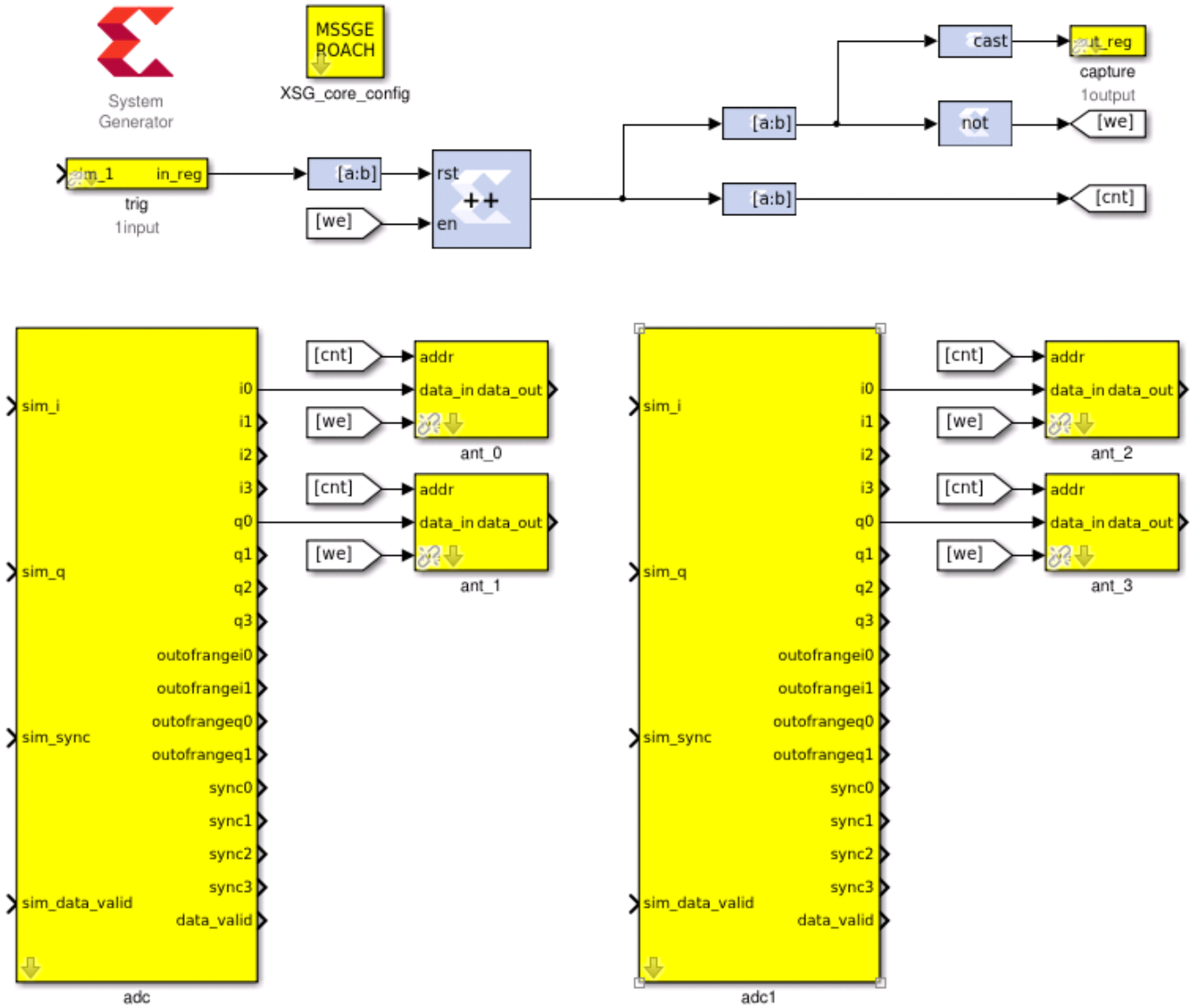
Creating a simple ADC capture on a ROACH

