4 Design

The PUF Interface design will function as the communication channel between the PUF circuit and the user by transmitting and receiving data via the UART module. The design will be able to take the user data received from the UART module and stored it in a RAM of size 2KB. Afterwards the data are process by the selected PUF circuit and stored in a second RAM which then can be downloaded to the terminal. The overall design consists of a mixture of controllers, memory modules, UART module, and multiplexers.

- Input Memory: This module stores the incoming data from the UART. The size of the memory is 2048 bytes with a width of 8-bits.
- Output Memory: This module stores the output values of the PUF circuit. The RAM size is 1024 bytes with a width of 8-bits.
- PUF Circuit: This is the user design component. The user constraints are defined as the following:
  - An input signal of 16-bits
  - An output signal of 8-bits
  - A done signal that will trigger active high when the output data is valid.
- Controller: Handles the communication between all the modules.
- UART: A simple serial module that will be able to ”dump” the memory contents of the attached memory module. In other words the UART module will cycle through the given memory location and transmit the data (8-bits) to the terminal one by one.

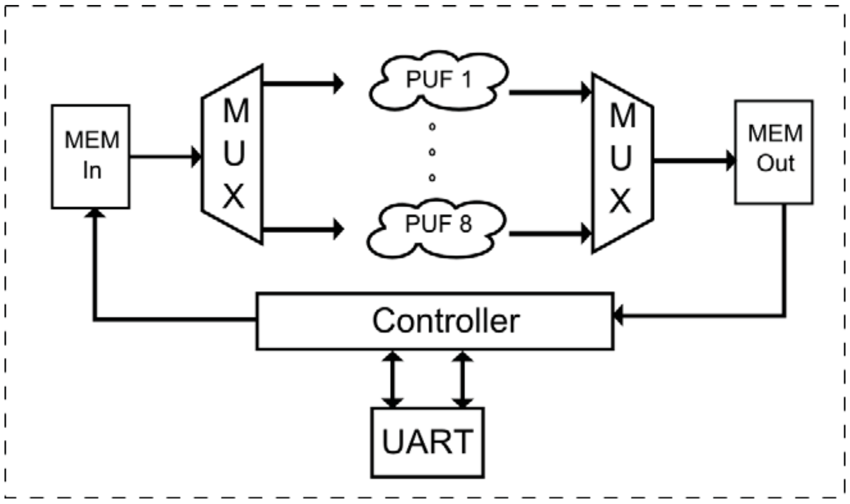


Figure 1: PUF Interface - Block Diagram

5 Evaluation metrics

Before explaining the metrics, we introduce few notations
K – Number of PUF circuits
C – Challenge to a PUF
R – Response from a PUF

N – Number of bits in a PUF's response
HD – Hamming distance
HD(R_i, R_j) – Hamming distance between response 'i' and response 'j'

Apart area, power, and delay every PUF design should satisfy some important criteria such as

- **Intra Hamming distance:** On flipping a bit in the challenge to a PUF, ideally, the hamming distance between the responses should differ 50% of total responses bits. If the there are k-chips and the challenges C1 and C2 differ by a bit, and the intra HD is estimated as

  $$\text{Intra HD} = \frac{1}{k}\sum_{i=1}^{k}\frac{HD(R_{i,1},R_{i,2})}{n} \times 100\%$$

  where $R_{i,1}$ is the response from chip 'i' for challenge C1 and $R_{i,2}$ is the response from chip 'i' for challenge C2.

  This metric is defined as

- **Inter Hamming distance:** On applying the same challenge to two different PUF designs, ideally, the hamming distance between their responses should differ 50% of total responses bits. If the there are k-chips, then the inter HD for a challenge C is defined as

  $$\text{Inter HD} = \frac{2}{k(k-1)}\sum_{i=1}^{k-1}\sum_{j=1+1}^{k}\frac{HD(R_i,R_j)}{n} \times 100\%$$

  where 'i' and 'j' are two different chips. R_i is the response from chip 'i' for the challenge C. R_j is the response from chip 'j' for the challenge C.

- **Bit aliasing:** If bit aliasing happens, then different chips will produce similar responses. Consequently, an attacker can easily guess the response. Bit aliasing for the i^{th} bit of a PUF across K different chips for a challenge C is estimated as

  $$\text{Bit aliasing}_i = \frac{1}{k}\sum_{j=1}^{k} R_{i,j} \times 100\%$$

  where $R_{i,j}$ is 1, if the i^{th} bit from the chip 'j' for the challenge C is 1. Otherwise, it is 0. Ideally, this value should be 50%.

