



Science & Technology
Facilities Council

Slow Control and Bulk Data Meeting 21,22 Jan'19

Dr. Harry Walton

Rutherford Appleton Laboratory, UK

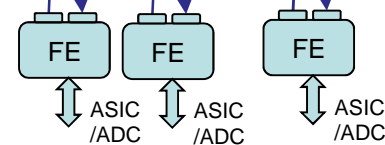
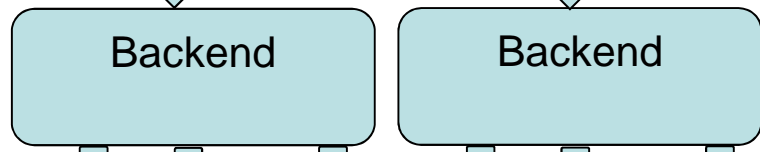
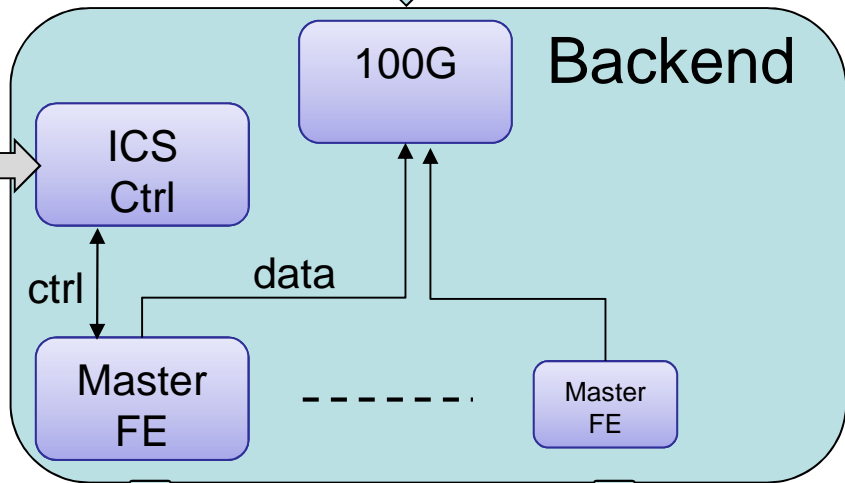
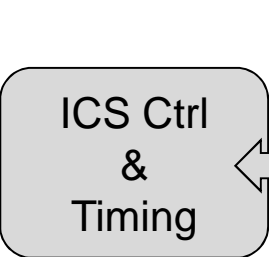
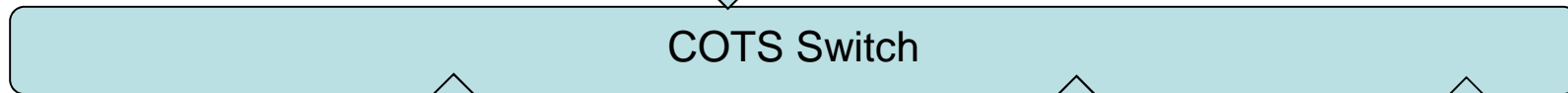


