Question:
Suppose there are two available modules for AES encryption and floating-point multiplication. Design a system which include one of them at once. But the system can switch to the another module during run-time based on partial reconfiguration mechanism.

Solution:

1- I found these two VHDL modules in internet:

1-1 A 32Bits floating-point multiplier [1]. It is a computational circuit with two inputs and one outputs and all are 32 bits. The VHDL code is in the '/Sources/hdl/mult32_float/mult32_float.vhd' file.

1-2 A 128Bits AES Encryption module [2]. It is a synchronous sequential circuit. It has a 'clk' input port, a 'reset' trigger input and a 'done' flag output. It has also two inputs (key, plaintext) and one output (ciphertext) which are 128Bits. It is in the '/Sources/hdl/aes128_enc/' folder.

2- I modified these two modules to have identical ports (inputs and outputs) with the same size,type and name. This should be done to make these two modules interchangeable. In practice these extra ports of the multiplier module are not using and this module just using the first 32Bits of the input/output data.

3- I initially tested the multiplier module by a test bench was available with the behavioral simulation. But to complete the testing, I tested both of modules simultaneously on the hardware using Xilinx VIO IP core. This test system is available in the 'Two_modules_hardware_test' folder. The device I used is an Arty z7-20 board which has a series 7 Zynq Soc which the part number is xc7z020clg400-1. But it is not dependent to this special FPGA. The only file you need to modify is the constraint file in the which define the clock. I also using Vivado 2019.1

3-1 In this test system I wrote a top module and I added a VIO module with two input (128,1 Bits) and three outputs(128,128,1 Bits) which drive the inputs of two above modules and read the modules' output. You can also trig the reset port of modules. Both modules has the same input but the outputs are separated. You can test the modules after implementation and programming the device.
An example for multiplier: 40200000 (2.5) * C11B3333 (-9.7) = C1C20000 (-24.25)
If you also turn on the reset, the AES module output start to change.

3-2 VIO module is just for the test then it can be removed later and the ports can connected to I/O pins or connect to IP cores which convert serial protocols inputs/outpus (like AXI) to parallel ports of modules.

Setup for partial reconfiguration

4- I changed the two modules entity name to 'function128'.

5- Making the 'top' module's checkpoint

5-1 I wrote a top module with one ‘function128’ module as a black box and VIO ip for testing. It is in the folder ‘top’

5-2 I synthesize it and made a checkpoint, ‘top_synth.dcp’

6- Making the ‘aes128_enc’ module’s checkpoint

6-1 I synthesized the module in the out_of_context mode and made a checkpoint, ‘rm1_synth.dcp’ . The project is in ‘aes128_enc’ folder.

7- Making the ‘mult32_float’ module’s checkpoint

7-1 I synthesized the module in the out_of_context mode and made a checkpoint, ‘rm2_synth.dcp’ . The project is in ‘mult32_float’ folder.

8- I followed the instructions in the UG947 and UG909 manual [3, 4] and I modified them to match my setting to make full and partial bitstream files.

8-1- Two full bitstream files including one of these two modules,
‘Config_aes.bit’ and ‘Config_mult.bit’ and I made two partial bit files for each module,
‘Config_aes_pblock_inst_function128_partial.bit’ and
‘Config_mult_pblock_inst_function128_partial.bit’ files.

8-2- I put this instruction in ‘instruction_bit.txt’ file.

8-3 Extra files with valuable information are also generated during following the instruction which you can find in the root folder.

Testing the partial reconfiguration

9- Program the hardware with one of full bitstream file.

9-1 You need ‘probes.ltx’ file to test the device with VIO. This file is in the root folder.

10 – You can program the device with one of partial bitstream files and you can also check the device.

11- The test was successful my device.

Using Xilinx PR controller IP core – To be done in future

12- The bit files are ready but these steps is needed to be available for PR controller:

12-1 Writing bit files in SD card.

12-2 Using Zynq processor to read data from SD card and send them to DDR3 memory.

12-3 Connecting DDR3 memory to PR controller using AXI-HP bus available in Zynq.

Some comments:

1- I didn’t checked if these two modules are efficient enough.

2- It was my first time to work with partial configuration then it was so challenges for me to understand and implement it. Then this practice was more about knowing better the hardware and its regions and how to work with Vivado and tcl commands and how to define Pblocks and not so much VHDL code was required.

3- I like it.

References:

1- https://github.com/avirlrma/Floating-Point-Multiplier-32-bit/blob/master/Module.vhd
2- https://github.com/hadipourh/AES-VHDL/tree/master/AES-ENC/RTL
3- Vivado Design Suite Tutorial, Partial Reconfiguration, UG947
4- Vivado Design Suite User Guide, Partial Reconfiguration, UG909
5- Conceptual Design and Realization of a Dynamic Partial Reconfiguration Extension of an Existing Soft-Core Processor, Philipp Kerling
6- https://forums.xilinx.com/t5/Vivado-TCL-Community/Partial-reconfiguration-black-box-with-no-cells-after-synthesis/td-p/800692