- **Uniformity:** This metric defines how uniform the proportion of '1's and '0's in the response bits of a PUF. If the PUF has a bias towards '1' or '0' in its responses, then the attacker can guess that response. For an ideal PUF, the proportion of '1's and '0's in its responses should be equal. Uniformity for a PUF 'i' producing a n-bit response for a challenge C is estimated as

  $$\text{Uniformity}_i = \frac{1}{n}\sum_{j=1}^{n} R_{i,j} \times 100\%$$

  where $R_{i,j}$ is 1, if the j^{th} bit from the chip 'i' for the challenge C is 1. Otherwise, it is 0. Ideally, this value should be 50%.

- **Reliability:** On applying the same challenge several times, one should always get the same response. Consider a chip 'i' that produces the response R_{T1} for a challenge C at time T1 and produces the response R_{T2} for the same challenge at time T2. If the PUF is reliable, then R_{T1} and R_{T2} should be equal. Reliability can be quantified as

$$\text{Reliability} = 100 - \frac{1}{m}\sum_{t=T2}^{Tm}\frac{HD(R_{T1},R_t)}{n} \times 100\%$$

where T1, T2, ... Tm are different time instances. Ideally, this value should be 100.

- **Confidence interval:** Confidence interval estimates an interval within which the estimated value of a population lies with some confidence. In this lab, we will be estimating the interval for 95% confidence interval for the above metrics.

  If μ and σ are the mean and standard deviation, respectively, of the metric, then the lower value for 95% confidence interval is $\mu - 1.96(\frac{\sigma}{\sqrt{N}})$ and the upper value is $\mu + 1.96(\frac{\sigma}{\sqrt{N}})$.

  More details can be found at `http://web.utk.edu/~leon/stat201/Confidence%20Interval%20Concept.html`

Apart from criteria, there are several other criteria such as temperature stability, entropy, etc.,. A good discussion on metrics can be found at `http://rijndael.ece.vt.edu/puf/paper/iacr2011.pdf`

6 FPGA Board Interface

- `BTNC` - "Re-dump" command. If the UART controller is in the transmit state (`SW0` = 1) then pressing BTNC will "re-dump" the RAM contents. Note, pressing BTNC will not erase the RAM contents.
- `BTNL` - Start the selected PUF circuit.
- `SW0` - Download (0) / Upload (1) data from either Memory In or Memory Out.
- `SW1` - Select switch between Memory In (1) or Memory Out (0). Memory In will contain the upload contents, can be use to verify a correct upload. Memory Out will contain the calculated values.
- `LED` - Enable when PUF circuit calculation is done.
- `SW7`, `SW6`, `SW6` - serve as the multiplexer controlled switches for choosing the PUF circuit.

7 Experiment Flow

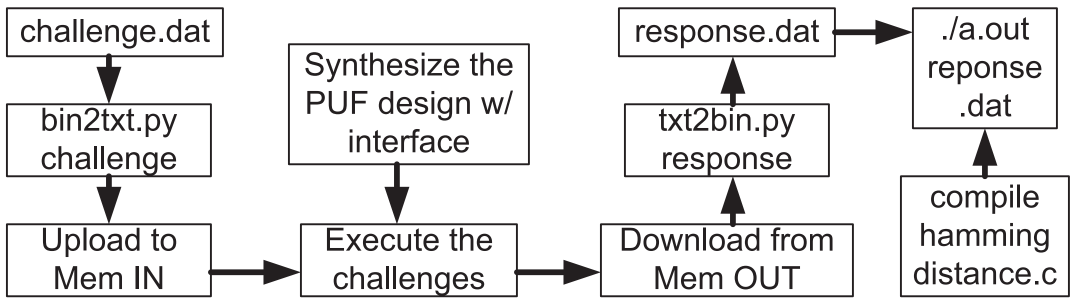


Figure 2: Experiment flow for this lab.

1. Using Table 1, pick the challenge file for the metric that you want to evaluate.
2. Use the python script `bin2txt.py <challengeFileName>` to convert your challenge file to ASCII format so that you send it to the FPGA.

Table 1: File containing the challenges to evaluate the respective metrics.