EUROPEAN
SPALLATION
SOURCE

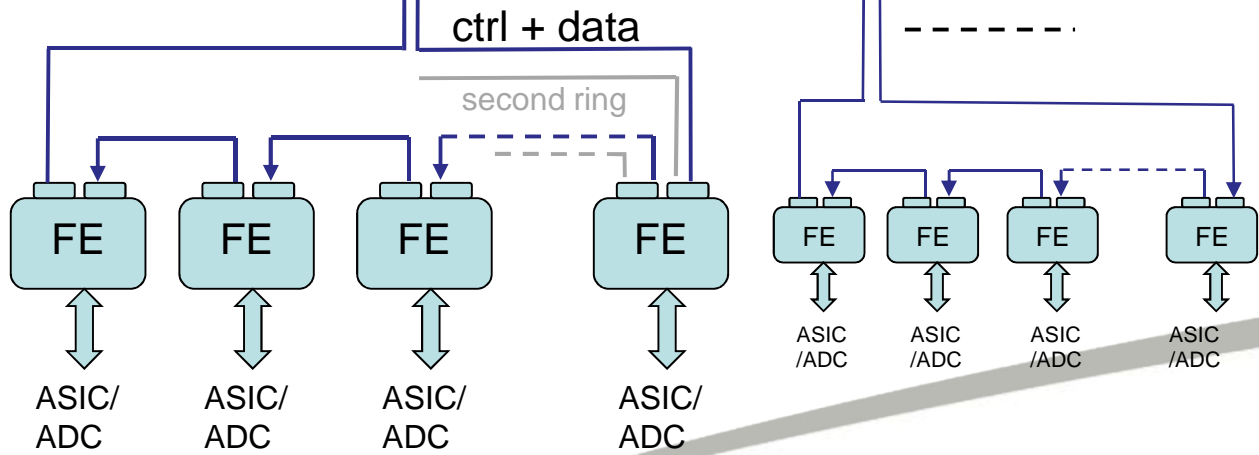


System Architecture

To DMSC



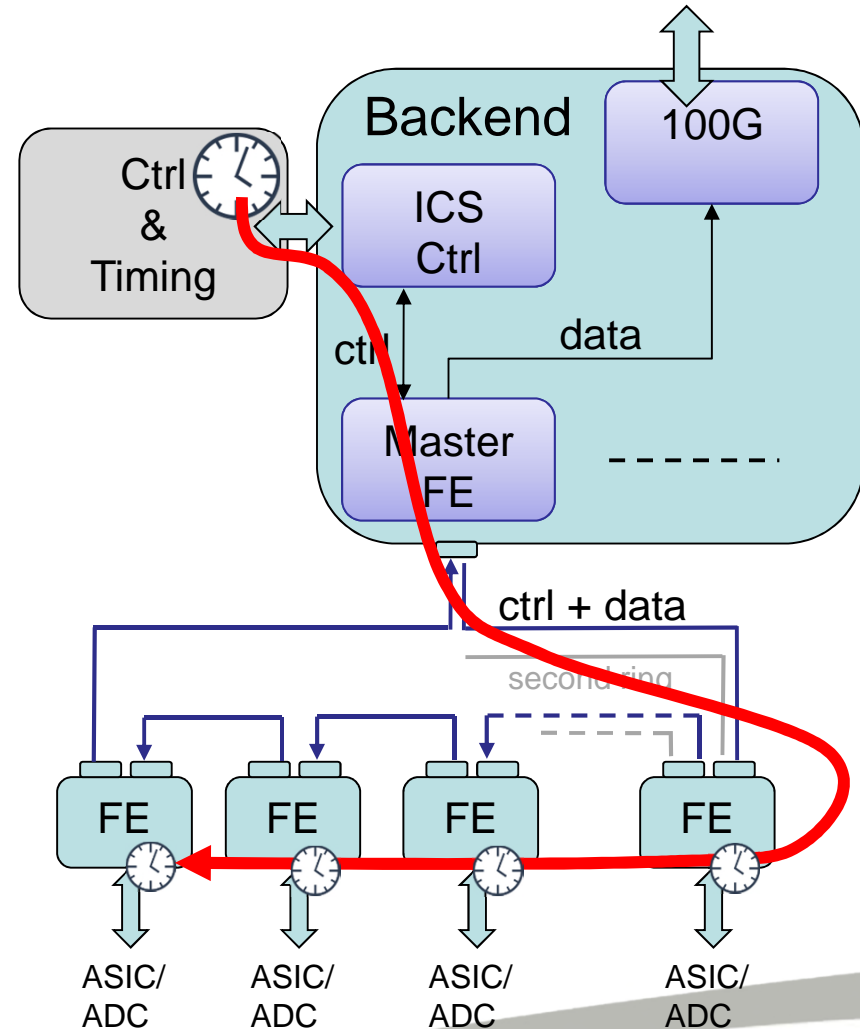
P-to-P Topology



Ring Topology

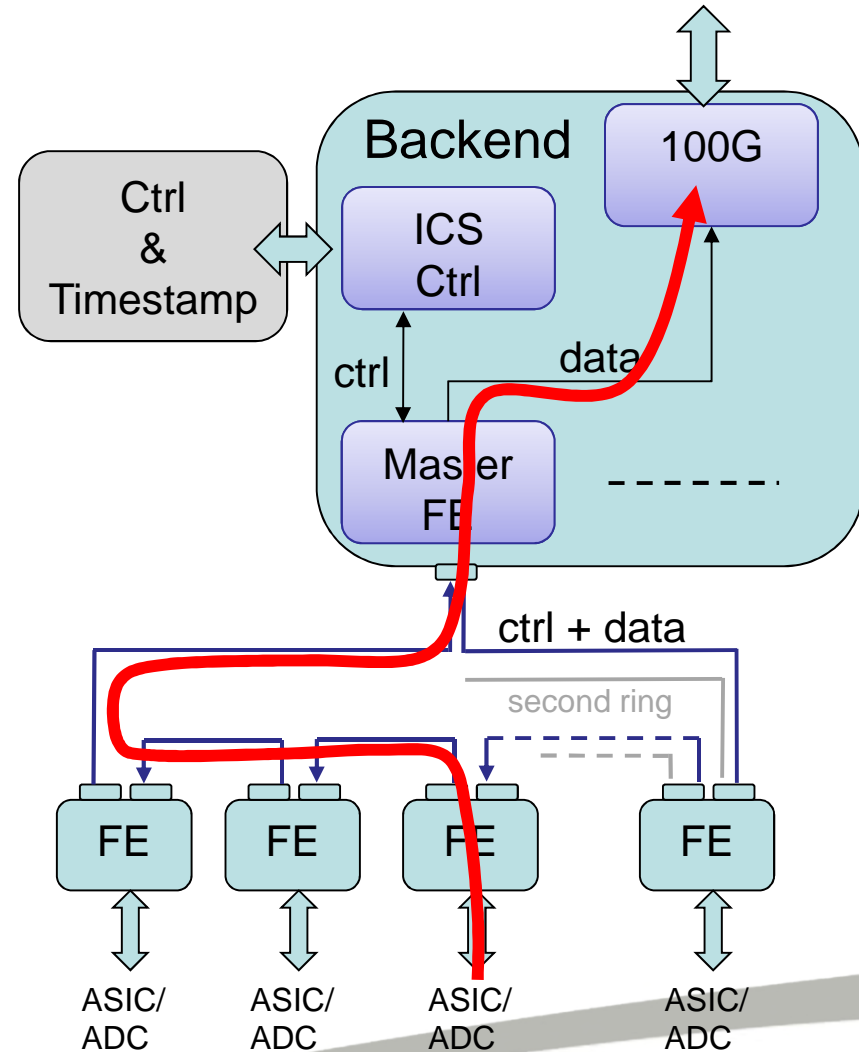
Front End Functions

- Acquire accurate timestamp.
- Collect digitized (timestamped) Bulk data, and downstream it to the BE.
- Receive & Return Memory Mapped Slow Control data (e.g. ADC-register W/R's)



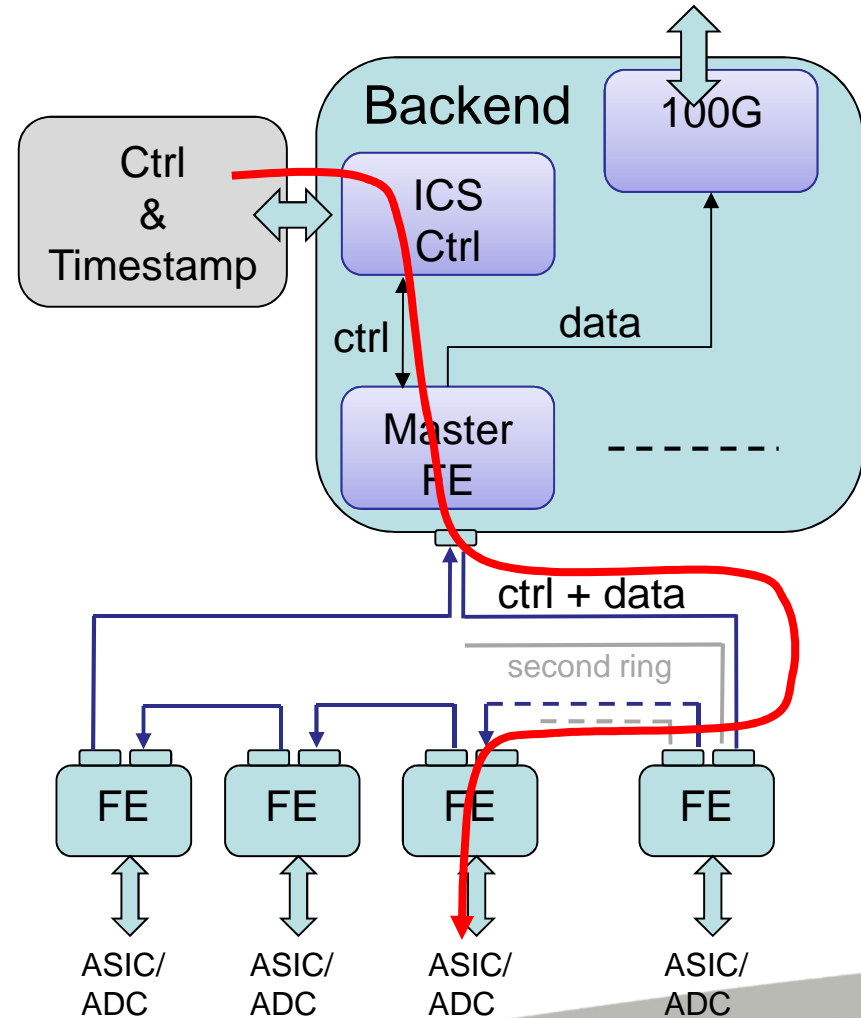
Front End Functions

- Acquire accurate timestamp.
- Collect digitized (timestamped) Bulk , and downstream it to the BE.
- Receive & Return Slow Control data (e.g. ADC-register W/R's)