Rachel Domagalski

University of California, Berkeley

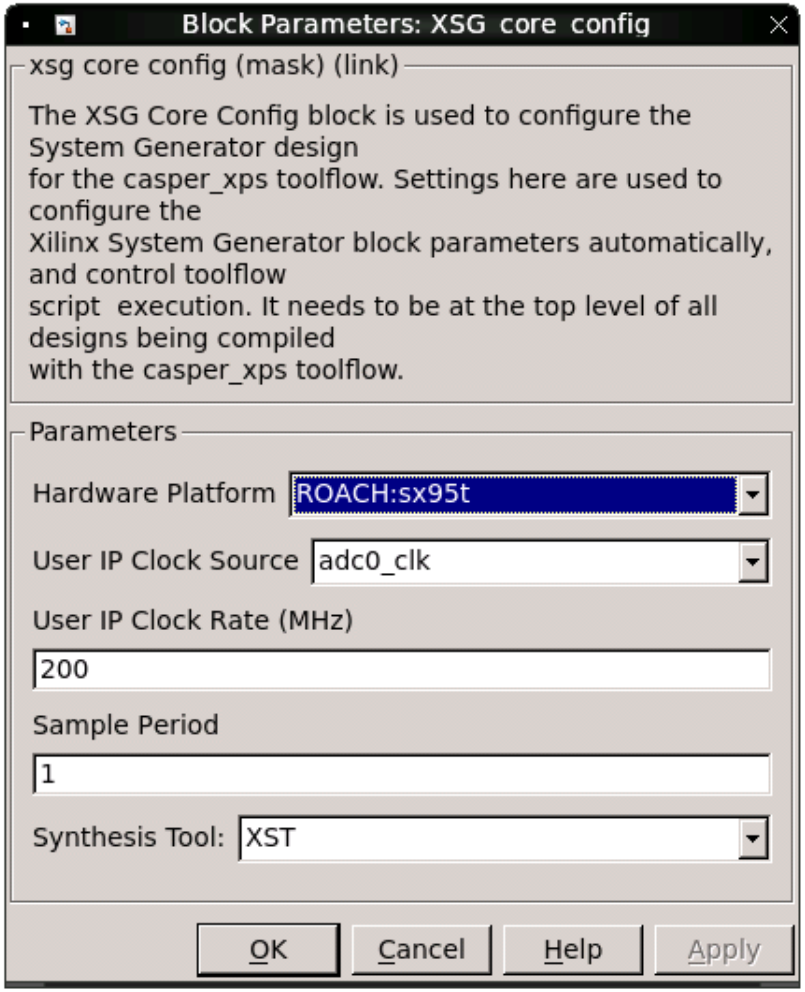
Overview


System Generator
MSSGE ROACH
XSG_core_config
sim_1
in_reg
trig
1input
[a:b]
[we]
rst
++
en
[a:b]
[a:b]
cast
not
ut_reg
capture
1output
[we]
[cnt]
sim_i
sim_q
sim_sync
sim_data_valid
i0
i1
i2
i3
q0
q1
q2
q3
outofrangei0
outofrangei1
outofrangeq0
outofrangeq1
sync0
sync1
sync2
sync3
data_valid
adc
[cnt]
addr
data_in data_out
[we]
ant_0
ant_1
ant_2
ant_3
adc1

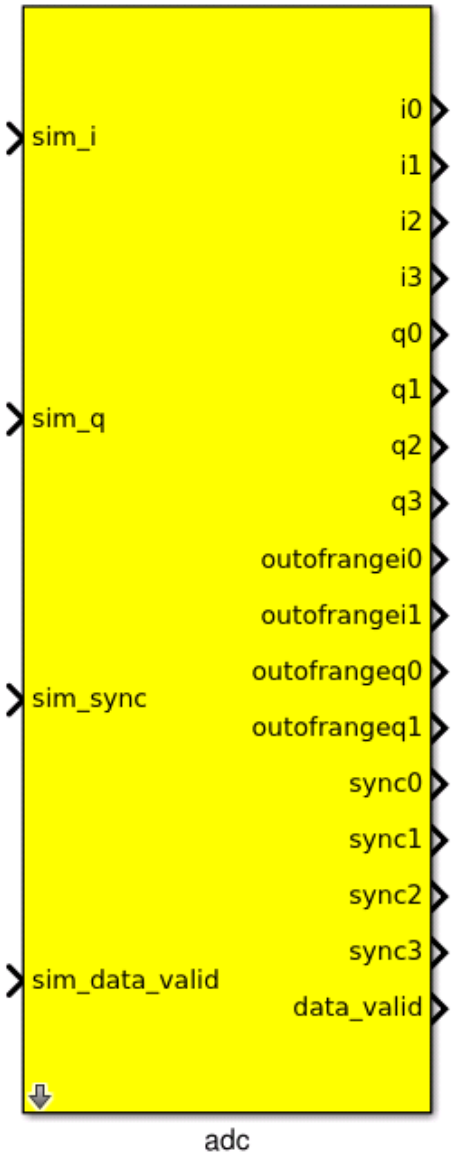
Getting started. . .

