NTNU

RSA Integration kit 2025

Designed for use in TFE4141. This kit includes all necessary files for implementing and testing a RSA encryption circuit on the Pynq Z1 as provided by NTNU.

Getting started

- If you have not yet installed vivado, it can be downloaded from: Xilinx.com/download. Install vivado design suit HLx edition (requires user).
- Install the pynq board files in vivado. Default location in widows is `C:/Xilinx/Vivado/2024.1/`. Download board files and extract to `Vivado_installation_directory/data/boards/board_files/`.
    - This is also necessary to do on the lab computers. They use the default location.
- Start by downloading RSA integration kit and extracting the `.zip` file to a desired location. The path of the location may not include whitespace or special characters such as `øæå`. This location is henceforth referred to as `PROJECT_FOLDER`.
- After the project structure is extracted, then open up `vivado 2024.1`.
- After starting vivado, click `Tools -> Run TCL script`. In the window that pops up, navigate to the directory you extracted the project folders to and select `regenerate_projects.tcl`.
- This will open other windows which will close. Wait for the process to finish (Marked by `Done` appearing in the Tcl console). This should take approximately a minute.
- By now the projects `rsa_soc`, `rsa_accelerator`, `exponentiation` should have appeared in recent projects on the right side on the main window. If they have not, then they can be opened from
    - `rsa_soc` is located in `PROJECT_FOLDER/RSA_soc/RSA_soc/RSA_soc.xpr`.
    - `rsa_accelerator` is located in `PROJECT_FOLDER/RSA_accelerator/RSA_accelerator/rsa_accelerator.xpr`.
    - `exponentiation` is located in `PROJECT_FOLDER/Exponentiation/Exponentiation/Exponentiation.xpr`.
- At this point everything is set up correctly and you may start working on the project.

Content

This folder structure contains 3 vivado projects. You may add more as you see fit, but it is not necessary. When this project says run script, it means `Tools->Run tcl script`.

- `Master_constraints/` contains the constraints file for all the projects.

    - `PYNQ-Z1_C.xdc` is the constraint for the Pynq-z1 platform. By default it does nothing, but any constraints can be added to it.
- `Bitfiles/` contains the bitfiles that are generated when the project is synthesized.
    - `rsa_soc.bit` is the configuration for the programmable logic substrate for the soc.
    - `rsa_soc.hwh` is the configuration that the jupyter notebook uses to access the rsa accelerator.
- `Reports/` contains the report from the synthesized designs.
    - `placed_design_timing_summary.txt` contains the timing report for the whole design as placed on the fpga.
    - `placed_design_utilization.txt` contains the utilization report for the whole design as placed on the fpga.
    - `rsa_accelerator_utilization.txt` contains the utilization report for the rsa_accelerator Ip.
- `RSA_soc/` contains all the necessary component for the project to be synthesized and the bitstream generated.
- `RSA_accelerator/` should contain the whole RSA core, include the infrastructure for the AXI streams. All sources from `Exponentiation` should be automatically included in this, but you may have to regenerate the project. The top level design `rsa_accelerator.vhd` is configured to `VHDL`, whilst everything else is configured to `VHDL 2008`. This project will be compiled as an IP which is included in `RSA_soc`.
- `Exponentiation/` should contain the circuit for calculating the modular exponentiation.

Editing projects

For editing the projects after generation, you may add new files as you would normally, except you should make sure the files are placed in `PROJECT_FOLDER/Current_project/source/` for components and `PROJECT_FOLDER/Current_project/testbench/` for simulation sources such as testbenches, where `Current_project` is the project you are editing. In the case that you forget this, `cleanup.tcl` can copy files from the default directory to `source/` or `testbench/`. `cleanup.tcl` will also rename the files if a file of the same name already exists as to not overwrite any existing files. This means that you are responsible for managing collisions. When regenerating, the files in `source/` and `testbench/` are automatically added to the project. When you change the RSA_acelerator, you may want to update the IP core. This can be done with `generate_IP.tcl` or by clicking `Package IP` with the project open, then going through the steps.
After this is done, the ip core needs to be updated in rsa_soc, this can be done manually by editing the project (faster, but more involved) or by running `RSA_soc/cleanup.tcl` followed by `PROJECT_FOLDER/RSA_soc/rsa_soc.tcl`. This will reset any synthesis runs. It will yield consistent results albeit at a slow pace.
If you want to change the clock period of the driving clock, this has to be done by opening the `rsa_soc` project, then opening `rsa_soc.bd` in vivado and modifying the `processing_system7_0`(double clicking it) IP. On the window that opens up click `Clock configuration` of the left, then expand `PL Fabric Clocks` and change `FCLK_CLK0` to the desired frequency in MHz then press done. This change is stored, even after a project regeneration assuming you chose to save the changes.