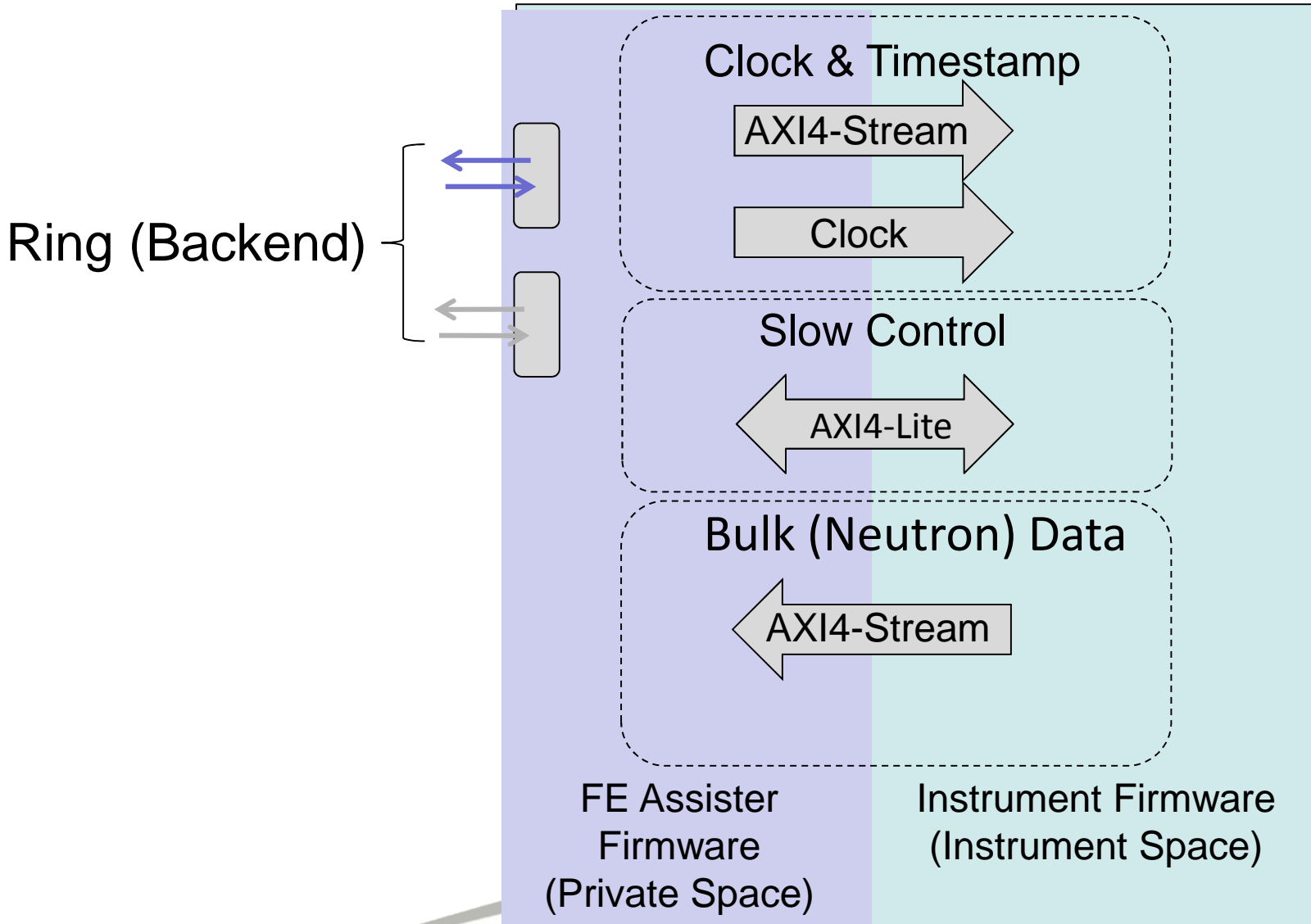
Front End (FE) Functions

- Acquire accurate timestamp.
- Collect digitized (timestamped) Bulk data, and downstream it to the BE.
- Receive & Return Memory Mapped Slow Control data (e.g. ADC-register W/R's)

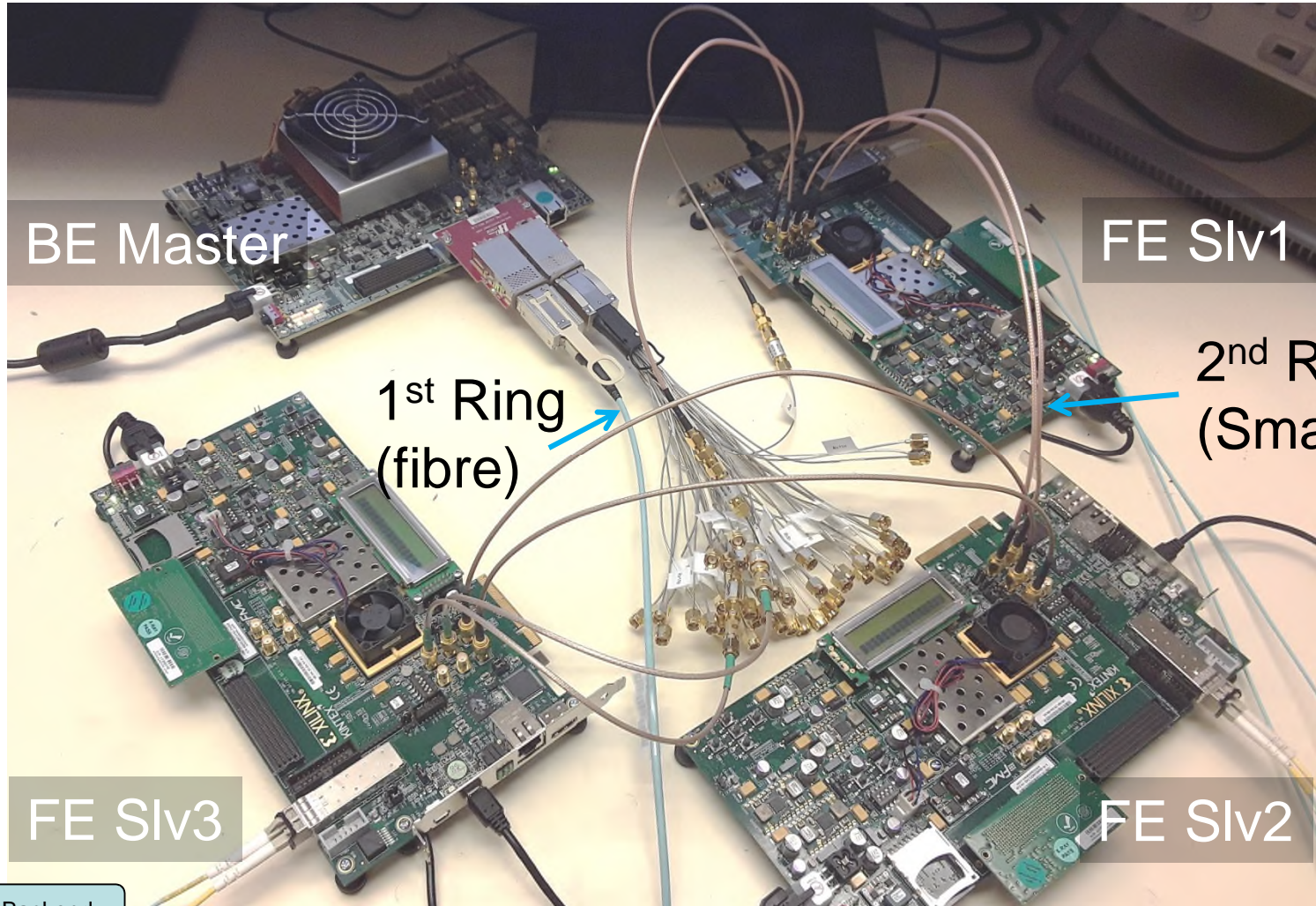




Front End User Interface



Previously (IKON 2018)



BE Master

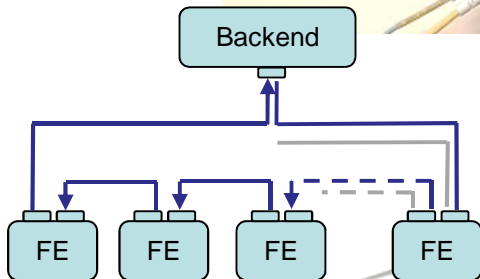
FE Slv1

1st Ring
(fibre)

2nd Ring
(Sma Cu)

FE Slv3

FE Slv2



Previously shown proof-of-concept

- Acquire a timestamp. ✓
- Collect digitized (timestamped) Bulk data, and downstream it to the BE. ✓
- Receive & Return Memory Mapped Slow Control data (e.g. ADC-register W/R's) ✓

Previous Limitations

- V. limited Slow Control & Diagnostics during ring bring-up

- Dual-ring & P-to-P operation

*Today's
update*

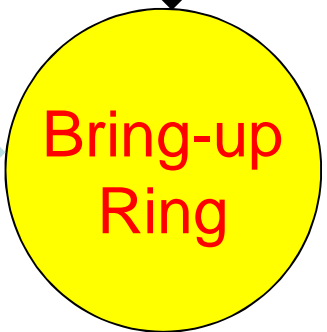
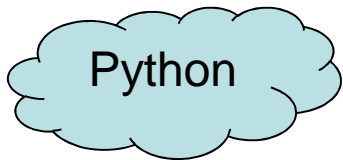
- Integration & test of larger (multi-FE) ring