System Generator

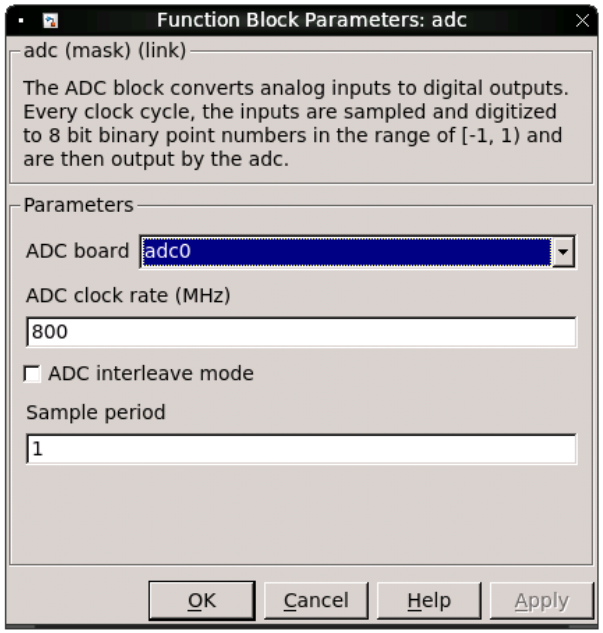
XSG_core_config



The iADC

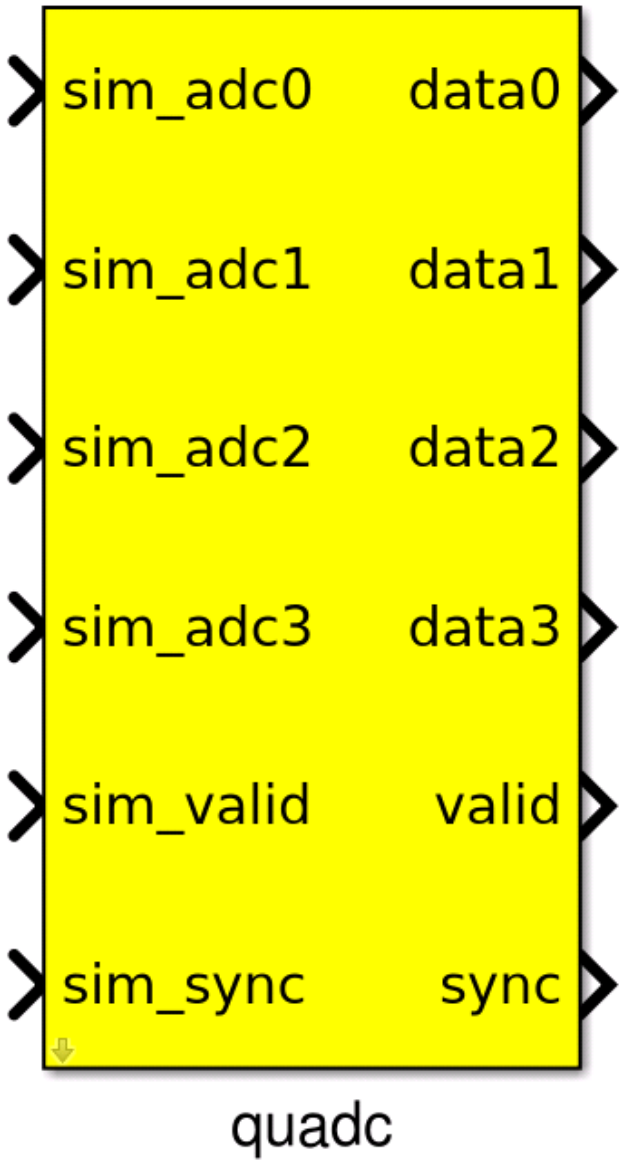


The iADC sample rate is four times the FPGA sampling rate.

