1 Initial Logic Synthesis

Learning Objectives

During this lab, you will work with Fusion Compiler tool and study design library creation and reading the RTL description.

After completing this lab, you should be able to:

- Invoke Fusion Compiler tool
- Create design library
- Read RTL
- Examine the technology independent circuit
- Impliment initial mapping
- Review the circuit
- Generate reports

Lab Duration: 40 minutes

Lab Instructions

Lab work structure

Lab directory organized as follows:

```
FC_Labs/labs/lab1_inital_syn/
   scripts/
        inital_syn.tcl
   work/
        logs/
        Makefile
```

Move to work directory:

```
cd FC_Labs/labs/lab1_inital_syn/work
```

You can run this lab by using Makefile in work directory, for that use the following command in terminal:

```
make run
```

Or invoke the Fusion Compiler and source the **read_rtl.tcl** script:

```
fc_shell -gui -no_log -f ../scripts/inital_syn.tcl | tee
./logs/read_rtl.log
```

In case you want to rerun the lab, you can clean your working area by using the following command in terminal:

```
make clean
```

At the beginning of lab work you **must** source the following setup scripts in Fusion Compiler to correcly setup the flow:

```
#### Sourcing common setup script
source -echo ../../setup/fc_common_setup.tcl
#### Sourcing flow setup script
source -echo ../../setup/fc_flow_setup.tcl
```

Invoke Fusion Compiler

There are 3 possible scenarios to invoke the Fusion Compiler:

1. To invoke Fusion Compiler in the terminal use the following command:

```
fc_shell
```

   To start or stop Graphical User Interface (GUI) from the fc_shell, use the below commands:

```
gui_start
gui_stop
```

2. To launch Fusion Compiler immediately with GUI, use the following command in the terminal:

```
fc_shell -gui &
```

3. To run Fusion Compiler with immediately sourcing script, use the below commands:

```
fc_shell -f path/to/script.tcl
fc_shell -f path/to/script.tcl -gui
```

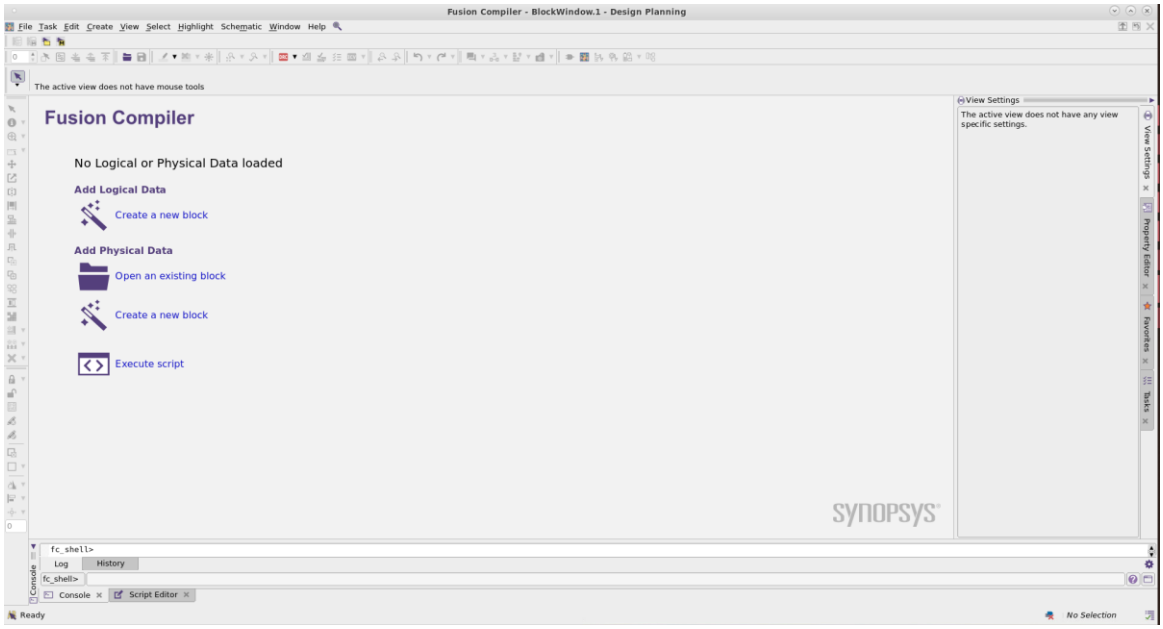

Fig. Main window of Fusion Compiler

Design Library Creation

Before design implementation the design library must be created. Design library can be created by following scenarios:

a. By using timing NDM and Technology File.

b. By using timing NDM and Technology Library. In this case Technology Library must be specified in the reference libraries list.

c. By using frame_only NDM, Liberty with required PVT corners (.db) and Technology File. In this case user must specify the corresponding .db files before design library creation by using below command:

```
set_app_var link_library "${libraries}"
```

Fusion Compiler will automatically call Library Manager tool to generate cell libraries with combined physical and logical data to use them as references for the design library.