- Integration with ESS 88MHz Timing

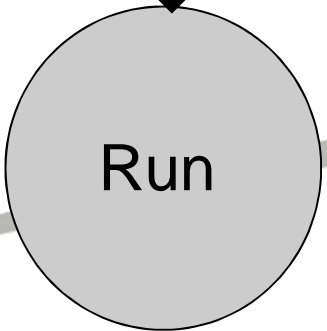
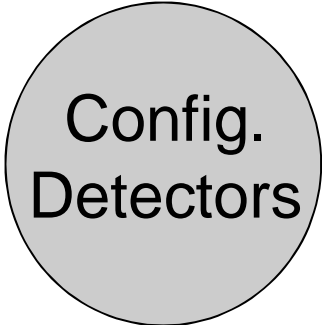
*Awaiting
requisite H/W*

“Getting to Science”

Ctrl I/f at present:



Power-on ↓



Control Protocol for ICS-DG Demonstrator

The demonstrator system uses 'serial-over-USB' to transmit binary information between the control box and FPGA detector nodes. The receiver hardware will be based on FTDI components for which drivers are freely available – more info on this later.

The basic unit of transmission is a byte. A 'big endian' model is used.

Transmitted commands and received responses (from ICS perspective) are handled as fixed size packets. This may not be a suitable in the final system, although the overheads for transmitting large amounts of configuration data probably wouldn't be an issue.

Generally speaking, the protocol needs to be tailored to the requirements of the FPGA slave implementation rather than the ICS master, because the master is implemented in software and can be more flexible, the slave is implemented in firmware which has less flexibility and tighter constraints.

The transmit format is (from ICS to slave)

	1 st Byte	2 nd Byte
Header Pattern	0xCA	0xFE
Command Word	Command MSB	Command LSB
Data/Addr Word 1		
Data/Addr Word 2		
Data Word 3		
Data Word 4		
Index Word	Index MSB	Index LSB
Checksum/Seq Word	Sequence Byte	Byte-wise Checksum

The receive format is (from slave to ICS)

Python over UART in accordance with ESS doc. *draft_protocol.pdf*
 'Control Protocol for ICS DG Demonstrator'





Memory Mapped System

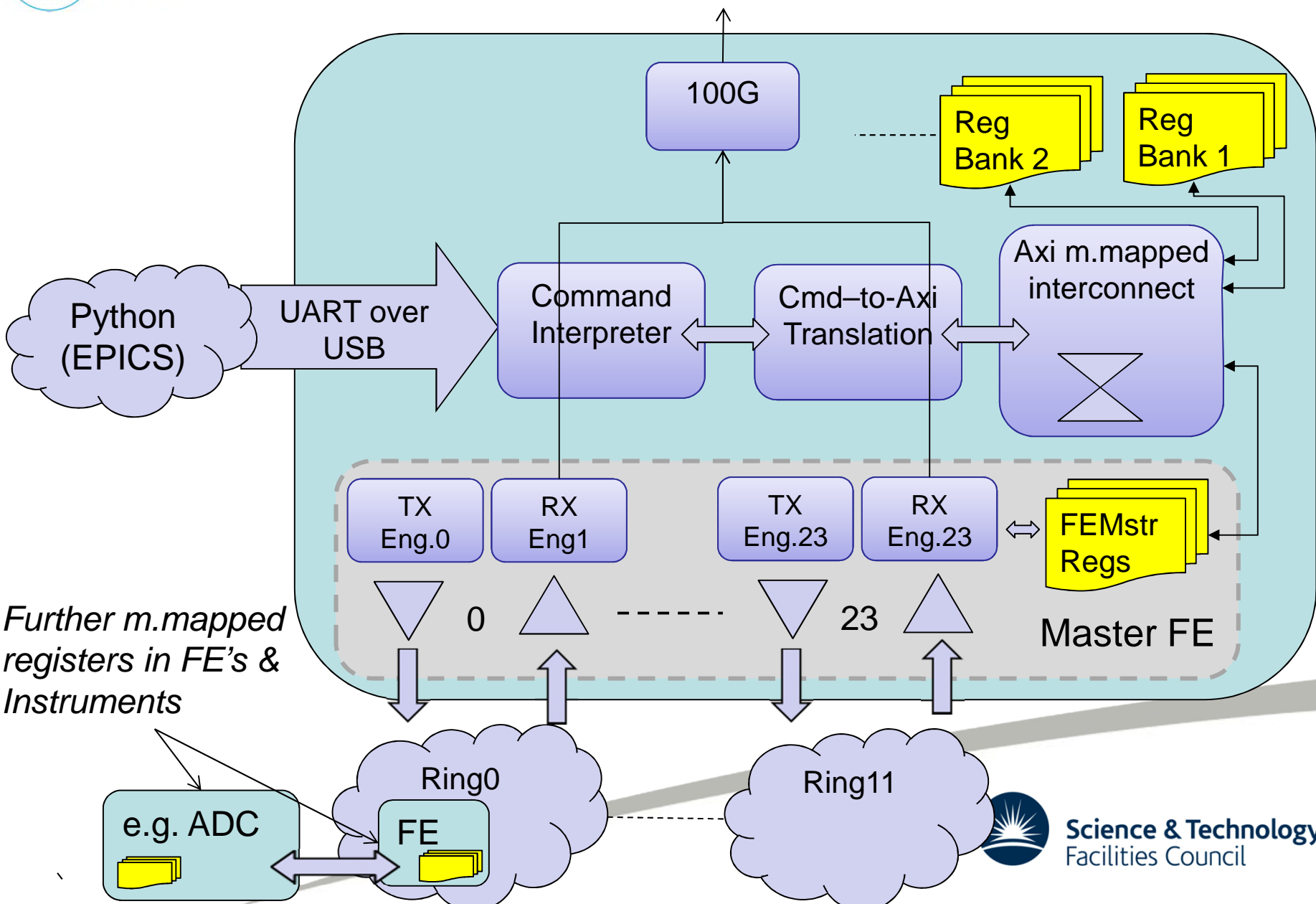
ESS Demonstrator Controls Address Space (for 24 Pt-2-Pt)

Address Range	Size	Region	Comments
0000'0000 – 3FFF'FFFF	1 GB	BE	
4000'0000 – 47FF'FFFF	128 MB	Ring 0	Local + User spaces
4800'0000 – 4FFF'FFFF	128 MB	Ring 1	
		...	
		...	
F800'0000 – FFFF'FFFF	128 MB	Ring 23	Max 24 Pt-2-Pt