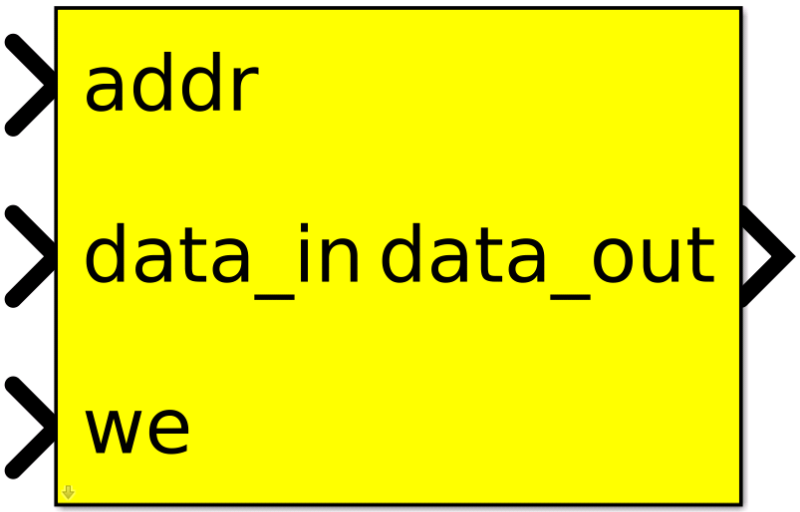


The QuadADC

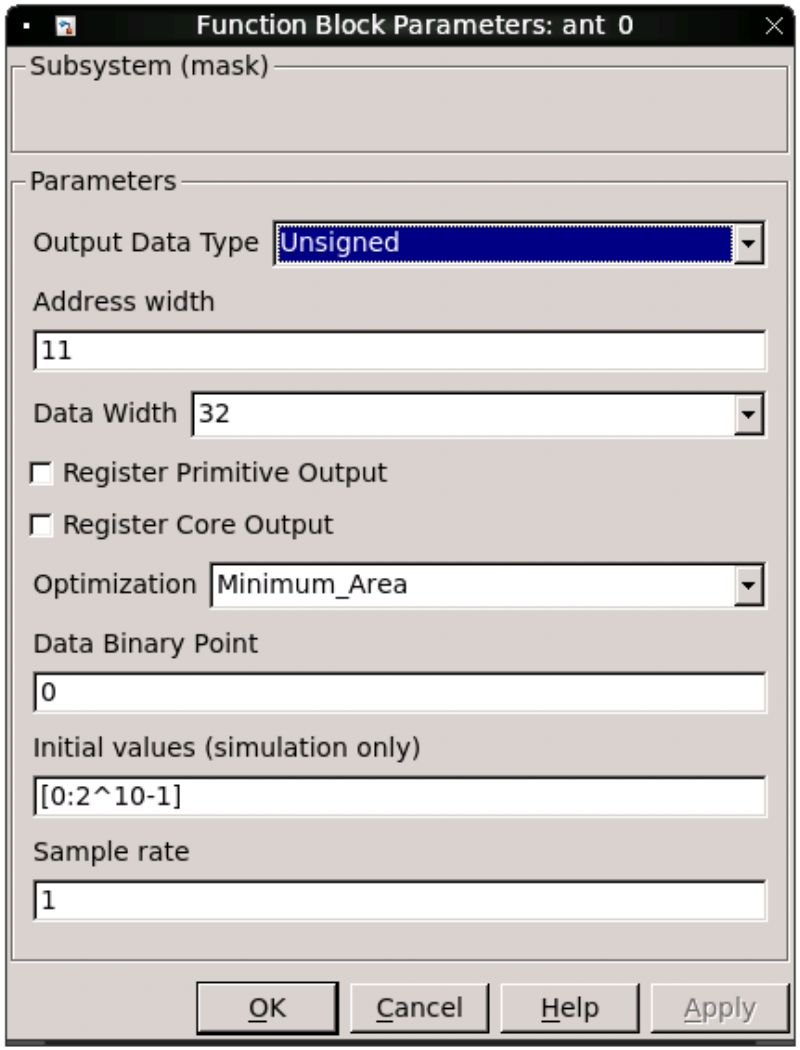
- ADC card that can take up to four signal inputs.
- The QuadADC sampling rate is the same as the FPGA.