Design library can be created by using GUI or by using tool commands:

1. To create design library with GUI open **File -> Create Library** window:

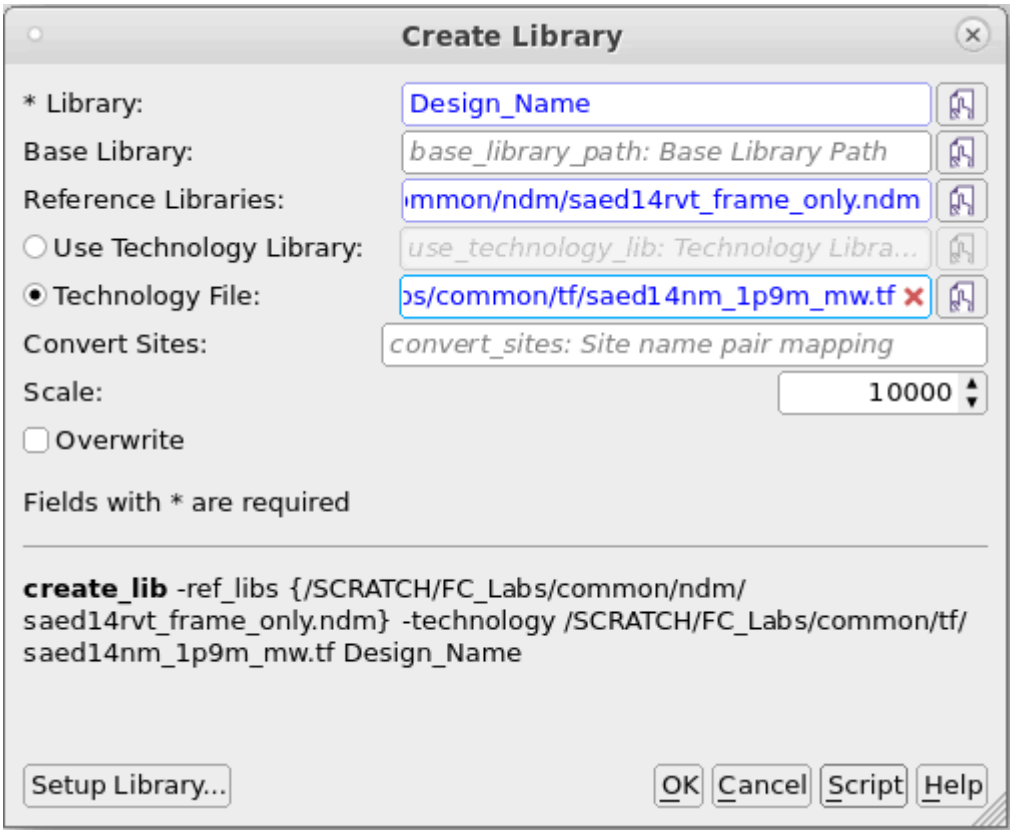

Fig. Create Library window

In the above window user can specify the library name, reference libraries, technology library or technology file.

2. The following script demonstrates the different ways to create the design library:

```
#### Library creation based on the Technology File
create_lib ${RESULTS_PATH}/${DESIGN_LIBRARY} \
    -technology $TECH_FILE -ref_libs ${REFERENCE_LIBRARY}

#### Library creation based on the Technology Library
create_lib ${RESULTS_PATH}/${DESIGN_LIBRARY} \
    -use_technology_lib ${TECH_NDM} \
    -ref_libs ${REFERENCE_LIBRARY}
```

To report reference libraries us the following command:

```
report_ref_libs
```

Read RTL

Fusion Compiler supports RTL descriptions developed by using the following Hardware Description Languages (HDLs):

a. Verilog

b. System Verilog

c. VHDL

To correctly read the Verilog RTL description, use the below commands:

1. Firstly the RTL must be analyzed, for that use the following command:

```
analyze -format verilog ${VERILOG_FILES}
```

The above command translates the specified HDL files and stores the intermediate format.