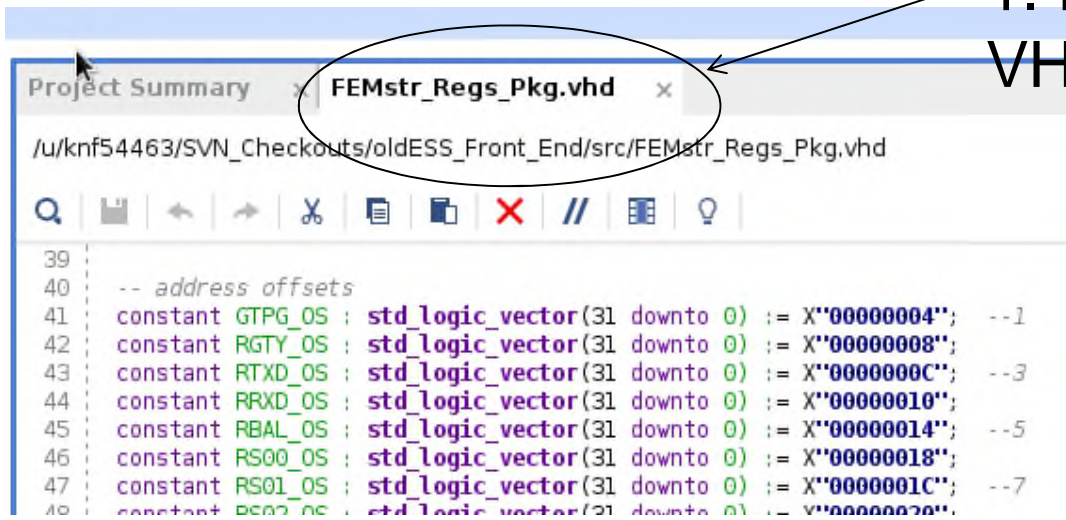


New Version Master Front End



Register Definition

1. Registers defined in original VHDL



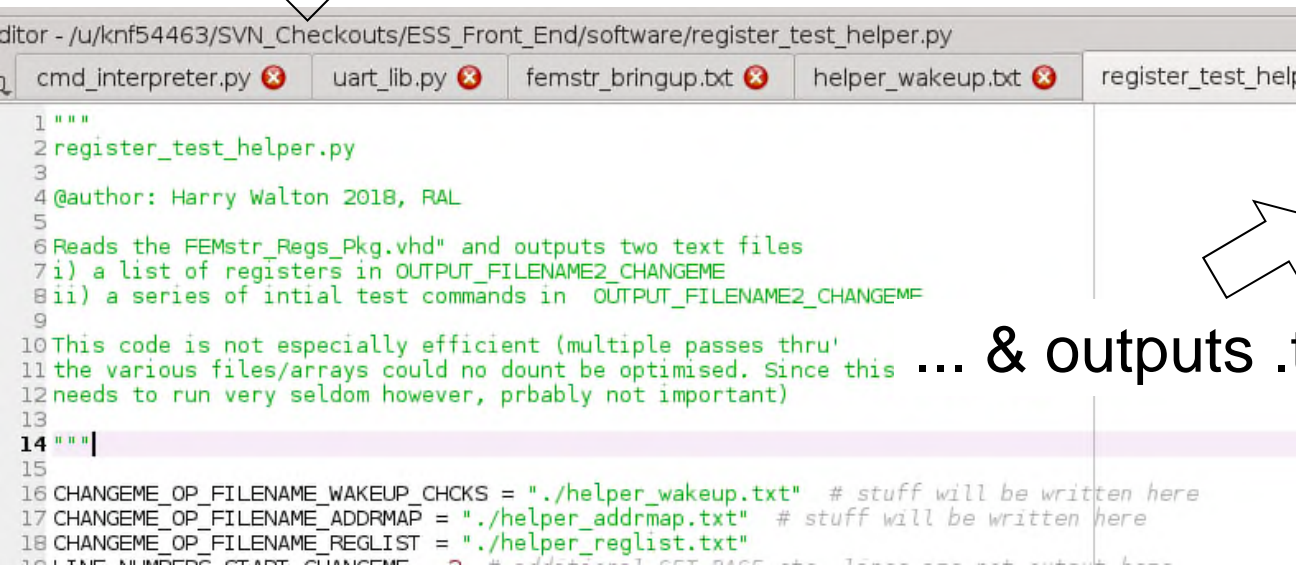
```

Project Summary | FEMstr_Regs_Pkg.vhd x
/u/knf54463/SVN_Checkouts/oldESS_Front_End/src/FEMstr_Regs_Pkg.vhd

39
40 -- address offsets
41 constant GTPG_OS : std_logic_vector(31 downto 0) := X"00000004"; --1
42 constant RGTY_OS : std_logic_vector(31 downto 0) := X"00000008";
43 constant RTXD_OS : std_logic_vector(31 downto 0) := X"0000000C"; --3
44 constant RRXD_OS : std_logic_vector(31 downto 0) := X"00000010";
45 constant RBAL_OS : std_logic_vector(31 downto 0) := X"00000014"; --5
46 constant RS00_OS : std_logic_vector(31 downto 0) := X"00000018";
47 constant RS01_OS : std_logic_vector(31 downto 0) := X"0000001C"; --7
48 constant RS02_OS : std_logic_vector(31 downto 0) := X"00000020";

```

2. Python helper-program parses...

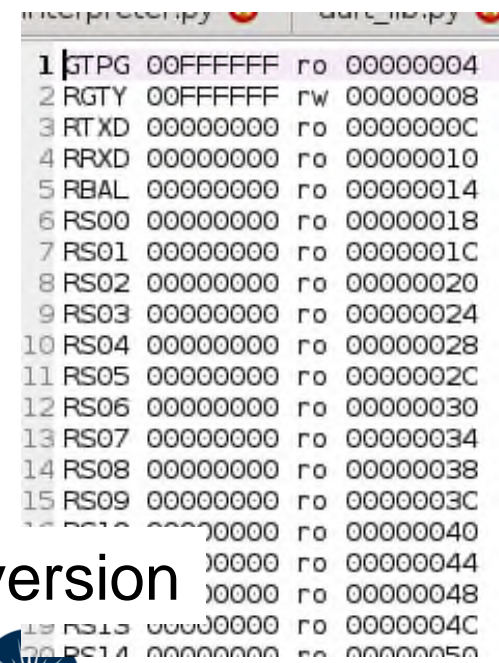


```

editor - /u/knf54463/SVN_Checkouts/ESS_Front_End/software/register_test_helper.py
cmd_interpreter.py x  uart_lib.py x  femstr_bringup.txt x  helper_wakeup.txt x  register_test_help

1 """
2 register_test_helper.py
3
4 @author: Harry Walton 2018, RAL
5
6 Reads the FEMstr_Regs_Pkg.vhd" and outputs two text files
7 i) a list of registers in OUTPUT_FILENAME2_CHANGEME
8 ii) a series of initial test commands in OUTPUT_FILENAME2_CHANGEME
9
10 This code is not especially efficient (multiple passes thru'
11 the various files/arrays could no doubt be optimised. Since this
12 needs to run very seldom however, probably not important)
13
14 """
15
16 CHANGEME_OP_FILENAME_WAKEUP_CHKCS = "./helper_wakeup.txt" # stuff will be written here
17 CHANGEME_OP_FILENAME_ADDRMAP = "./helper_addrmap.txt" # stuff will be written here
18 CHANGEME_OP_FILENAME_REGLIST = "./helper_reglist.txt"
19 LINE_NUMBERS_START_CHANGEME = 2 # additional CSS_PACG etc. lines can get output here

```



1	GTPG	00FFFFFF	ro	00000004
2	RGTY	00FFFFFF	rw	00000008
3	RTXD	00000000	ro	0000000C
4	RRXD	00000000	ro	00000010
5	RBAL	00000000	ro	00000014
6	RS00	00000000	ro	00000018
7	RS01	00000000	ro	0000001C
8	RS02	00000000	ro	00000020
9	RS03	00000000	ro	00000024
10	RS04	00000000	ro	00000028
11	RS05	00000000	ro	0000002C
12	RS06	00000000	ro	00000030
13	RS07	00000000	ro	00000034
14	RS08	00000000	ro	00000038
15	RS09	00000000	ro	0000003C
16	RS10	00000000	ro	00000040
17	RS11	00000000	ro	00000044
18	RS12	00000000	ro	00000048
19	RS13	00000000	ro	0000004C
20	RS14	00000000	ro	00000050

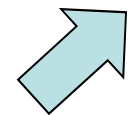
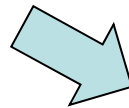
... & outputs .txt version