Experiment	File Name
Intra Hamming distance	Challenge1.txt
Inter Hamming distance, Bit aliasing, Uniformity	Challenge2.txt
Reliability	Challenge3.txt

3. Upload the challenge file into Mem IN using the `Terminal` program.
4. Meanwhile, you should have synthesized with the PUF design with the memory cores and interfaces on the target FPGA.
5. Apply the challenges from Mem IN and store the responses to Mem OUT.
6. Collect the responses from Mem OUT using the `Terminal` program.
7. Use the python script `txt2bin.py <responseFileName>` to convert your response file from ASCII file to binary format so that the C program can process it. Name the output file as `response.dat`
8. Meanwhile, compile the C code to calculate the Hamming distance. If you are using gcc compiler, you can use the command `gcc hammingdistance.c`.
9. Now, execute the compiled code to calculate the Hamming distance `.\a.out response.dat`

8 Software Packages

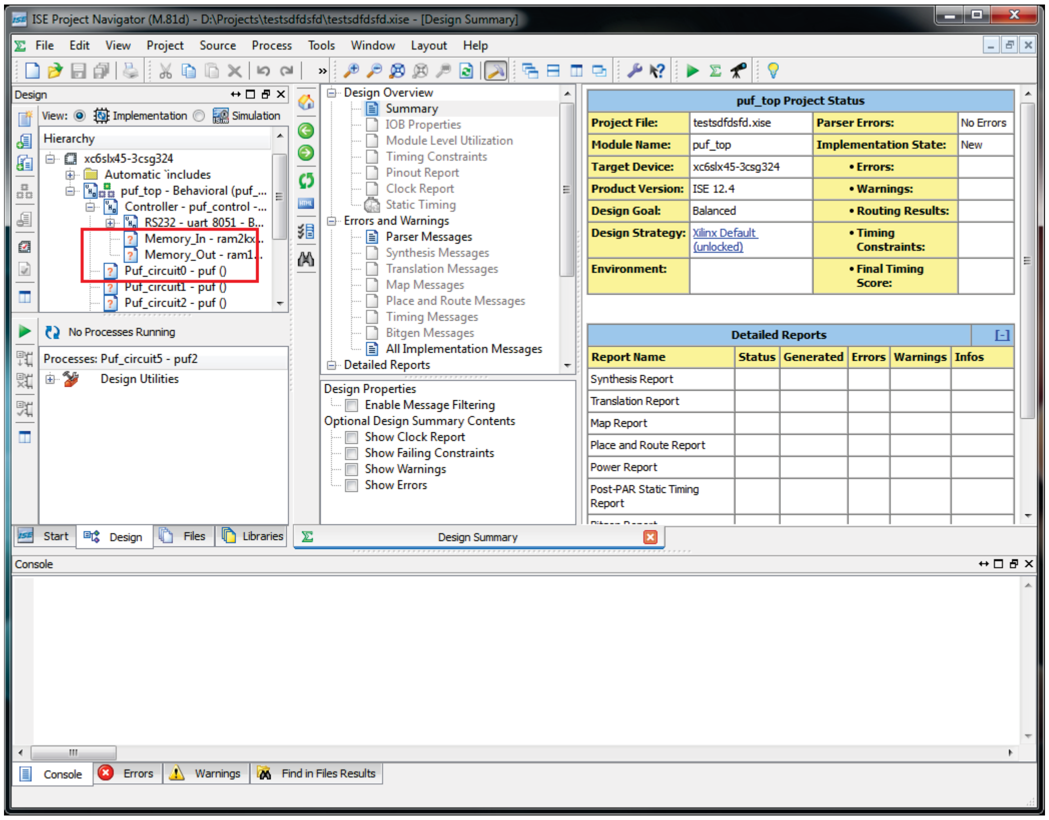
In order to synthesis the PUF design, there will be a few tools used throughout the guide.

