Exercise 10: A2D Intf Design and Test Bench

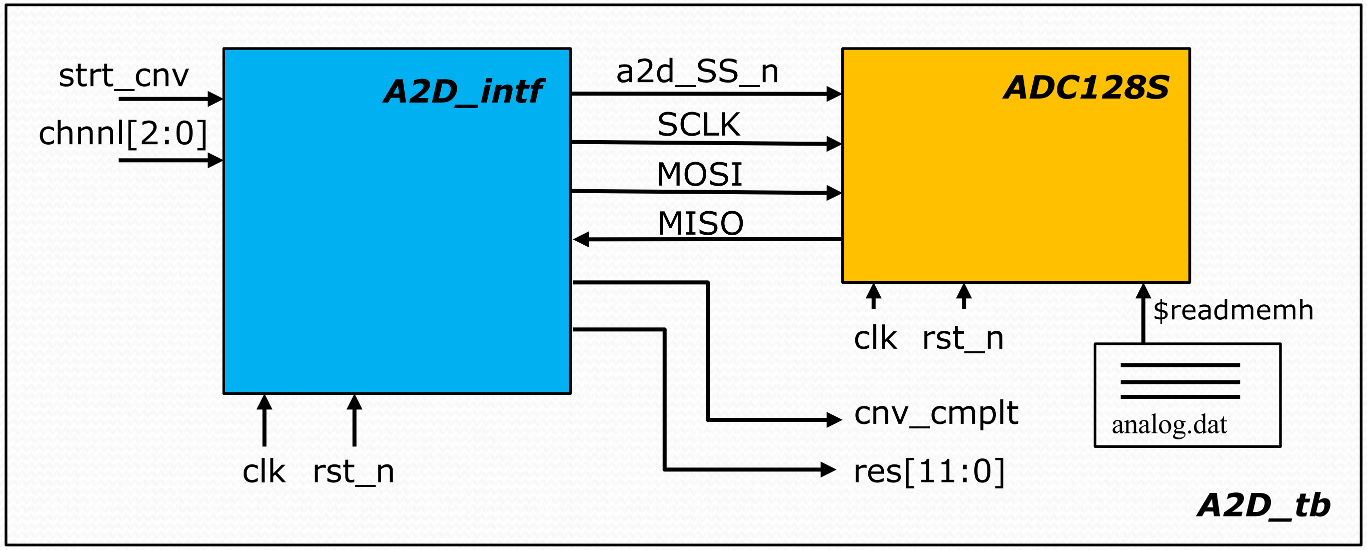
You will be producing a module called **A2D_intf.sv** with the following interface:

Signal:	Dir:	Description:
clk, rst_n	in	clock and asynch active low reset
strt_cnv	In	Asserted for at least one clock cycle to start a conversion
cnv_cmplt	out	Asserted by A2D_intf to indicate the conversion has completed. Should stay asserted till the next **strt_cnv**.
chnnl[2:0]	in	Specifies which A2D channel (0..7) to convert
res[11:0]	out	**1's complement (inversion)** of the 12-bit result from A2D.
a2d_SS_n	out	Active low slave select (part of SPI interface to A2D)
SCLK	out	Serial clock to the A2D
MOSI	out	Master Out Slave In (serial data to the A2D)
MISO	in	Master In Slave Out (serial data from the A2D)

Note the interface is almost the same as the SPI interface in HW 3, except for res[11:0]. So you need to simply instantiate your SPI master and add some peripheral logic.
Why 12-bit? Why 1's complement?

Exercise 10 : A2D Intf Design and Test Bench

A model of the A2D converter is provided on the course website (**ADC128S.sv**). Download this and make a test bench that incorporates your A2D_intf and ADC128S.



ADC128S reads in values that represent the "analog" data from a file called **analog.dat**. An example of this file can also be downloaded from the website.

Exercise 10 : A2D Intf Design and Test Bench

There are lots of ways you can validate your A2D is getting the correct results. That is up to your team. Just remember you need this block to work correctly so don't cut corners.

Regarding sharing the work. Two team members can work on the test bench while the other(s) works on the DUT.