Python Bring-up

3. Python takes *register.txt* + a *bring-up-script*, and drives the UART

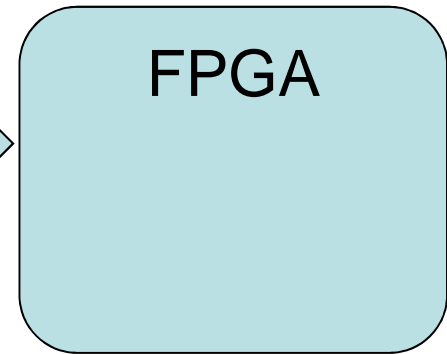
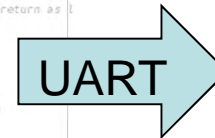
```

1 |PTG 00FFFFFF ro 00000004
2 |RTY 00FFFFFF rw 00000008
3 |RTX 00000000 ro 0000000C
4 |RRX 00000000 ro 00000010
5 |RBA 00000000 ro 00000014
6 |RS0 00000000 ro 00000018
7 |RS1 00000000 ro 0000001C
8 |RS2 00000000 ro 00000020
9 |RS3 00000000 ro 00000024
10 |RS4 00000000 ro 00000028
11 |RS5 00000000 ro 0000002C
12 |RS6 00000000 ro 00000030
13 |RS7 00000000 ro 00000034
14 |RS8 00000000 ro 00000038
15 |RS9 00000000 ro 0000003C
16 |RS10 00000000 ro 00000040
17 |RS11 00000000 ro 00000044
18 |RS12 00000000 ro 00000048
19 |RS13 00000000 ro 0000004C
20 |RS14 00000000 ro 00000050
  
```



```

46 seq_number = 0
47
48 # default the base address to 0
49 base_addr = 0
50
51 # open command script (this specifies the reads/writes to issue)
52 script_path = script_file
53 fpr = open(script_path, 'r')
54
55 # parse script and execute read/writes accordingly
56
57 for line in fpr:
58     logging.debug("Parsing %s", script_file)
59     logging.debug("Read line %s :", line[:-1]) #remove carriage return as l
60     word_arr = line.split(" ")
61     # ignore commented lines
62     if ("##" == word_arr[0][0]) or ('\n' == word_arr[0][0]):
63         continue
64     # define behaviour for 'SET' commands
65     elif "MSG" == word_arr[1]:
66         logging.info("%s", line[:-1])
67         seq_number += 1 # dont increment seq_number for messages
68     elif "SET" == word_arr[1]: #word_arr[0]:
69         # set base address
70         if "BASE" == word_arr[2]: #word_arr[1]:
71             if word_arr[3][0:2] == "0x": #word_arr[2][0:2] == "0x":
72                 base_addr = int(word_arr[3].rstrip(), 16) #int(word_arr[2].r
73             else:
74                 base_addr = int(word_arr[3].rstrip(), 10) #int(word_arr[2].r
75             logging.info("Setting Base address set to: %s", hex(base_addr))
76             #print "base address set to: ", base_addr #try: base_addr.hex()
77             if base_addr == GLOBAL_BASE:
78                 address_file = address_dir + "address_map_global.txt"
79             elif base_addr == RING_BASE:
80                 address_file = address_dir + "helper_addrmap.txt"
  
```



```

cmd_interpreter.py x  uart_lib.py x  femstr_bringup.txt x  helper_w
1 10 MSG Reading file femstr_bringup.txt
2 1 SET BASE 0x44A00000
3 #
4 2 MSG Checking all registers are awake and correctly intialised
5 3 AGET GTPG 0x00FFFFFF
6 4 AGET RGTY 0x00FFFFFF
7 5 AGET RTXD 0x00000000
8 6 AGET RRXD 0x00000000
9 7 AGET RBAL 0x00000000
10 8 AGET RS00 0x00000000
11 9 AGET RS01 0x00000000
12 10 AGET RS02 0x00000000
  
```


Bring-Up Script Snippet

(Example : powering up 3 of the available 24 “rings”)

print message to python console

```
128 120 AGET PDPL 0xFFFFFFFFB6
129 #
130 121 MSG Reset all GTY
131 122 SET RGTY 0xFFFFFFFF
132 123 AGET RGTY 0xFFFFFFFF
133 124 MSG Un-reset 1st, 4th and 7th GTYs (0xB6 = 1011_0110)
134 125 SET RGTY 0xFFFFFB6
135 126 AGET RGTY 0xFFFFFB6
136 #
137 127 MSG Check reset cntrlr sees 1st, 4th & 7th GTY out of rst (0x49=0100_1001)
138 128 AGET RTYD 0x00000049
```

poke 0xFF...B6 to RGTY reg.

check (peek) RGTY reg. now holds 0xFF...B6

(Future nice-to-haves:

- More commands: Read-Set-a-bit-Writeback etc.
- Named bitmasks etc.)