- Xilinx ISE Webpack - Download from `http://www.xilinx.com/support/download/`. For compatibility, install version 12.4
- Digilent's Adept - `http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,66,828&Prod=ADEPT2`
- USB-UART Driver - Window's 7 will automatically download the required drivers. If it doesn't work, install the drivers from the link below:

  `http://www.exar.com/Common/Content/ProductDetails.aspx?ID=9546&ParentID=4`
- Terminal - is a simple serial port terminal program that served as the communication between the 8051 and computer. Can be download from `http://sites.google.com/site/terminalbpp/`
- txt2bin.py - Python script that takes a text file and converts it to a binary file. This is needed since the text file will be in ascii characters and we need the actual binary values. Usage: `txt2bin.py <filename>`
- bin2txt.py - Python script that perform the inverse of the above script. Usage: `bin2txt.py <filename>`
- HammingDistance.c - A C code to calculate hamming distance. Usage: compile using `gcc HammingDistance.c` and execute using `.\a.out responses.dat`

9 Design Flow

- Create a new Xilinx project and copy all the required files into the project. There will be several files which will appear to be missing. Two of them are the Memory IP Core file which we need to generate. The rest of them are the user-created PUF design.


ISE Project Navigator (M.81d) - D:\Projects\testsdfdsfd\testsdfdsfd.xise - [Design Summary]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
Console
Errors
Warnings
Find in Files Results
Start Design Files Libraries
Design Summary

```
xc6slx45-3csg324
  Automatic `includes
  puf_top - Behavioral (puf_...
    Controller - puf_control -...
      RS232 - uart 8051 - B...
      Memory_In - ram2kx...
      Memory_Out - ram1...
    Puf_circuit0 - puf ()
    Puf_circuit1 - puf ()
    Puf_circuit2 - puf ()

No Processes Running
Processes: Puf_circuit5 - puf2
  Design Utilities

Design Overview
  Summary
  IOB Properties
  Module Level Utilization
  Timing Constraints
  Pinout Report
  Clock Report
  Static Timing
Errors and Warnings
  Parser Messages
  Synthesis Messages
  Translation Messages
  Map Messages
  Place and Route Messages
  Timing Messages
  Bitgen Messages
  All Implementation Messages
Detailed Reports

Design Properties
  Enable Message Filtering
Optional Design Summary Contents
  Show Clock Report
  Show Failing Constraints
  Show Warnings
  Show Errors
```

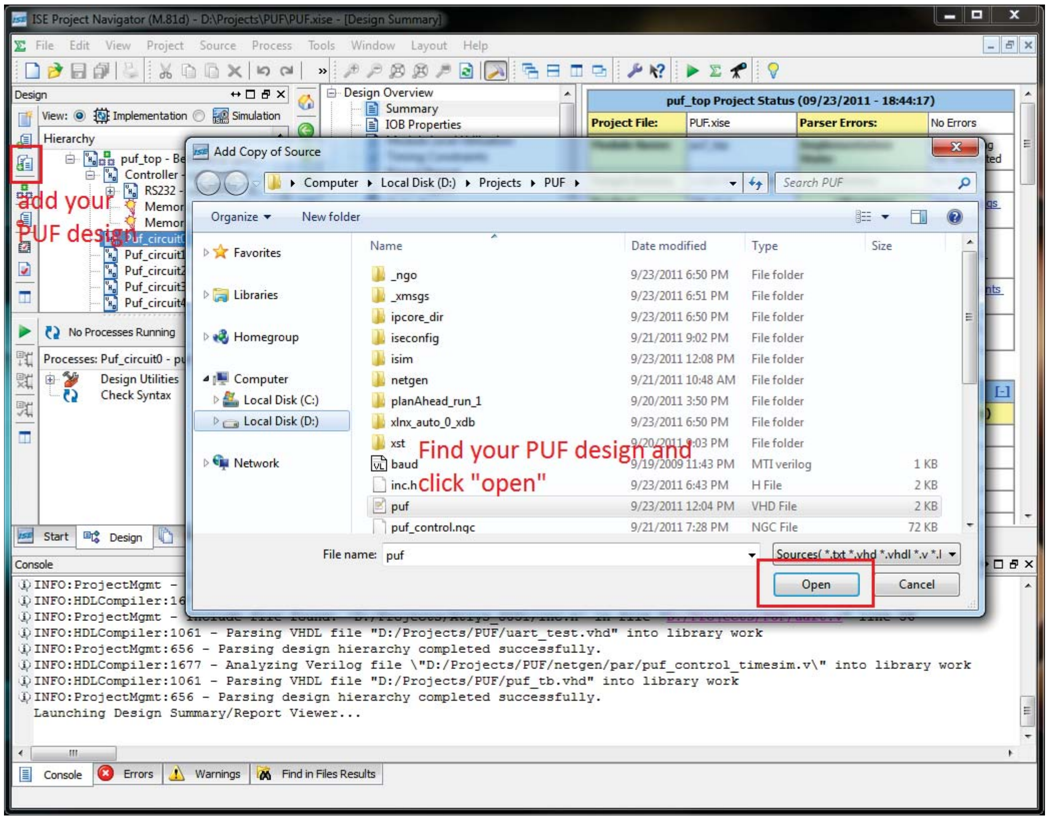
puf_top Project Status			
Project File:	testsdfdsfd.xise	**Parser Errors:**	No Errors
Module Name:	puf_top	**Implementation State:**	New
Target Device:	xc6slx45-3csg324	**• Errors:**	
Product Version:	ISE 12.4	**• Warnings:**	
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:		**• Final Timing Score:**	

