

Vitis Code Tutorial

Ahmet Can Mert <u>ahmet.mert@iaik.tugraz.at</u>



Overview

• Structure of Vitis_code:

```
main.cc
                      : Main file calling test function.
  lscript.ld
                      : It sets the stack/heap size of the processor. Do NOT modify.
  README.txt
  Xilinx.spec
                      : -
— communication.c : It has FPGA-CPU communication related functions. You do NOT have to make any changes on this file.
  communication.h
                      : -
  instruction.c
                      : It has functions for generating INS array (i.e., instructions encoded with address, OPCODE etc.) for different operations.
  instruction.h
                      : -
  platform.c
                      : It has FPGA platform related functions. You do NOT have to make any changes on this file.
  platform.h
                      : -
  platform_config.h: -
  pke
       modular_arithmetic.cc
       modular_arithmetic.h
       pke parameters.h
       poly arithmetic2.c
       poly arithmetic2.h
       randombytes.c
       randombytes.h
       test polymul.c
       test polymul.h
```

main.cc

 This is the main file that Vitis runs. It performs some initilizations first and then calls the function test_polymul() which is defined in test_polymul.c/test_polymul.h in pke folder. You can keep this file the same and just modify test_polymul().

<pre>init_platform(); //axi_address_base = (uint32_t *) 0x00A0000000; // For zcu102 board axi_address_base = (uint32_t *) 0x40000000; // For PYNQ-z2 board</pre>
<pre>int TEST_TYPE = 0; // 0: TRNG, 1:AES, 2:PKE, 3:End</pre>
printf("************************************
<pre>while(TEST_TYPE !=3){ if(TEST_TYPE==0) test_polymul(); else break;</pre>
<pre>printf("Type of test [0: PolyMul, 3:End] : "); scanf("%d", &TEST_TYPE); }</pre>

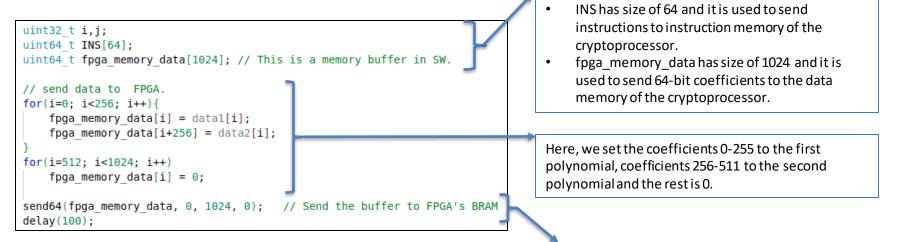
pke/test_polymul.c

- It has two functions: test_polymul() and poly_mult_HW().
 - test_polymul() function generates two polynomials of size 256, a and b, and then performs coefficient-wise multiplication of a and b using HW and SW. HW operation is performed by poly_mult_HW() function.
 - poly_mult_HW() function presents an example for executing operations on FPGA. It performs the following steps:
 - Sends data to FPGA
 - Encodes instructions for the cryptoprocessor and sends instructions to FPGA
 - Executes the instructions on FPGA
 - Reads data back from the FPGA

For the Task-3 of your assignment, you can either modify poly_mult_HW() or write a new function similar to poly_mult_HW().

poly_mult_HW() function

- Now, we'll look into poly_mult_HW() function in detail.
 - Sending data to FPGA.



send64(uint64_t *p, uint32_t base_address, uint32_t num_words, uint32_t INS_flag);

function (defined in communication.c) sends data to the FPGA from CPU. It takes 4 inputs.

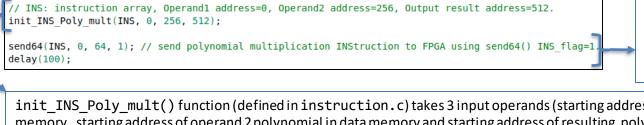
- *p: array of 64-bit data that you want to send to FPGA
- $\bullet \quad base_address: starting write address for data/instruction memory of the cryptoprocessor$
- num_words: number of 64-bit data (word) that you want to send to FPGA
- INS_flag: If this is 0, you are sending coefficients to the data memory of the cryptoprocessor If this is 1, you are sending instructions to the instruction memory if the cryptoprocessor

For this example, we're sending 1024 words in fpga_memory_array to the data memory of the cryptoprocessor starting from address 0 of the data memory of the cryptoprocessor.

We define two arrays of 64-bit data.

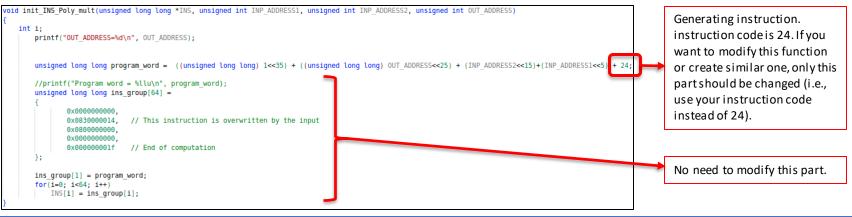
poly_mult_HW() function

- Now, we'll look into poly_mult_HW() function in detail.
 - Encoding instructions for the cryptoprocessor and sends instructions to FPGA.



For this example, we're sending whole INS array to the program memory of the cryptoprocessor starting from address 0 of the program memory. The content of INS is already set by init_INS_Poly_mult().

init_INS_Poly_mult() function (defined in instruction.c) takes 3 input operands (starting address of operand 1 polynomial in data memory, starting address of operand 2 polynomial in data memory and starting address of resulting polynomial in data memory), encodes these operands with instruction code=24. Please note that this function is specific to the instruction instruction code =24. You can create a similar function for every instruction (and instruction code) that you define. In instruction.c, you can find some other examples as well.



poly_mult_HW() function

- Now, we'll look into poly_mult_HW() function in detail.
 - Executing instructions on the FPGA and reading data back to CPU.

```
exeIns(); // Now ask the FPGA to compute the instruction (i.e., polynomial multiplication)
delay(100);
```

receive64(fpga_memory_data, 0, 1024); // Read the BRAM of FPGA into the SW-side buffer delay(100);

exeIns() function(defined in communication.c) sends commands to the FPGA for performing the instructions. You do NOT have to modify this function.

receive64(uint64_t *p, uint32_t base_address, uint32_t num_words); function reads data from the data memory of the cryptoprocessor.

It takes 3 inputs:

- *p: array of 64-bit data that will store the incoming data from FPGA
- base_address: starting read address for data memory of the cryptoprocessor
- num_words: number of 64-bit data (word) that you want to read from the FPGA

For this example, it reads 1024 coefficients from data memory of the cryptoprocessor into the array fpga_data_memory, starting from address 0 of the data memory of the cryptoprocessor.

Other files in pke folder

- In pke folder, we provide some functions that you can use.
 - pke_parameters.h: You can ignore this file.
 - poly_arithmetic2.c: Empty file, just provides template for implementing schoolbook method.
 - randombytes.c: Includes a function to generate random byte.
 - modular_arithmetic.cc: Includes field arithmetic functions (integer addition/subtraction and modular multiplier) for 64-bit integers. Note that modular multiplier method is NOT using Montgomery method.
- You can use these functions (or you can write your own functions) to implement schoolbook/Karatsuba functions (in case you want to implement this approach and perform a part of operation in SW).