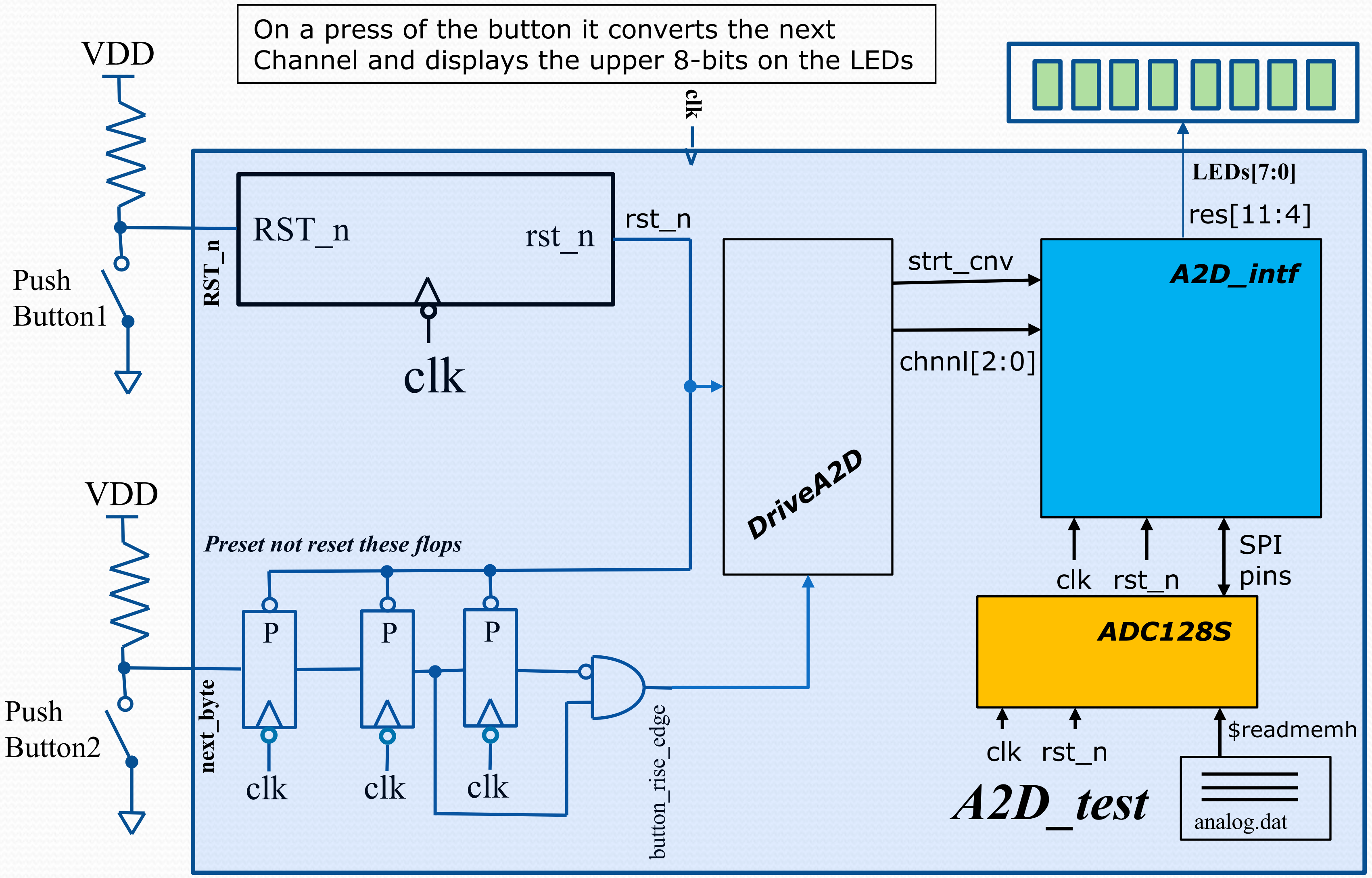
The next section regards mapping your A2D_intf.sv to the DE0. The driver for this (see next slide) is something a team member can work on while others are getting the A2D_intf to work.

One the website under Exercise10 you will find .qpf and .qsf files for the project to map your A2D_intf to the DE0-Nano.

Note: **initial** block in ADC128S.v is **synthesizable** (at least by Quartus). Why? This block actually infers an initialized memory block, which is a piece of hardware.

Exercise 10: Testing Your A2D_intf on DE0-nano

Demo 1: Take a video and upload. (no in-person check-off)



Exercise 10 : A2D Intf Design and Test Bench

Now, we connect to the physical ADC, instead of the ADC model on FPGA.

Remove ADC128S module.

