Tcl scripting to run Vivado

We will reuse the design that we used for the first class assignment.

A design that performs Multiply and accumulate (MAC) is provided to you along with the testbench.
1. Create a new Vivado project with this design and testbench. **As you perform each step in Vivado, note down its equivalent command that created in the Tcl console in a text file.**

For example

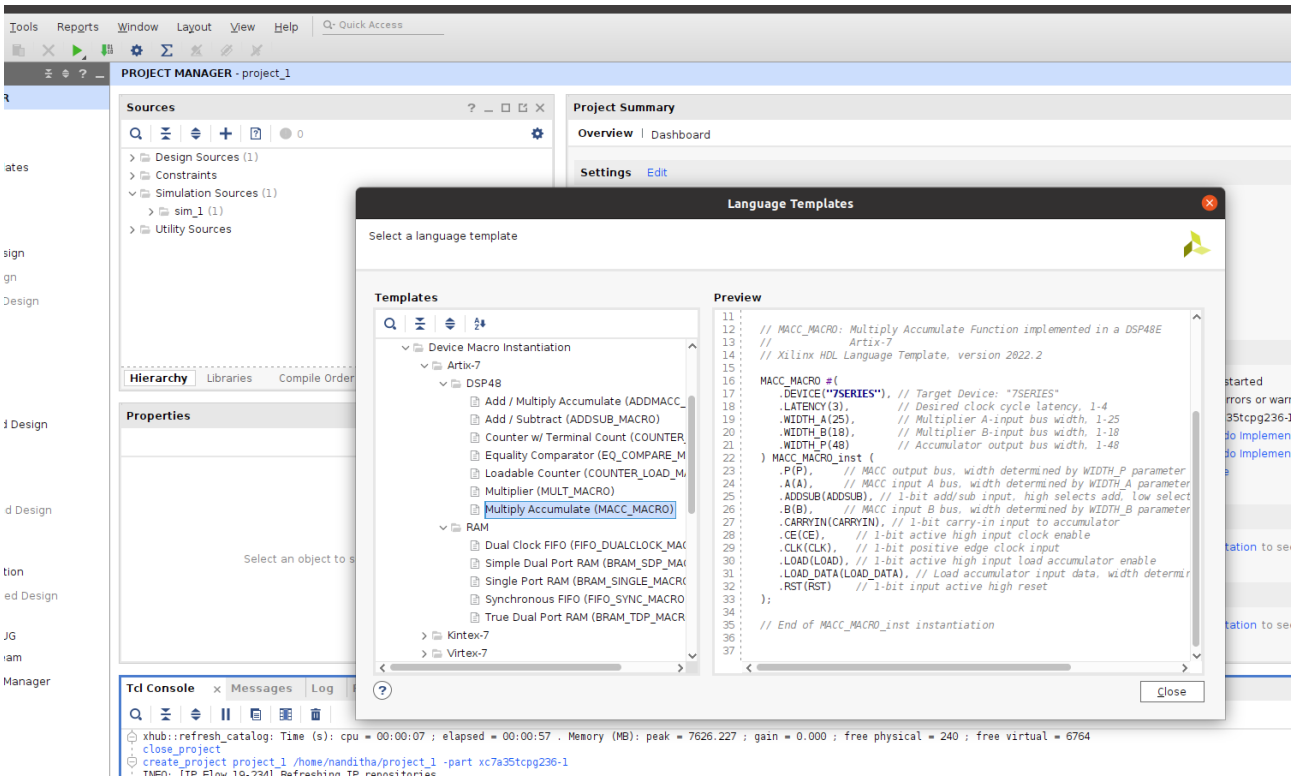
- Create a new RTL Project will create the following command. In other words, running the following command will perform the same task.
  create_project project_1 /home/user/project_1 -part xc7a35tcpg236-1
- Add Sources results in the following command
  import_files -norecurse /home/user/memory.v
  update_compile_order -fileset sources_1\

After running all steps manually, use the commands in the Tcl script to run Vivado on the same design instead of using the GUI in Vivado. Run all steps until post-implementation simulation and bitstream and verify the result.

vivado -mode tcl -source <run.tcl>

2. Now, let us create a DSP Macro without using the IP Catalog. In Vivado, go to Tools --> Language Templates and choose the following option:

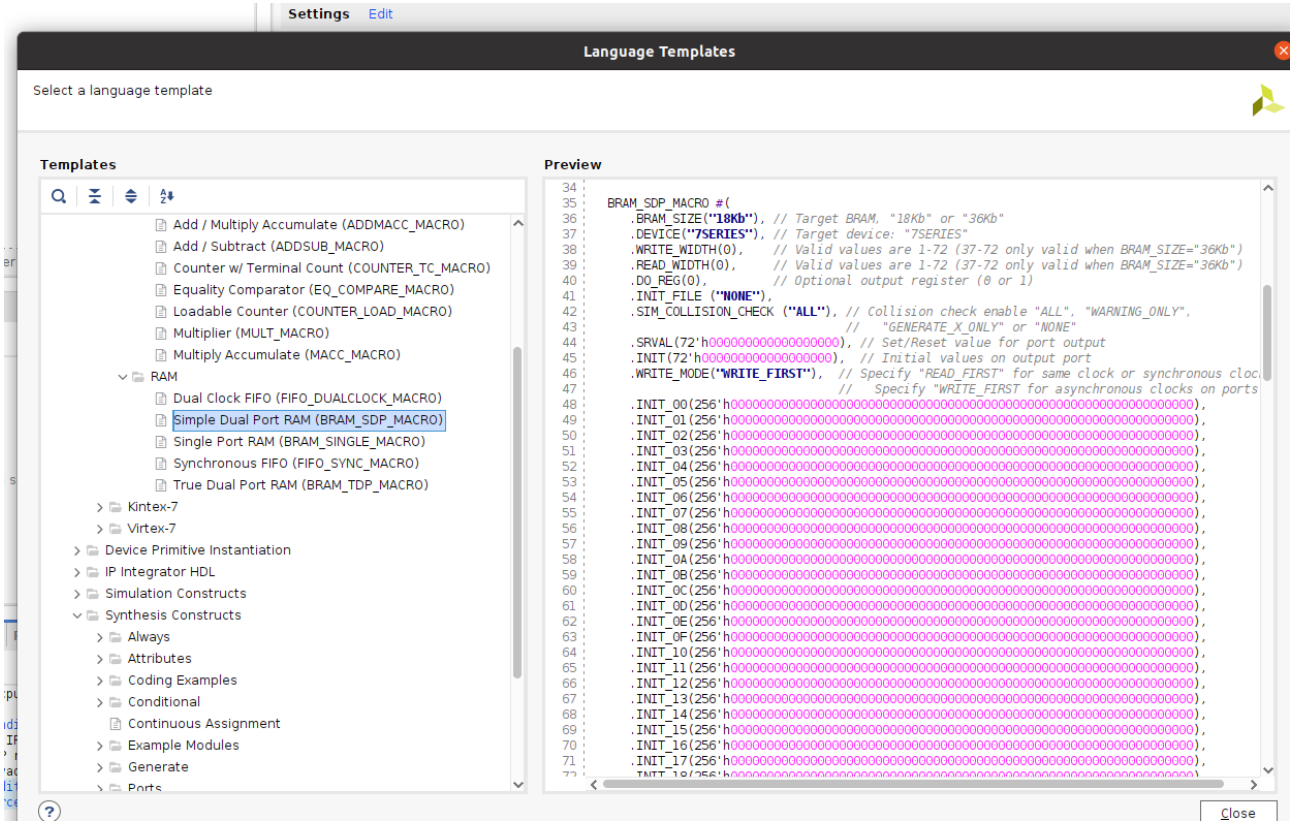
Verilog --> Device Macro Instantiation --> Artix-7--> DSP --> MAC
Note down all the tcl equivalent commands. Copy this macro into the verilog file. (This step will not create any tcl command). Now, simulate this Verilog file using tcl commands instead of the mac.v that was provided to you.



Note that the Verilog code generation itself cannot be automated using tcl scripting (at least for now). We can automate the Vivado flow without having to use the GUI.
It is recommended that the designer writes the Verilog code with these macros, and use tcl to run Vivado.

3. Try the DSP IP Catalog to create a DSP instance. This will create a tcl command called “create_ip”. Copy paste the veo contents into the Verilog code. Now, you will realise that the DSP MACRO using Language template is equivalent to “Creating DSP using IP Catalog + veo code”. Simulate this using the tcl script.

Similar to DSP, Block RAM Macro can also be created using this template. Try this out on your own!


Settings
Edit
Language Templates
Select a language template
Templates
Add / Multiply Accumulate (ADDMACC_MACRO)
Add / Subtract (ADDSUB_MACRO)
Counter w/ Terminal Count (COUNTER_TC_MACRO)
Equality Comparator (EQ_COMPARE_MACRO)
Loadable Counter (COUNTER_LOAD_MACRO)
Multiplier (MULT_MACRO)
Multiply Accumulate (MACC_MACRO)
RAM
Dual Clock FIFO (FIFO_DUALCLOCK_MACRO)
Simple Dual Port RAM (BRAM_SDP_MACRO)
Single Port RAM (BRAM_SINGLE_MACRO)
Synchronous FIFO (FIFO_SYNC_MACRO)
True Dual Port RAM (BRAM_TDP_MACRO)
Kintex-7
Virtex-7
Device Primitive Instantiation
IP Integrator HDL
Simulation Constructs
Synthesis Constructs
Always
Attributes
Coding Examples
Conditional
Continuous Assignment
Example Modules
Generate
Ports
Preview
BRAM_SDP_MACRO #(
.BRAM_SIZE("18Kb"), // Target BRAM, "18Kb" or "36Kb"
.DEVICE("7SERIES"), // Target device: "7SERIES"
.WRITE_WIDTH(0), // Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
.READ_WIDTH(0), // Valid values are 1-72 (37-72 only valid when BRAM_SIZE="36Kb")
.DO_REG(0), // Optional output register (0 or 1)
.INIT_FILE ("NONE"),
.SIM_COLLISION_CHECK ("ALL"), // Collision check enable "ALL", "WARNING_ONLY",
// "GENERATE_X_ONLY" or "NONE"
.SRVAL(72'h000000000000000000), // Set/Reset value for port output
.INIT(72'h000000000000000000), // Initial values on output port
.WRITE_MODE("WRITE_FIRST"), // Specify "READ_FIRST" for same clock or synchronous cloc
// Specify "WRITE_FIRST for asynchronous clocks on ports
.INIT_00(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_01(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_02(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_03(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_04(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_05(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_06(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_07(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_08(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_09(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_0A(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_0B(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_0C(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_0D(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_0E(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_0F(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_10(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_11(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_12(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_13(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_14(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_15(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_16(256'h0000000000000000000000000000000000000000000000000000000000000000),
.INIT_17(256'h0000000000000000000000000000000000000000000000000000000000000000),
Close