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report					
Translation Report					
Map Report					
Place and Route Report					
Power Report					
Post-PAR Static Timing Report					

Figure 3: PUF Project

- Generate two IP memory files; one called `Memory_In` with a write width of 8 and a write depth of 2048. The second memory is called `Memory_Out` with a write width of 8 and a write depth of 1024. See the appendix 6.1 for more information.

- Integrate the PUF design into the project. Repeat the step to add all eight designs.

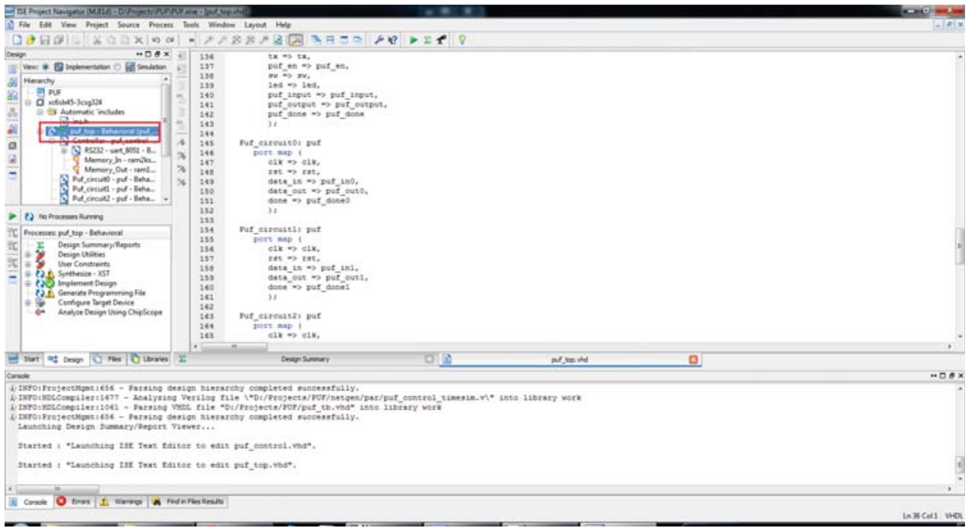


Name	Date modified	Type	Size
_ngo	9/23/2011 6:50 PM	File folder	
_xmsgs	9/23/2011 6:51 PM	File folder	
ipcore_dir	9/23/2011 6:50 PM	File folder	
iseconfig	9/21/2011 9:02 PM	File folder	
isim	9/23/2011 12:08 PM	File folder	
netgen	9/21/2011 10:48 AM	File folder	
planAhead_run_1	9/20/2011 3:50 PM	File folder	
xlnx_auto_0_xdb	9/23/2011 6:50 PM	File folder	
xst	9/20/2011 9:03 PM	File folder	
baud	9/19/2009 11:43 PM	MTI verilog	1 KB
inc.h	9/23/2011 6:43 PM	H File	2 KB
puf	9/23/2011 12:04 PM	VHD File	2 KB
puf_control.ngc	9/21/2011 7:28 PM	NGC File	72 KB

```
INFO:HDLCompiler:1061 - Parsing VHDL file "D:/Projects/PUF/uart_test.vhd" into library work
INFO:ProjectMgmt:656 - Parsing design hierarchy completed successfully.
INFO:HDLCompiler:1677 - Analyzing Verilog file \"D:/Projects/PUF/netgen/par/puf_control_timesim.v\" into library work
INFO:HDLCompiler:1061 - Parsing VHDL file "D:/Projects/PUF/puf_tb.vhd" into library work
INFO:ProjectMgmt:656 - Parsing design hierarchy completed successfully.
Launching Design Summary/Report Viewer...
```

Figure 4: Add PUF design

- Double click on the `puf_top` file in the design window (top left) and modified the file to include your component declaration and port mapping.

