read's as well as some DC_Write's may be necessary.)

The DC FPGA contains several configuration registers. Host software typically accesses these registers only for initialization and fault recovery. All registers are 32 bits wide, though many bits are unused. The CPU must use the DMA controller to access the configuration registers, because the DC FPGA is on the DSP's expansion bus and the configuration registers were deliberately made inaccessible via the DC Command Stream mechanism.

The configuration registers define two pages in Host memory and one page in DPU memory.

## DC Bus Protocol: DCC Definitions

### Opcodes

Opcodes (DCC) 11 bits	DCD Dir	Target(s)	Data (DCD) 32 bits	Operation	Meaning to DPU
1 TTT TTT 0000	wr	TTTTTT	data	write_fifo	write word to command FIFO
1 TTT TTT rrrr	wr	TTTTTT	data	write_register	write to register rrrr (rrrr != 0)
0 001 TTT rrrr	rd	TTT	data	read_register	read from register rrrr (rrrr != 0)
0 001 TTT 0000	rd	TTT	data	read_fifo	read from data FIFO
0 010 --- ----	rd	all	flags from all DPUs	drive_flags	drive four flag bits <b>RRCC</b> onto appropriate bits of DCD
0 011 0-- ----	float	all	float	pause	same as nop but legal only within a burst of read_fifo's
0 011 1-- ----	float	all	float	nop	no operation
0 011 1-- ----	wr	all	0xffffffff	nop_default	no operation, DC FPGA drives DCD to all 1's
0 1-- --- ----	float	all	float	capture_next_dcc	on the next clock cycle, ignore the DCC lines and capture their value in the DCR_DPU_Capture register

### Notes:

- A command consists of at most 65 words (DC\_Write + address + 63 data words).
- Absent DPU detection assumes DCD bus has been driven to all 1's via nop\_default. See status codes below.
- CC** = command status codes:
  - 11: fatal fault / DPU absent -- all fatal faults except Data FIFO misread
  - 00: reserved -- Command FIFO is not almost full, ok to write up to ~ 65 + 30 words
  - 01: reserved -- Command FIFO is not almost full, ok to write up to ~ 65 + 30 words
  - 10: AF -- Command FIFO is almost full
- RR** = return status codes:
  - 11: fatal fault / DPU absent -- Data FIFO misread (over/underread) fault only
  - 00: NR -- nothing to read out (both Data FIFO and Command FIFO are empty)
  - 01: PE -- a fetch is pending (data words are being read into the Return FIFO or a DC\_Read command is queued)
  - 10: RR -- ready for readout (the Return FIFO contains one command's worth of return data)
- DPU-detected fatal faults only affect the operation of the following opcodes:
  - write\_fifo (down DPU's will be removed from the target list)
  - read\_fifo (down DPU's will be removed from the target list and dummy data will be substituted for down DPU data)
- All user-issued commands except DC\_Pad and DC\_MarkReturn cause a status word to be returned after any returned data. The status word has the following format:

```

Side:          ----- Side B ----- ----- Side A -----
Target:                5 4 3 2 1 0                5 4 3 2 1 0
Status Word:   rr rr ss ss ss ss ss ss   rr rr ss ss ss ss ss
  
```

r = reserved, e.g., for DC FPGA status, zero when no faults

S = status for the target:

- 00 = success / non-targeted
- 01 = a DPU-local fatal fault was detected
- 10 = a DC-global fatal fault was detected
- 11 = reserved

## DC Bus Protocol: Write Sequence

DCC	DCD	Comment
		<b>***** Write words to the Command FIFO's of a set of targets *****</b>
drive_flags		do this initially only if previous indication of AF command FIFO
drive_flags		repeat until AF disappears (including due to timeout)
drive_flags		
drive_flags	flags	flags indicate not AF
drive_flags	flags	
write_fifo + TTTTTT	flags	all targets are written simultaneously
write_fifo + TTTTTT	flags	
write_fifo + TTTTTT	flags	
write_fifo + TTTTTT	D	
write_fifo + TTTTTT	D	write all words for the command in a single burst
write_fifo + TTTTTT	D	
write_fifo + TTTTTT	D	
write_fifo + TTTTTT	D	
drive_flags	D	terminal drive_flags (optional)
write_fifo + TTTTTT	D	next write (4 words)
write_fifo + TTTTTT	D	
write_fifo + TTTTTT	flags	flags indicate AF
write_fifo + TTTTTT	D	
drive_flags	D	repeat until AF disappears (including due to timeout)
drive_flags	D	
drive_flags	D	
drive_flags	flags	
drive_flags	flags	
write_fifo + TTTTTT	flags	next write (3 words)
write_fifo + TTTTTT	flags	
write_fifo + TTTTTT	flags	
drive_flags	D	terminal drive_flags (optional)
	D	
	D	
		<b>***** Write a word to a register in a set of targets *****</b>
write_register + TTTTTT		
	D	