2. Then the design needs to be built. For that, use the following command:

```
elaborate ${DESIGN_NAME}
```

Here, the top design name is specified to elaborate the whole design.

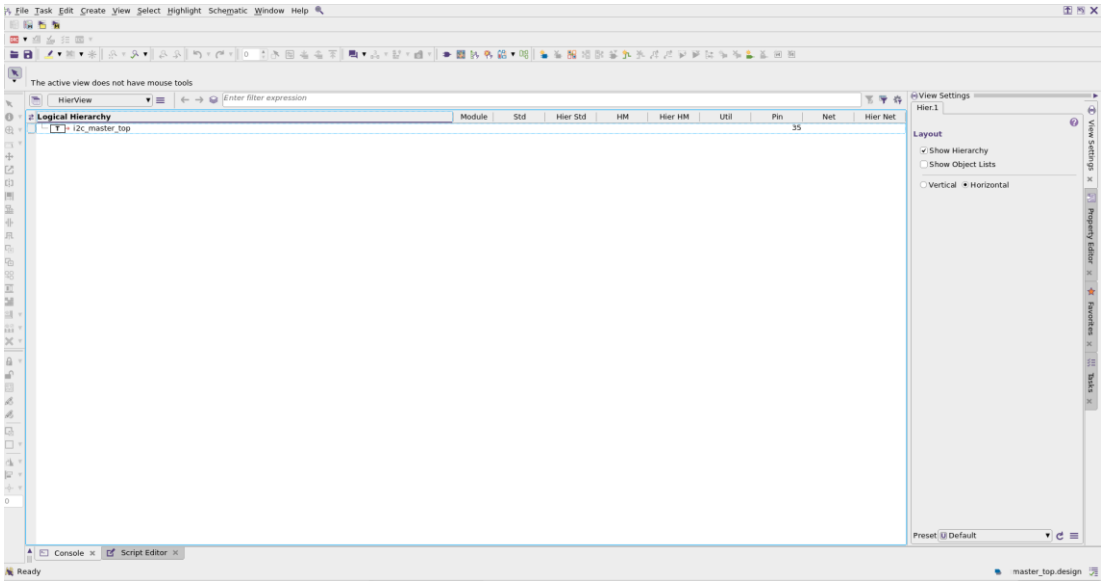

Fig. Logical Hierarchy Window after elaborate

3. After analyze and elaborate commands, use below command to create a design block and link the design:

```
set_top_module ${DESIGN_NAME}
```