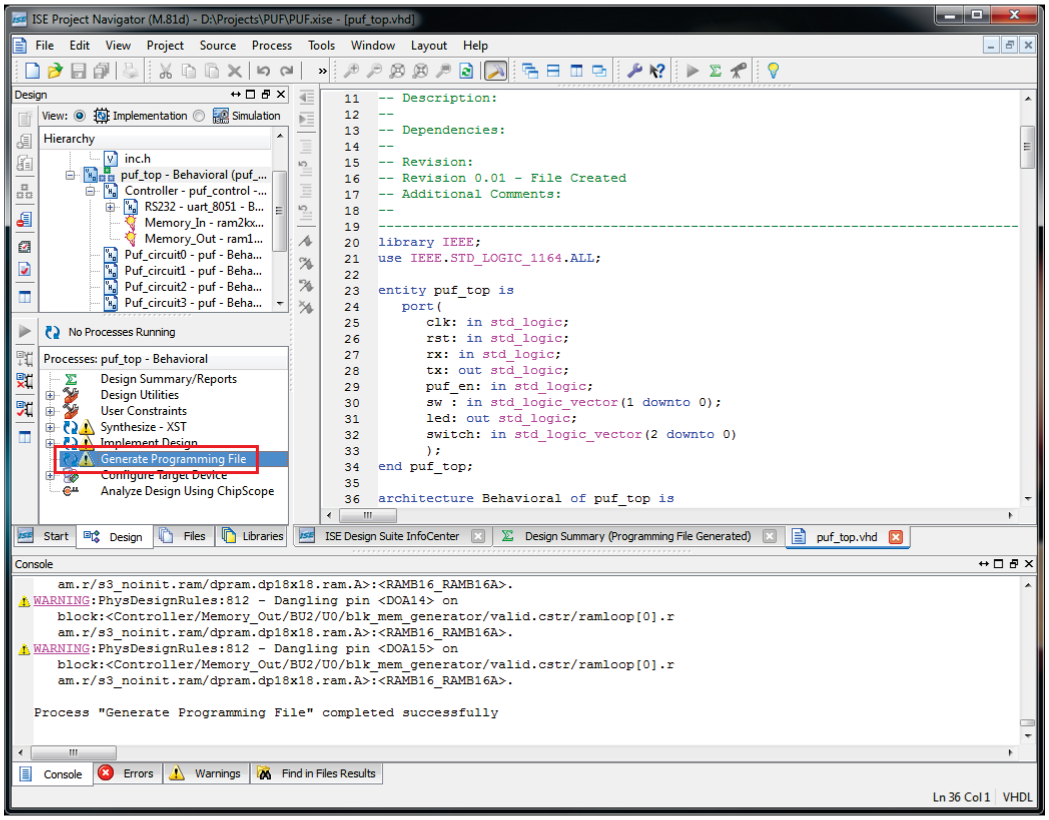


```
            tx => tx,
            puf_en => puf_en,
            sw => sw,
            led => led,
            puf_input => puf_input,
            puf_output => puf_output,
            puf_done => puf_done
            );

    Puf_circuit0: puf
        port map (
            clk => clk,
            rst => rst,
            data_in => puf_in0,
            data_out => puf_out0,
            done => puf_done0
            );

    Puf_circuit1: puf
        port map (
            clk => clk,
            rst => rst,
            data_in => puf_in1,
            data_out => puf_out1,
            done => puf_done1
            );

    Puf_circuit2: puf
        port map (
            clk => clk,
```

```
INFO:ProjectMgmt:656 - Parsing design hierarchy completed successfully.
INFO:HDLCompiler:1677 - Analyzing Verilog file \"D:/Projects/PUF/netgen/par/puf_control_timesim.v\" into library work
INFO:HDLCompiler:1061 - Parsing VHDL file "D:/Projects/PUF/puf_tb.vhd" into library work
INFO:ProjectMgmt:656 - Parsing design hierarchy completed successfully.
Launching Design Summary/Report Viewer...

Started : "Launching ISE Text Editor to edit puf_control.vhd".

Started : "Launching ISE Text Editor to edit puf_top.vhd".
```

Figure 5: Open top level file

- Double click on ”Generate Programming File” to get the bit file.