Shared BRAM

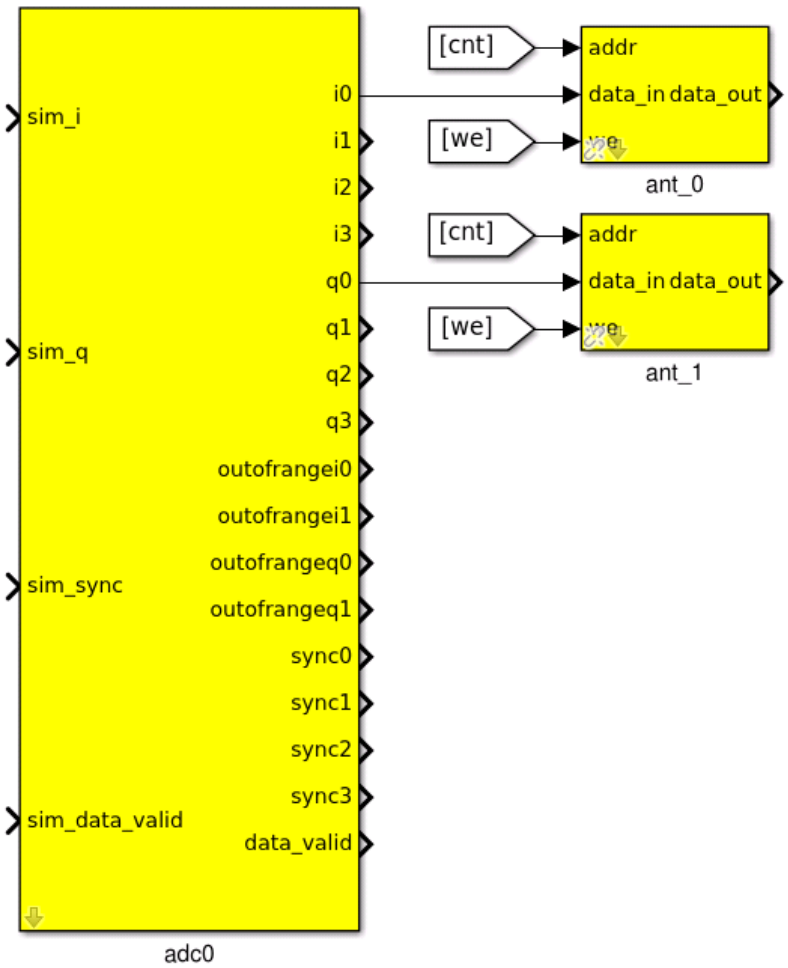


Shared BRAM

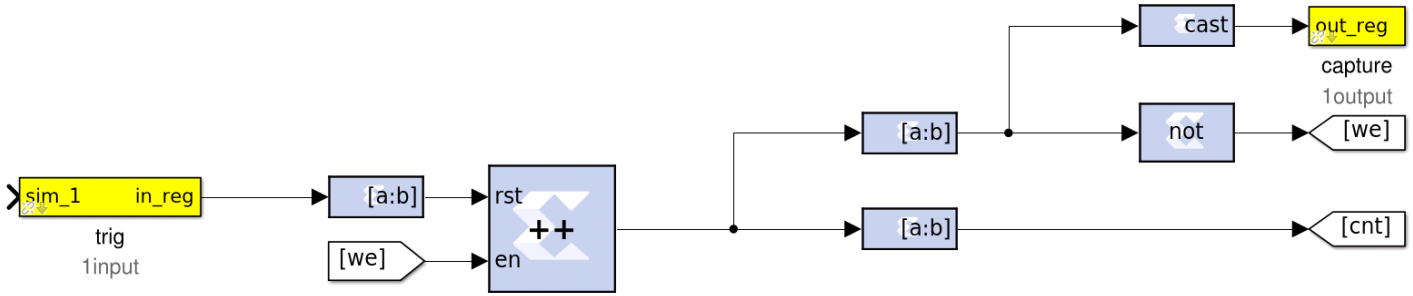


Putting it together...

- This reads out one sample per channel every FPGA clock cycle and stores it in BRAM.
- Data can be read out of the BRAM using KATCP in Python with the `FpgaClient.read()` function.
- Integers are stored in big-endian format and are read out as a sequence of hex strings.



Capturing a sample



- This system generates addresses to store each sample in BRAM and stops writing when all samples are read.
- A sample is captured by successively writing 1 and then 0 to the `trig` software register.
- The counter has one more bit than the BRAM address so that the counter's MSB acts and a write-enable.

The full design