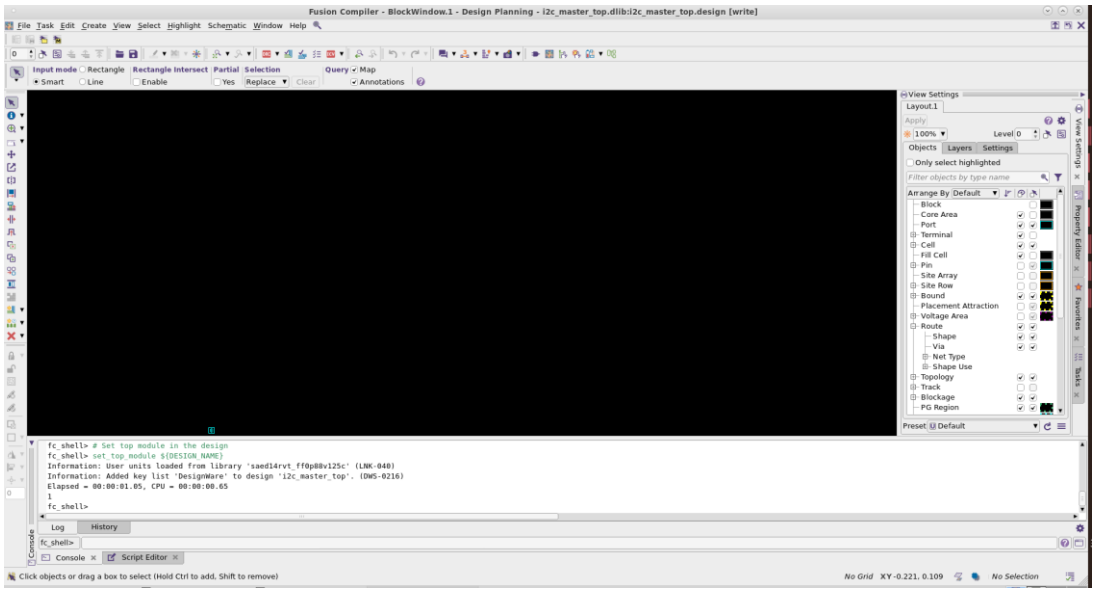

Fig. Block Window after set_top_module

After implementing set_top_module command Fusion Compiler automatically creates and links the block with **${library_name}:${design_name}.design** name. There is an empty area as no cell is used from the technology library yet.

To see the technology-independent circuit built by using integrated **WordView GTECH (WVGTECH)** library, click on **Create schematic of selected objects** tool on the top tool panel. After that, the below top-level schematic will be displayed.

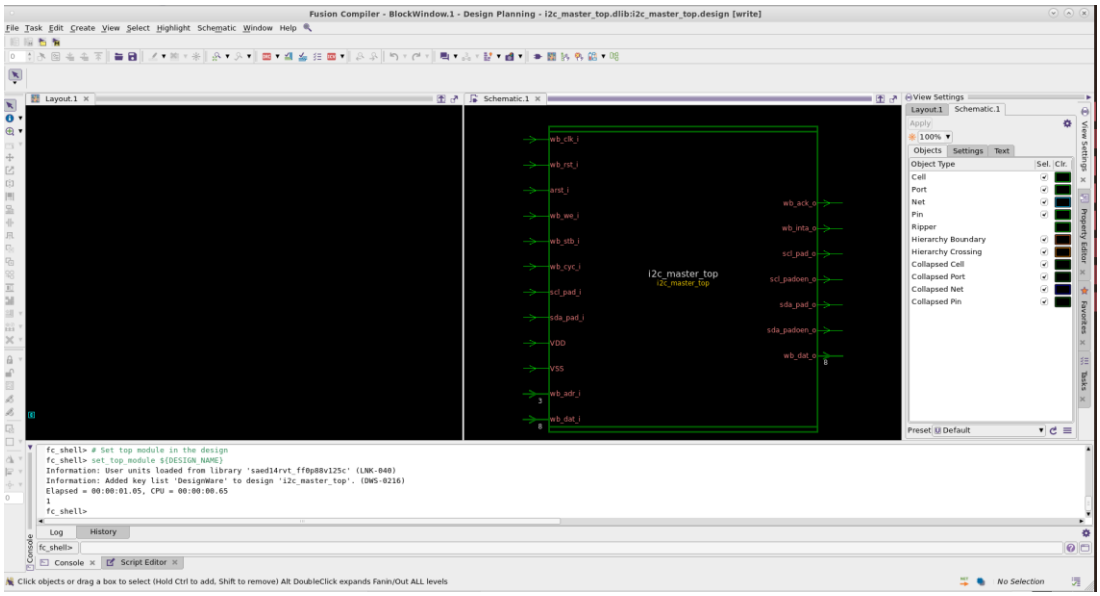

Fig. Block Window with schematic view

Double-click on the top block will expand it and show the cells inside the block.