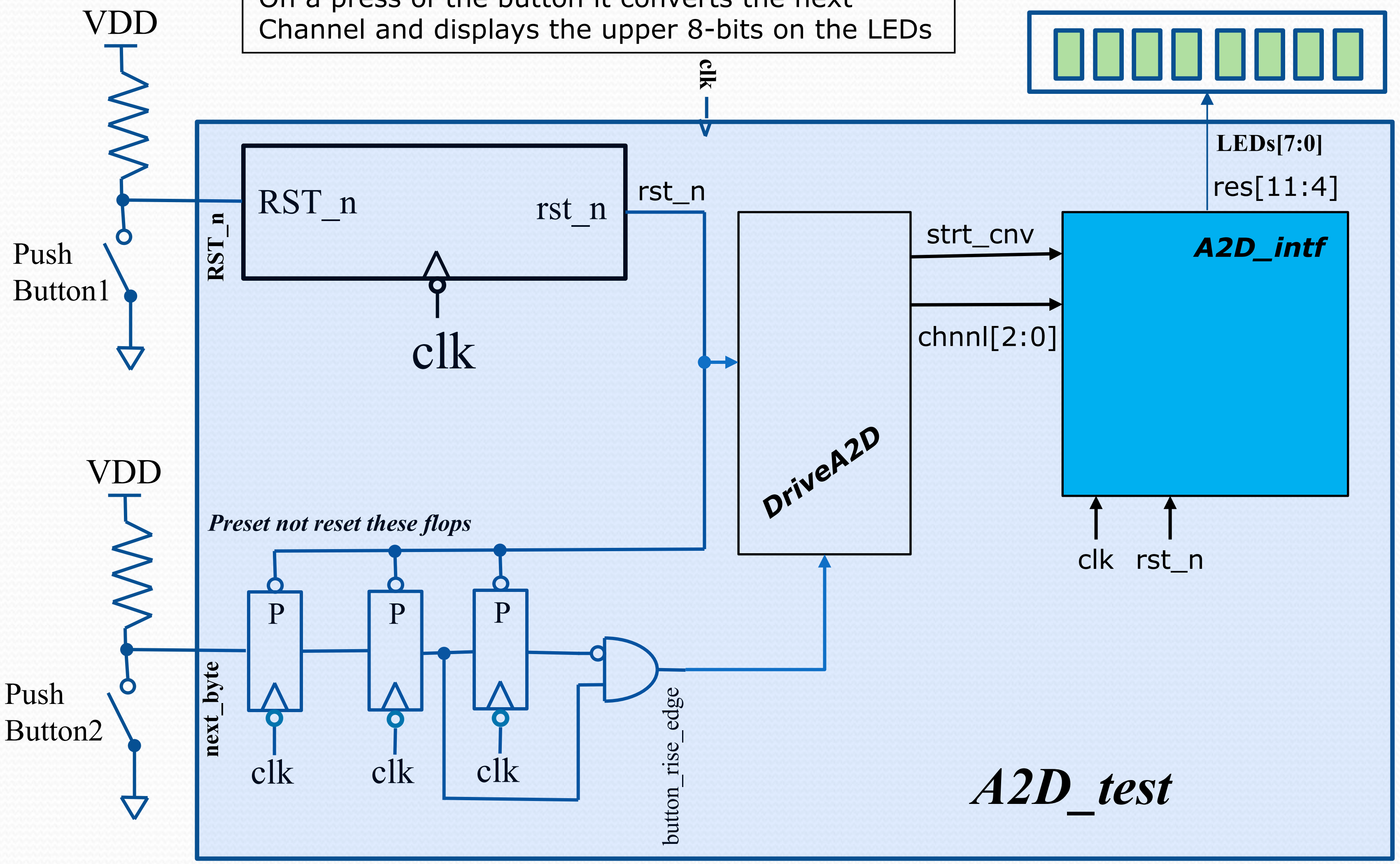
Map SPI ports (SS_N, MOSI, MISO, and SCLK) to FPGA pins in the .sqf file

```
#===========================================================
# ADC
#===========================================================
set_location_assignment PIN_A10 -to a2d_SS_n
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to a2d_SS_n
set_location_assignment PIN_B10 -to MOSI
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to MOSI
set_location_assignment PIN_B14 -to SCLK
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to SCLK
set_location_assignment PIN_A9 -to MISO
set_instance_assignment -name IO_STANDARD "3.3-V LVTTL" -to MISO
```

Exercise 10: Testing Your A2D_intf on DE0-nano

Demo 2: Demo to Prof. Kim/TAs. No video upload available.

On a press of the button it converts the next Channel and displays the upper 8-bits on the LEDs



VDD
Push Button1
RST_n
RST_n
rst_n
clk
clk
rst_n
LEDs[7:0]
res[11:4]
strt_cnv
chnnl[2:0]
A2D_intf
DriveA2D
clk
rst_n
VDD
Preset not reset these flops
P
P
P
Push Button2
next_byte
clk
clk
clk
button_rise_edge
A2D_test