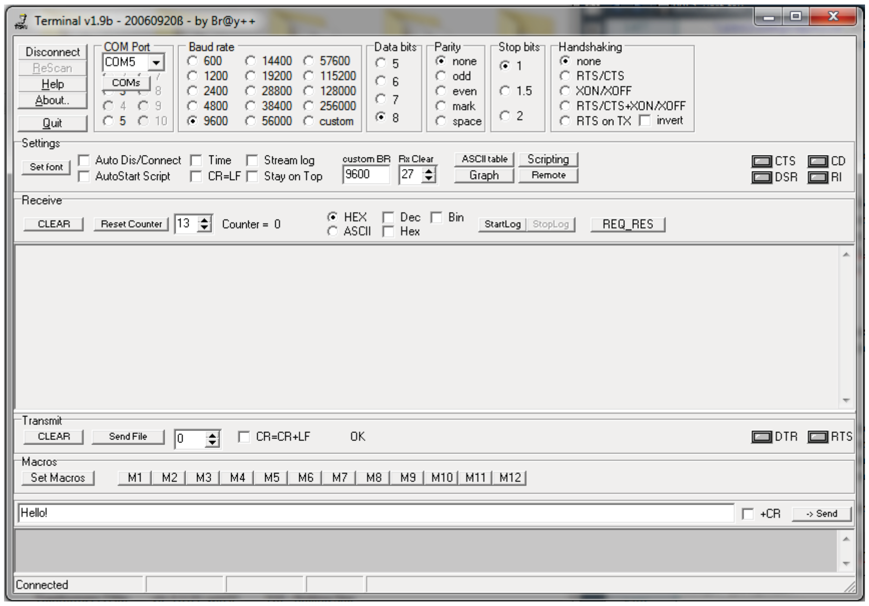

ISE Project Navigator (M.81d) - D:\Projects\PUF\PUF.xise - [puf_top.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
inc.h
puf_top - Behavioral (puf_...
Controller - puf_control -...
RS232 - uart_8051 - B...
Memory_In - ram2kx...
Memory_Out - ram1...
Puf_circuit0 - puf - Beha...
Puf_circuit1 - puf - Beha...
Puf_circuit2 - puf - Beha...
Puf_circuit3 - puf - Beha...
No Processes Running
Processes: puf_top - Behavioral
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
ISE Design Suite InfoCenter
Design Summary (Programming File Generated)
puf_top.vhd
Console
Console Errors Warnings Find in Files Results
Ln 36 Col 1 VHDL

```
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity puf_top is
   port(
      clk: in std_logic;
      rst: in std_logic;
      rx: in std_logic;
      tx: out std_logic;
      puf_en: in std_logic;
      sw : in std_logic_vector(1 downto 0);
      led: out std_logic;
      switch: in std_logic_vector(2 downto 0)
      );
end puf_top;

architecture Behavioral of puf_top is
```

```
   am.r/s3_noinit.ram/dpram.dp18x18.ram.A>:<RAMB16_RAMB16A>.
WARNING:PhysDesignRules:812 - Dangling pin <DOA14> on
   block:<Controller/Memory_Out/BU2/U0/blk_mem_generator/valid.cstr/ramloop[0].r
   am.r/s3_noinit.ram/dpram.dp18x18.ram.A>:<RAMB16_RAMB16A>.
WARNING:PhysDesignRules:812 - Dangling pin <DOA15> on
   block:<Controller/Memory_Out/BU2/U0/blk_mem_generator/valid.cstr/ramloop[0].r
   am.r/s3_noinit.ram/dpram.dp18x18.ram.A>:<RAMB16_RAMB16A>.

Process "Generate Programming File" completed successfully
```