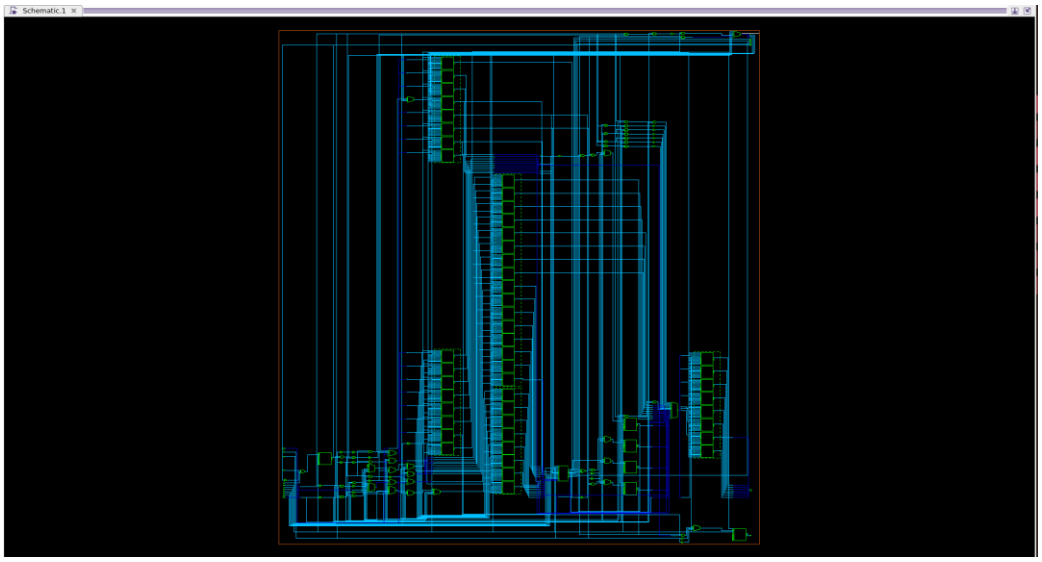

Fig. Technology-independent circuit

As was mentioned earlier, this circuit is **technology-independent** and built by using the integrated WVGTECH library. These cells have no technology-specific characteristics like occupied area, power consumption, or performance. They are only functional cells to represent the logic of provided RTL description.

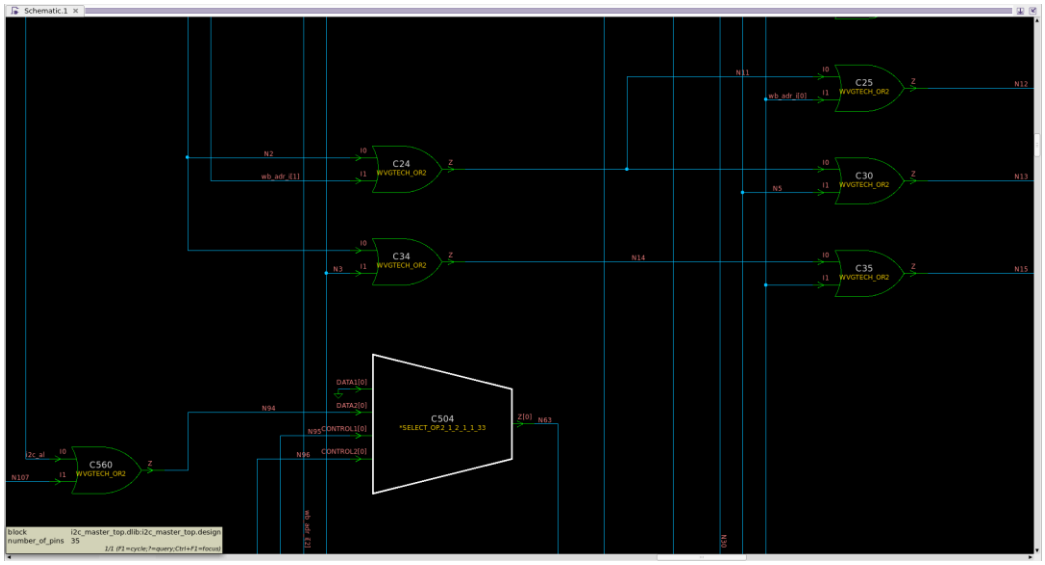

Fig. WVGTECH cells

4. Now lets perform initial mapping process to get **technology-dependent** circuit based on the provided library cells. For that use the following command:

```
compile_fusion -to initial_map
```