Running the bring-up script

```

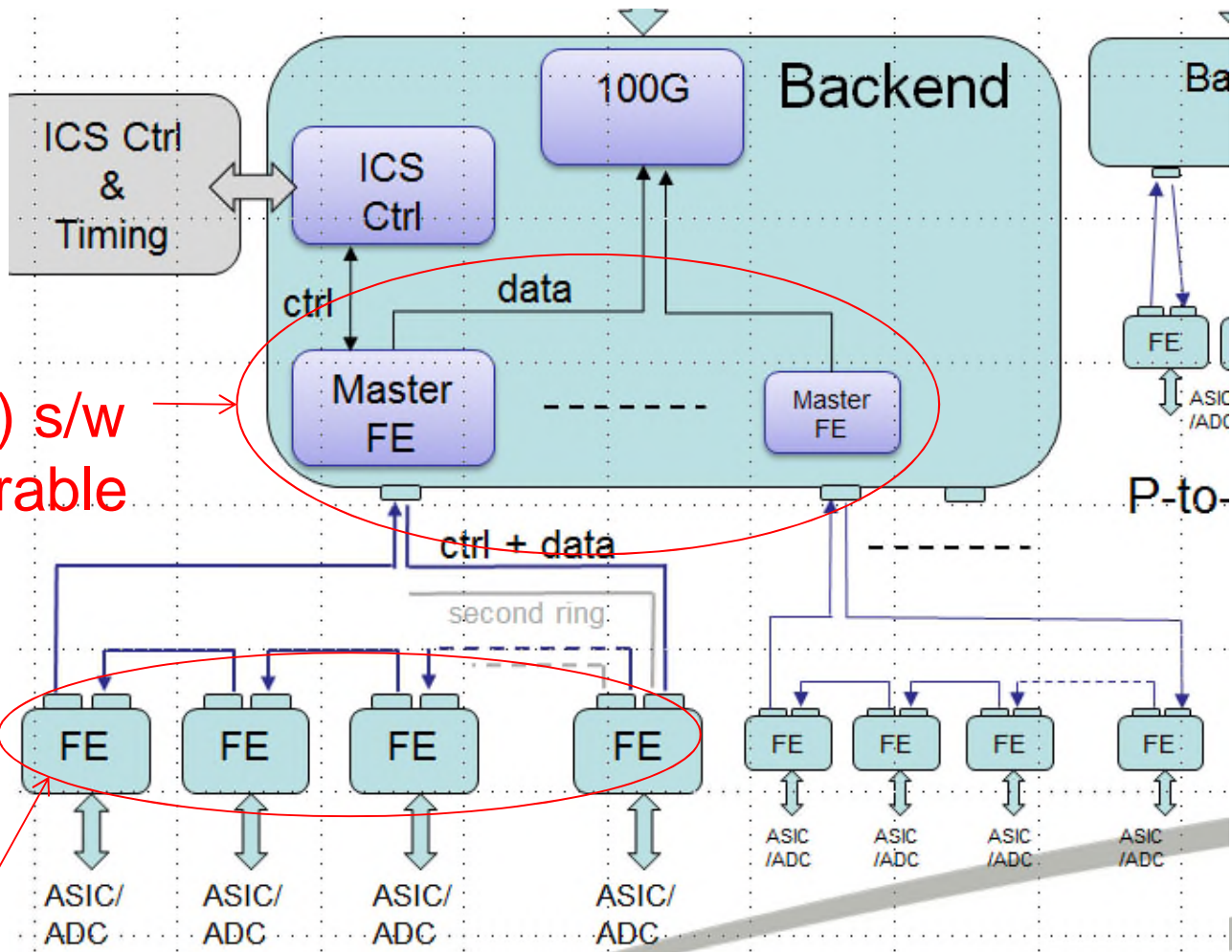
118 MSG ...So un-powerdown respective CPI
119 SET PDPL 0xFFFFFFFFB6
120 AGET PDPL 0xFFFFFFFFB6
#
121 MSG Reset all GTY
122 SET RGTY 0xFFFFFFFF
123 AGET RGTY 0xFFFFFFFF
124 MSG Un-reset 1st, 4th and 7th GTYs
125 SET RGTY 0xFFFFFFFFB6
126 AGET RGTY 0xFFFFFFFFB6
#
127 MSG Check reset cntrlr sees 1st, 4th
128 AGET RTXD 0x00000049
129 AGET RRXD 0x00000049
#
130 MSG Check the (state-machine) status
131 MSG ...Each nibble of TCSx encodes s
132 MSG ...include TXIDLEst=0000, TXBGUP
132 AGET TCS0 0x00000000
132 AGET TCS1 0x00000000
INFO: 118 MSG connects to cable-t-hell-9, ar
INFO: 118 MSG ...So un-powerdown respective
INFO: o.k.
INFO: 121 MSG Reset all GTY
INFO: o.k.
INFO: 124 MSG Un-reset 1st, 4th and 7th GTYs
INFO: o.k.
INFO: 127 MSG Check reset cntrlr sees 1st,
INFO: o.k.
INFO: o.k.
INFO: 130 MSG Check the (state-machine) stat
INFO: 131 MSG ...Each nibble of TCSx encodes
INFO: 132 MSG ...include TXIDLEst=0000, TXBC
INFO: o.k.
INFO: o.k.
INFO: 158 MSG Put all TX engines in TXIDLEs
INFO: 133 MSG In the TXIDLEst the TX engine
INFO: 134 MSG ...interrogating all the TSxx
INFO: o.k.
2019-01-15 13:40:03,148 uart_lib.py, 182 DEBUG ***** Sequence No.= 119
2019-01-15 13:40:03,149 uart_lib.py, 58 DEBUG Parsing femstr_bringup.txt
2019-01-15 13:40:03,149 uart_lib.py, 59 DEBUG Read line # :
2019-01-15 13:40:03,149 uart_lib.py, 58 DEBUG Parsing femstr_bringup.txt
2019-01-15 13:40:03,149 uart_lib.py, 59 DEBUG Read line 121 MSG Reset all GTY :
2019-01-15 13:40:03,149 uart_lib.py, 66 INFO 121 MSG Reset all GTY
2019-01-15 13:40:03,150 uart_lib.py, 182 DEBUG ***** Sequence No.= 119
2019-01-15 13:40:03,150 uart_lib.py, 58 DEBUG Parsing femstr_bringup.txt
2019-01-15 13:40:03,150 uart_lib.py, 59 DEBUG Read line 122 SET RGTY 0xFFFFFFFF :
2019-01-15 13:40:03,150 uart_lib.py, 124 DEBUG Attempting write to RGTY = 0x44a00008 with 0xffffffff
2019-01-15 13:40:03,151 uart_lib.py, 231 DEBUG Setting write start address to 0x44a00008
2019-01-15 13:40:03,153 uart_lib.py, 238 DEBUG Writing data 0xffffffff
2019-01-15 13:40:03,155 uart_lib.py, 182 DEBUG ***** Sequence No.= 120
2019-01-15 13:40:03,155 uart_lib.py, 58 DEBUG Parsing femstr_bringup.txt
2019-01-15 13:40:03,156 uart_lib.py, 59 DEBUG Read line 123 AGET RGTY 0xFFFFFFFF :
2019-01-15 13:40:03,156 uart_lib.py, 168 DEBUG Attempting read-with-assert from RGTY = 0x44a00008
2019-01-15 13:40:03,156 uart_lib.py, 255 DEBUG Setting read start address to 0x44a00008
2019-01-15 13:40:03,161 uart_lib.py, 170 DEBUG Read value 0xffffffff