Generating IP

To generate the IP of `rsa_accelerator` run the `generate_IP.tcl` script. This is needed in order for the project to properly integrate into jupyter notebooks.
Running `generate_ip.tcl` will update the ip, if it is already generated.

Synthesizing

To synthesize the project, run the `synthesize.tcl` script. this will generate `rsa_soc.bit` and `rsa_soc.hwh` which you should copy over to jupyter notebooks. `synthesize.tcl` will also run `generate_IP.tcl` to ensure that the IP is up to date when synthesizing. In the case of `rsa_soc`, the project will take a while to synthesize the first time, and as such doing a full regeneration is undesireable. `synthesize.tcl` will automatically update the `rsa_accelerator` IP before synthesizing. This will also generate reports and copies them into the `Reports/` folder. If you are synthesizing the project manually, `extract_reports.tcl` will copy out the same reports and bitfiles.

Delivery

Before delivery you should follow the following steps:

1. Copy the project folder to a different place. (Do the subsequent steps on this copy)
2. Run `cleanup.tcl`. After this the project should only contain the necessary file for delivery.
3. Run `regenerate_projects.tcl`. This is to make sure the project will work after being delivered.
4. Run `synthesize.tcl`. This generates the bitfiles and reports (should be included in the delivery).
5. Test the newly generated bitfiles in jupyter notebooks on the Pynq z1.
6. If everything worked so far, run `cleanup.tcl` again.
7. Zip the project folders and submit this zipped folder along with any other this you should submit.

Adding new subprojects (OPTIONAL)

1. open vivado.
2. Run `procedures.tcl`.
3. Navigate to `PROJECT_FOLDER` using the `cd` command, you can check the current directory with `pwd`.
4. Use the `instantiate_subproject` command to instantiate a new subproject. usage: `instantiate_subproject desired_name "parent_1 parent_2 ..."`. Where desired_name is the name of the subproject and parent_1 and parent_2 is the projects that should include the code from this subproject. Subprojects are recursively added, such that if a requires b and b requires c, c will be included in a.

5. If you did not specify any parents, or you typed the names incorrectly, you can manually add dependencies by adding the name on a new line in `include.txt` in the subproject you want to add the dependency to.
6. Now you may now modify the new subproject.

The lazy way

If you know your design works, the you may use the lazy way. You may run `The_lazy_way.tcl` and it will regenerate all the projects, synthesize `rsa_soc`, generate bitfiles, generate the reports and do the cleanup. Depending on your design, the time of this may take upwards of an hour. If your design is simple, it may only take a few minutes.

Testing on the Pynq Z1

setting up the board

1. Download Files for the Pynq z1
2. Connect to the Pynq z1. If you have not done this before, It is described how to do it here.
3. The sd card is not already formatted with the host name `DDSGROUP<NN>` where `<NN>` is the board number. This must be setup, but can be done simply with a command.
4. Connect over USB serial, and once you have a terminal connection run “sudo pynq_hostname.sh `DSGROUP<NN>`” `where <NN> is the group number, password is xilinx. Then reboot the board.`
5. From the downloaded `.zip` file, copy the following files to the your pynq board. Replace `<NN>` with your group number
    - Copy `RSA Integration Kit.ipynb` to `ddsgroup<NN>/jupyter_notebooks/`.
    - Copy `/rsa_soc/` to `ddsgroup<NN>/pynq/overlays/rsa_soc/`.
    - Copy `/crypto/` to `ddsgroup<NN>/pynq/crypto/`.
6. Navigate to `DDSGROUP<NN>:9090/` in your browser to access the web server of the pynq. The password (and username if applicable) is `xilinx`.

***

If the above option is not working, set a static IP address for ethernet as mentioned in the PYNQ Website. This board is then at address http://192.168.2.99/

Testing your design

When you are going to test your design on your Pynq Z1, you first have to synthesize your design. Start by copying `PROJECT_FOLDER/Bitfiles/rsa_soc.bit` and `PROJECT_FOLDER/Bitfiles/rsa_soc.hwh` to `ddsgroup<NN>/pynq/overlays/rsa_soc/`. The notebook should now utilize your

design. Then you start `RSA Integration Kit.ipynb` in Jupyter notebook on the Pynq Z1. Remember to set the correct algorithm in the notebook.