Vivado Automation Script

The Vivado automation script automates the creation of the hardware shell + accelerator.
The hardware shell contains the following IP:

- Zynq UltraScale+ MPSoC Processing System
- AXI DMA
- Processor System Reset Module
- SmartConnect
- Concat
- AXI-4 Stream Data Width Converter (optional)

Script Parameterization

The generator for the Vivado autonation script is parameterized by the following:

- Vivado version: this affects the version of the shell IPs, available FPGA parts and boards, and hardware handoff file format. At this point, we should target Vivado 2019.1 and above
- Accelerator IP location: this needs to be added to the IP repository in Vivado in order to instantiate the IP
- Accelerator IP's VLNV
- (Names of widths of accelerator ports: helps in determining whether to instantiate width converters or not). This can be automatically introspected within the scripts and therefore doesn't need to be a parameter
- handoff location: this is where the output collateral is placed
- project name and path: the name of the Vivado project to create. (Generated bitstream will be named PROJECT_NAME.bit, hardware handoff file will be named PROJECT_NAME.xsa)
- FPGA chip part number and board number

IP Configuration

Zynq UltraScale+ MPSoC Processing System

- apply board presets
- AXI PL-PS interfaces:
  - Master Ports: enable M_AXI_HPM0_FPD only : connects to DMA's AXI-Lite interface (register

space) through a SmartConnect interconnect
    - Slave Ports: enable S_AXI_HPC0_FPD only : connects to DMA's M_AXI_SG, M_AXI_MM2S, and M_AXI_S2MM ports through a SmartConnect interconnect

SmartConnect

There are 2 instances of this IP.

The first should be configured with 1 input and 1 output:

- input connects to M_AXI_HPM0_FPD port on the PS
- output connects to S_AXI_LITE interface on DMA

The second should configured with 3 inputs and 1 output

- one input connects to DMA's M_AXI_SG
- one input connects to DMA's M_AXI_MM2S
- one input connects to DMA's M_AXI_S2MM
- output connects to S_AXI_HPC0_FPD on the PS

Processor System Reset Module

We will have only one clock domain with two reset domains (interconnect and peripheral)

- connect PS's `PL clock0` to `ext. slowest clock` on the reset module
- connect `peripheral aresetn` from reset module to DMA reset and Accelerator's reset
- connect `interconnect aresetn` from reset module to the 2 SmartConnect IPs instantiated

Concat

This should have 2 inputs and 1 output:

- connect one input to DMA mm2s interrupt line
- connect one input to DMA s2mm interrupt line
- connect output to PS interrupt

AXI DMA

- enable scatter-gather

- disable multichannel
- disable stream control signals
- set address width to 64 bits
- set SG length width to 26 bits
- enable S2MM (output) and MM2S (input) channels
- set both channels to 32-bit wide for data

AXI-4 Stream Data Width Converter

DMA channels are set up with 32-bit wide stream interfaces. When an accelerator port has a data width smaller than 32 bits, we need to instantiate a stream data width converter between that port and the DMA port to which it needs to be connected.

Accelerator IP

The accelerator IP should be connected as follows:

- ap_clk to `PL clk0` from the PS
- ap_resetn to `peripheral_aresetn` from reset module
- input axi stream to DMA's `MM2S` Stream port
- output axi stream to DMA's `S2MM` Stream port

Automation Script Interface

The automation script should be able to accept a configuration file that provides all the necessary information.
The configuration file is a JSON file of the form:

```
{
  "config" : {
    "version"             : "config schema version in the form x.y",
    "name"                : "configuration name. Use as prefix for project name in Xi
    "xcel_ip_vlnv"        : "VLNV used for IP instantiation in Vivado",
    "xcel_ip_inputs"      : [
      { "name" : "name of input port 0", "width" : W_IN_0 },
      { "name" : "name of input port 1", "width" : W_IN_1 },
      ...
      { "name" : "name of input port N", "width" : W_IN_N },
    ],
    "xcel_ip_output"      : { "name" : "name of output port", "width" : W_OUT },
    "xcel_rdai_vlnv"      : "VLNV used for RDAI tagging",
    "xlnx_chip_part"      : "part number for the FPGA chip",
    "xlnx_board_part"     : "board part number",
    "vivado_user_ip_repo" : "path to user IP repo",
    "vivado_version"      : "some_version",
    "vivado_handoff_dir"  : "path for directory where to put output collateral",
  }
}
```

NOTE:

- the `xcel_rdai_vlnv` property is not used in Vivado. It is present in the configuration because the same configuration file is used by another script to generate DT overlay and packaged bitstream files
- we currently support only accelerators with 1 input and 1 output
- all paths within the generated script should be absolute paths

The automation script should have the following interface:

```
$> SCRIPT -c config.json -o <VIVADO_SCRIPT_NAME>.tcl
```

The automation script should always:

- return 0 on success
- return any other value on failure

Implementation Language: this is left to the developer. Although, I would encourage using a mainstream scripting language such as Python