```





Status & Next Steps



Now, (largely) s/w configurable

Next, adapt FE RTL similarly for s/w bring-up



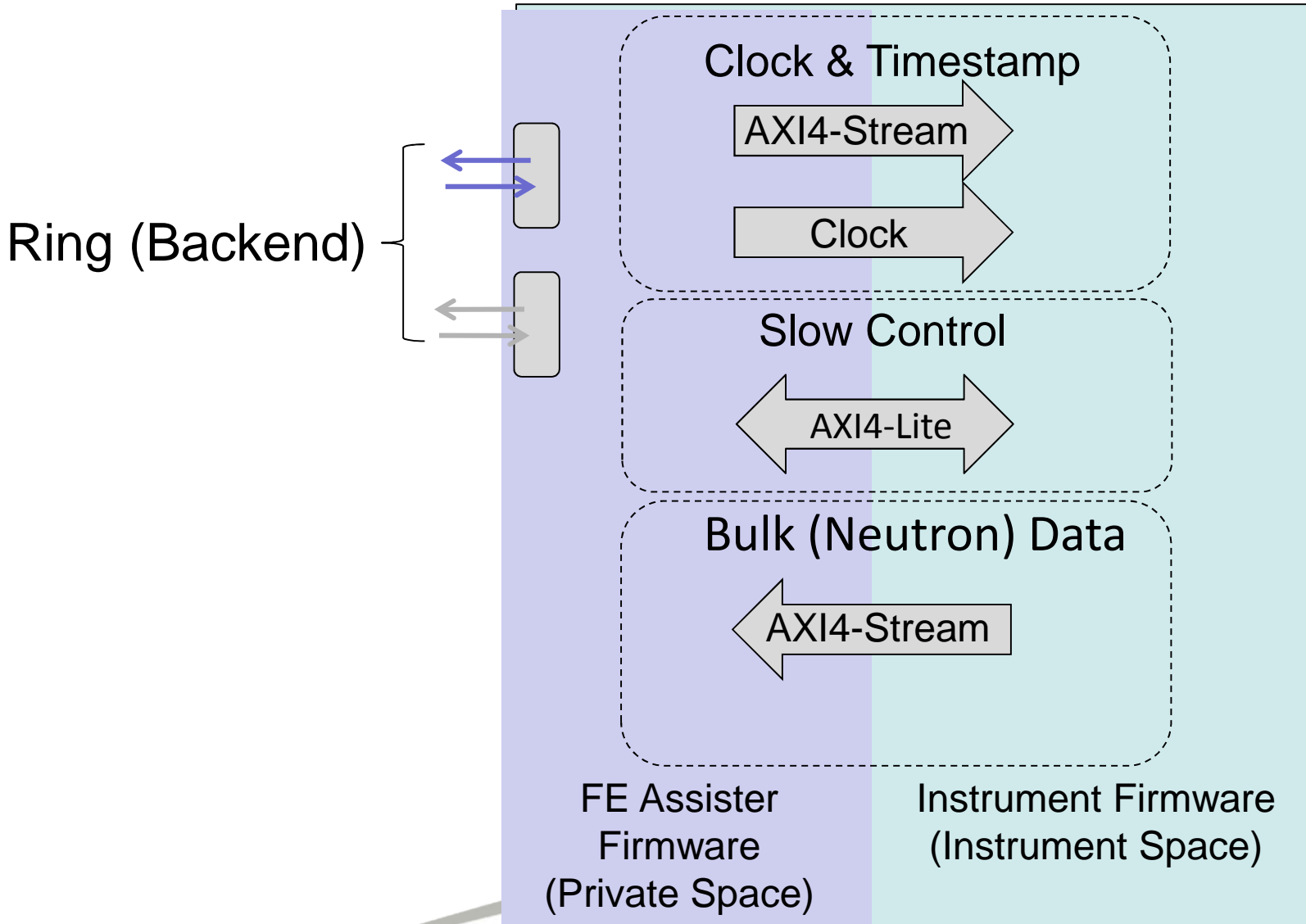


Bulk Data



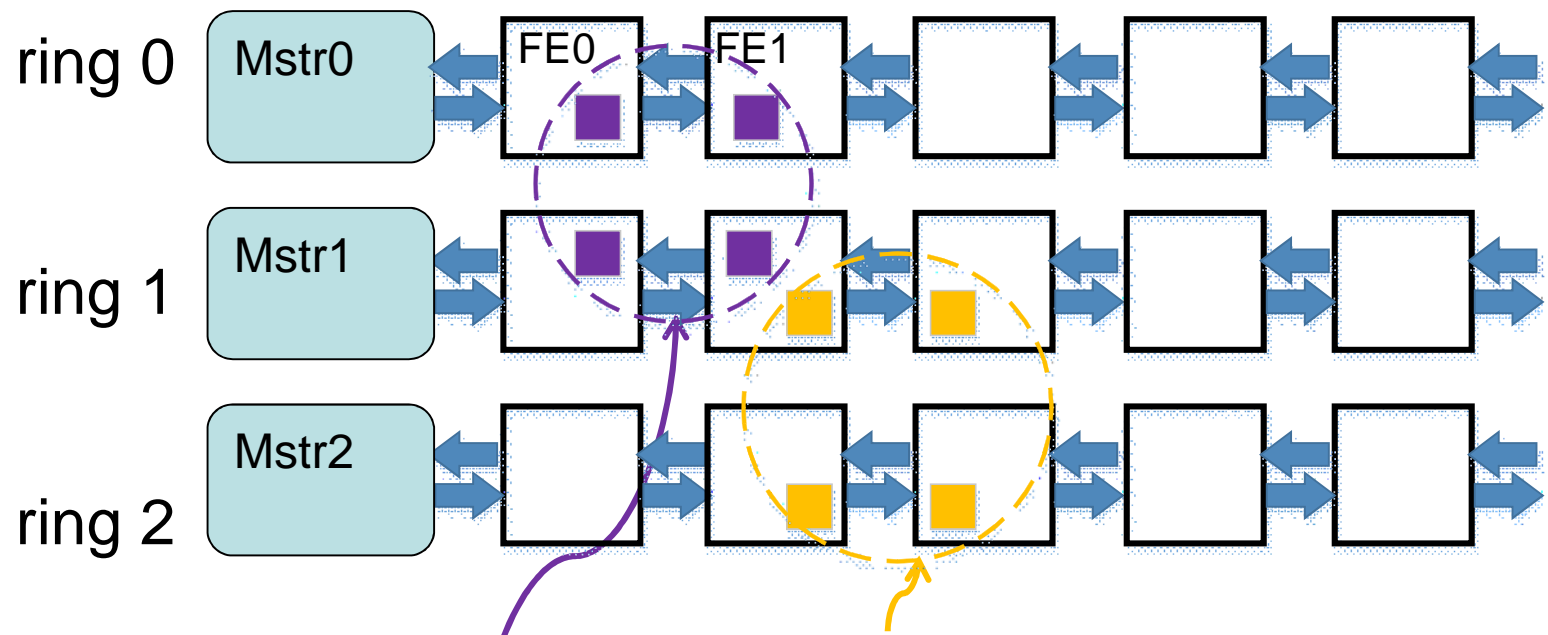


Front End User Interface



New understanding :

Bulk Data Assigned to IP Addr. based on ADC channel



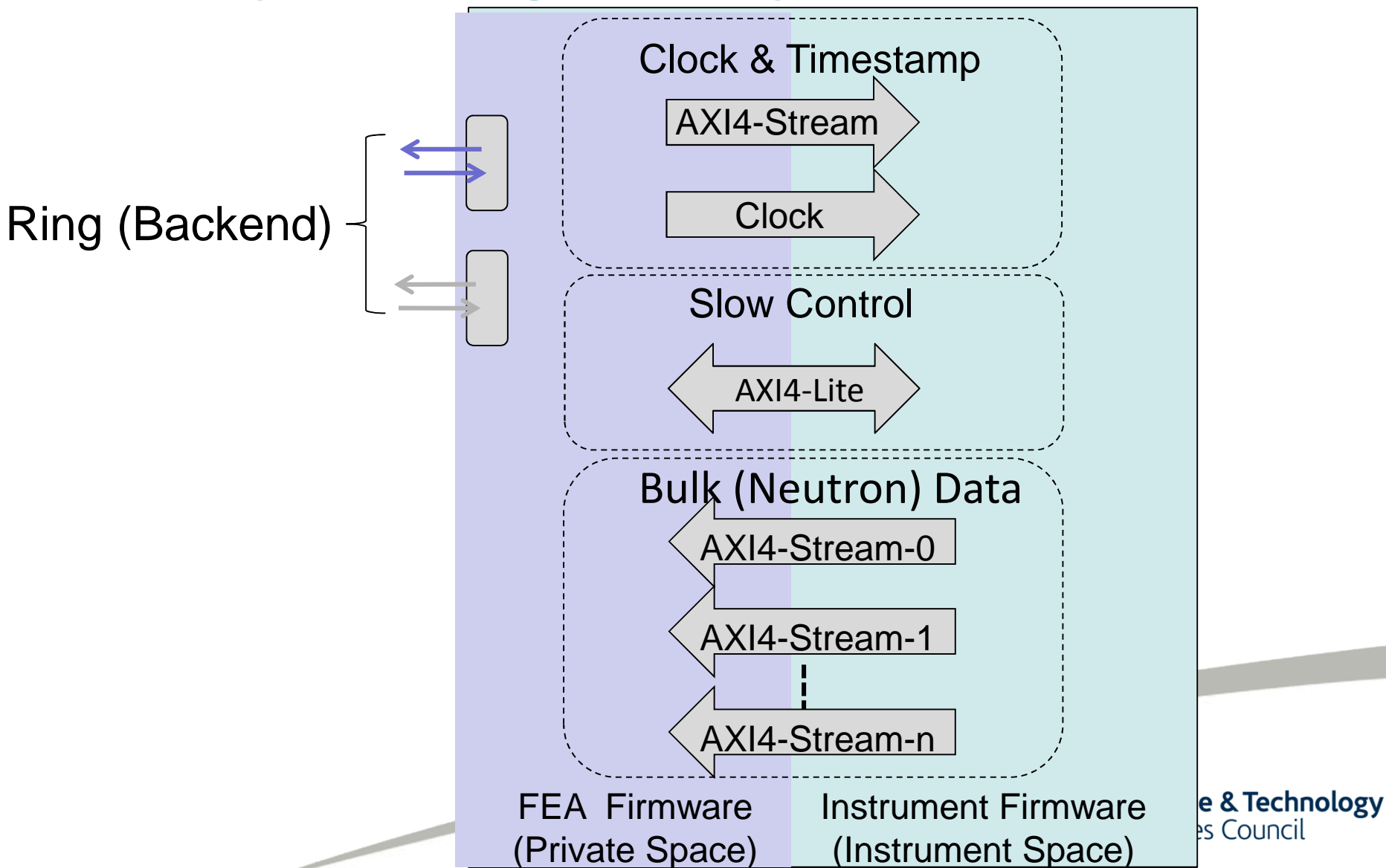
Data from these ADC channels sent to IP addr. 'A'

Data from these ADC channels sent to IP addr. 'B'



EUROPEAN
SPALLATION
SOURCE

New Front End User Interface incorporating multiple data streams



Ring Re-configuration

FE for crossbar