### Write strategy:

1. A DPU's command FIFO is considered almost full (AF) if it cannot accept the maximum sized command (65 words) plus a safety margin of 30 words. The safety margin allows for pipelining, imperfect DC FPGA knowledge of DPU status, etc.
2. The DC FPGA maintains internally the last known AF state of each DPU's command FIFO, updating it sufficiently often to insure that no DPU command FIFO overflows. The DC FPGA contains a counter that counts write\_fifo's. A drive\_flags is issued when the write\_fifo counter reaches 10. For commands that write more than 10 words, the drive\_flags is performed only once—a few clock cycles before the last word is written. This insures that an update is performed before execution of the next command can start. The internal AF state is updated whenever possible, e.g., during a DC\_Read's terminal drive\_flags. Any update resets the write\_fifo counter.
3. While waiting for the targeted DPU's to become not AF, the DC FPGA may perform any readouts associated with queued DC\_Read commands.
4. A timeout of a DPU results in that DPU being placed in the down list.
5. Once all targeted DPU's (except any in the down list) are not AF, the DC FPGA bursts all words for the command to the DPU's. The terminal drive\_flags for writes is optional. The status returned to the user is the status before the write burst begins, but after any DPU's were added to the down list.
6. For best coexistence with reads, write data follows the write instruction after three clock cycles.
7. Writes to registers proceed regardless of (1) AF flags and (2) the down list.

## DC Bus Protocol: Read Sequences

DCC	DCD	Comment
drive_flags		***** Read four words from each of three targets *****
drive_flags	prev	do this initially only if previous indication of not ready target
drive_flags	prev	repeat until first target is ready for readout
drive_flags	flags	flags indicate first target has all data ready for readout
drive_flags	flags	
read_fifo + TTT <sub>0</sub>	flags	
read_fifo + TTT <sub>0</sub>	flags	target TTT <sub>2</sub> first appears ready here → no need to drive_flags for it later
read_fifo + TTT <sub>0</sub>	flags	
read_fifo + TTT <sub>0</sub>	D <sub>0</sub>	DPU XB FPGA must always have next word ready for readout
drive_flags	D <sub>1</sub>	target TTT <sub>1</sub> is not known to be ready → drive_flags
drive_flags	D <sub>2</sub>	issue drive_flags's or nop's if DC FPGA Return FIFO is AF
drive_flags	D <sub>3</sub>	issue drive_flags's or nop's if next target is not ready for readout
drive_flags	flags	flags indicate target TTT <sub>1</sub> is ready for readout
drive_flags	flags	
read_fifo + TTT <sub>1</sub>	flags	
read_fifo + TTT <sub>1</sub>	flags	remaining targets are ready for readout, proceed with back-to-back bursts
read_fifo + TTT <sub>1</sub>	flags	
read_fifo + TTT <sub>1</sub>	D <sub>0</sub>	
read_fifo + TTT <sub>2</sub>	D <sub>1</sub>	
read_fifo + TTT <sub>2</sub>	D <sub>2</sub>	
read_fifo + TTT <sub>2</sub>	D <sub>3</sub>	
read_fifo + TTT <sub>2</sub>	D <sub>0</sub>	
drive_flags	D <sub>1</sub>	
	D <sub>2</sub>	get final status, e.g., to check for an overread fault
	D <sub>3</sub>	
	flags	
		***** Read a register from each of three targets *****
		wait if Return FIFO in DC FPGA is almost full
read_register + TTT <sub>0</sub>		
read_register + TTT <sub>1</sub>		wait above → read_register's are always back-to-back
read_register + TTT <sub>2</sub>		
	D	
	D	
	D	

### Read strategy:

1. The DC FPGA obtains a DC\_Read command from the RetCMD FIFO. This same command and its associated source address have already been queued in the command FIFO of the targeted DPU's.
2. A DPU is ready for readout (RR) when it has fetched all the words requested by the oldest queued DC\_Read command.
3. The DC FPGA maintains an internal version of each DPU RR flag. The DC FPGA examines its internal RR flags before starting to read out each target. If the internal RR flags indicate that the current target is not RR, the DC FPGA updates its internal flags (using drive\_flags) until either (1) the current target is RR or (2) a timeout occurs. When the flags indicate the current target is RR, the DC\_FPGA reads out the DPU in a burst. Pause cycles are inserted within the burst if the Return FIFO is almost full.
4. While waiting for the first target to become RR, the DC FPGA may queue subsequent "arriving" commands in the DPU's. Once execution of a DC\_Read command begins and until it completes, no other commands can be executed.
5. The DC FPGA reads out each target in turn, clearing the internal version of the DPU's RR flag as soon as readout starts.
6. A targeted DPU always resets its return FIFO once the DC FPGA has read it out (as indicated by read\_fifo to the target being followed by any other operation (except pause)).
7. The read timeout interval begins when a DPU becomes the current target for a DC\_Read. If the interval expires and the DPU is still not RR, the DPU is added to the down list.
8. A DPU will be added to the down list if its flags indicate a misread fault. The DC\_Read's terminal status word indicates all down targets at the time execution of the command completes. I.e., if a DC\_Read causes a fault, that fault will be reflected in the command's status word.