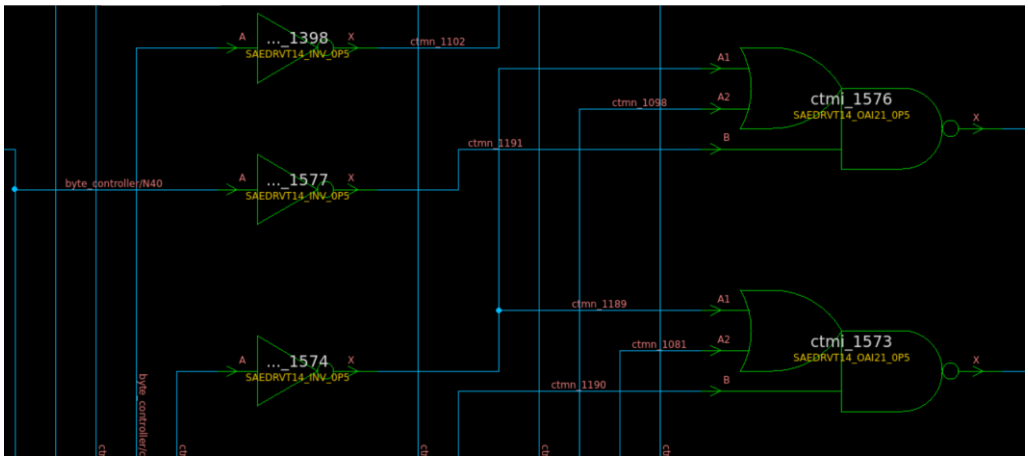

Fig. Logic circuit based on standard cells from the SAED14 RVT library

5. Performance, Power and Area (PPA) are critical parameters of integrated circuits. To analyze the design and understand which parameter of the design needs to be optimized, you can collect various types of reports for timing, power and area. To do that use the corresponding commands:

```
report_timing
report_power
report_area
```

You can notice that timing report is empty due to the missing constraints in the design:

```
fc_shell> report_timing
****************************************
Report : timing
        -path_type full
        -delay_type max
        -max_paths 1
        -report_by design
Design : i2c_master_top
Version: U-2022.12-SP4
Date   : Fri Mar  1 07:07:58 2024
****************************************
No paths.
1
```

Fig. Logic circuit based on standard cells from the SAED14 RVT library

Constraints and other necessary input data will be discussed in future labs.

To generate gate level netlist after initial mapping use the following command:

```
write_verilog ../../results/${DESIGN_NAME}_initial_syn.v
```

Gate level content is demonstrated in the below Figure:

```
SAEDRVT14_AO21_1 ctmi_1663 ( .A1 ( A[11] ) , .A2 ( ctmn_1236 ) ,
    .B ( ctmn_1237 ) , .X ( N_66 ) ) ;
SAEDRVT14_EO2_V1_0P75 ctmi_1664 ( .A1 ( A[10] ) , .A2 ( ctmn_1235 ) ,
    .X ( N_63 ) ) ;
SAEDRVT14_AO21_1 ctmi_1665 ( .A1 ( A[9] ) , .A2 ( ctmn_1234 ) ,
    .B ( ctmn_1235 ) , .X ( N_59 ) ) ;
SAEDRVT14_EO2_V1_0P75 ctmi_1666 ( .A1 ( A[8] ) , .A2 ( ctmn_1233 ) ,
    .X ( N_56 ) ) ;
SAEDRVT14_AO21_1 ctmi_1667 ( .A1 ( A[7] ) , .A2 ( ctmn_1232 ) ,
    .B ( ctmn_1233 ) , .X ( N_52 ) ) ;
SAEDRVT14_EO2_V1_0P75 ctmi_1668 ( .A1 ( ctmn_1231 ) , .A2 ( A[6] ) ,
    .X ( N_49 ) ) ;
SAEDRVT14_AO21_1 ctmi_1669 ( .A1 ( A[5] ) , .A2 ( ctmn_1230 ) ,
    .B ( ctmn_1231 ) , .X ( N_45 ) ) ;
SAEDRVT14_EO2_V1_0P75 ctmi_1670 ( .A1 ( ctmn_1229 ) , .A2 ( A[4] ) ,
    .X ( N_42 ) ) ;
SAEDRVT14_AO21_1 ctmi_1671 ( .A1 ( A[3] ) , .A2 ( ctmn_1228 ) ,
    .B ( ctmn_1229 ) , .X ( N_38 ) ) ;
SAEDRVT14_EO2_V1_0P75 ctmi_1672 ( .A1 ( ctmn_1239 ) , .A2 ( A[2] ) ,
    .X ( N_35 ) ) ;
SAEDRVT14_INV_0P5 ctmi_1674 ( .A ( A[0] ) , .X ( N_28 ) ) ;
```

Fig. Gate level netlist

After reading RTL and getting the initial circuit the design data need to be saved. To save the current block, use the following command:

```
save_block
```

To save a block with a specific name, use the below command:

```
save_block -as ${ANOTHER_BLOCK_NAME}
```

To report blocks in your library, use one of the following commands:

```
get_blocks -all
list_blocks
```

To save the library, use the below command:

```
save_lib
```

To exit the Fusion Compiler tool, use:

```
exit
```