Figure 6: Generating bit file

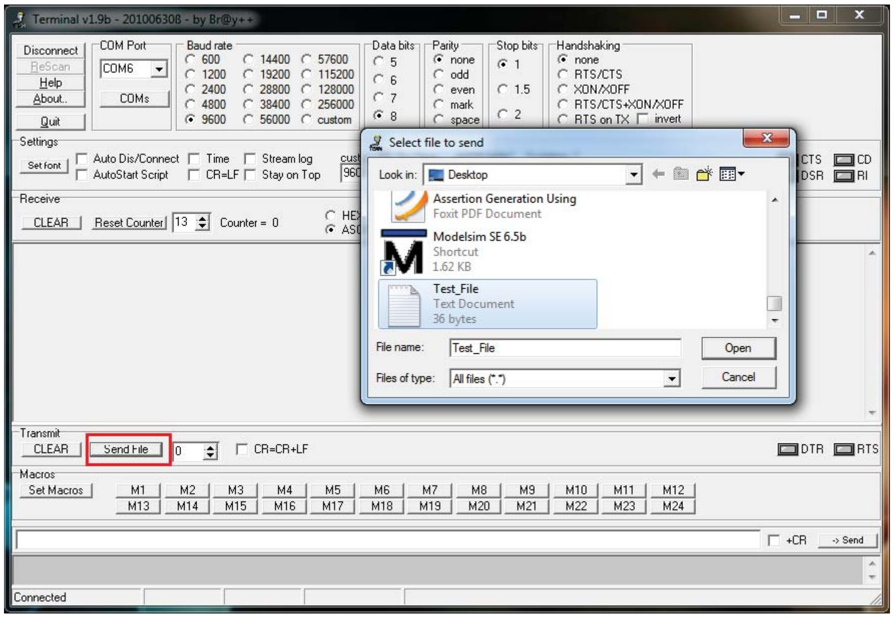
- Attach the USB cable to the correct slot and launch Digilent’s Adept program. Locate the PUF bit file and hit ”program”.
- Since Digilent only provided one USB cable, use the same cable and change from the PROG to the UART port.
- Launch the terminal program and connect to the appropriate COM port. The baud rate is 9600, 8 data bits, no parity, 1 stop bit and no handshaking. Set the receive option to view hex.


Terminal v1.9b - 20060920B - by Br@y++

COM Port: COM5
Baud rate: 9600
Data bits: 8
Parity: none
Stop bits: 1
Handshaking: none
custom BR: 9600
Rx Clear: 27
Receive: HEX
Counter = 0
Transmit: 0
OK
Hello!
Connected

Figure 7: Terminal program

- To upload a file, set SW0 to "1" and press BTNC. This is to insure that the file is upload to the beginning of the memory location. Then click "send file" and choose the file to upload. Once the file is sent, set SW0 back to "0". Any old data in the terminal window can be clear using the CLEAR option.

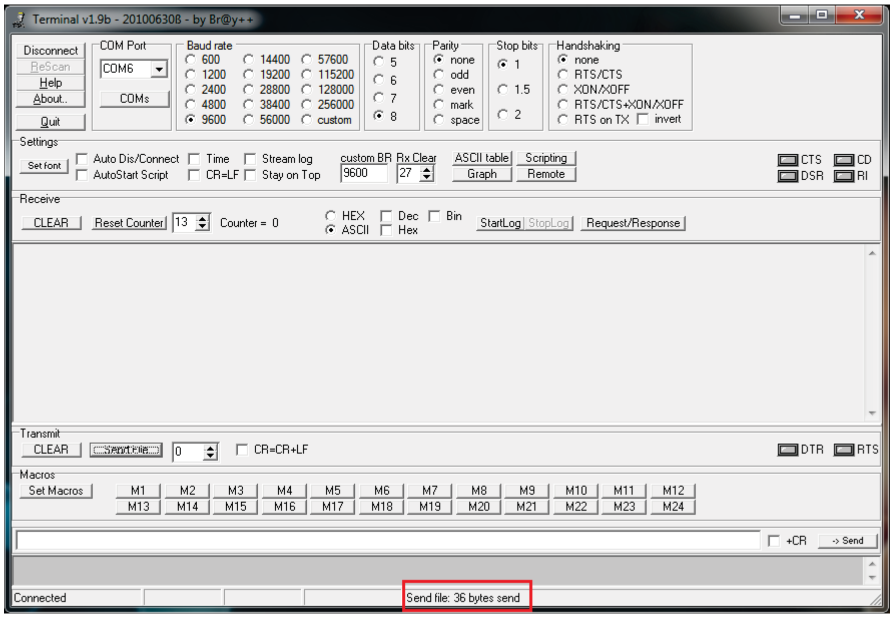

Terminal v1.9b - 20100630B - by Br@y++

COM Port: COM6
Baud rate: 9600
Data bits: 8
Parity: none
Stop bits: 1
Handshaking: none
Counter = 0
Connected

Select file to send
Look in: Desktop
Assertion Generation Using — Foxit PDF Document
Modelsim SE 6.5b — Shortcut — 1.62 KB
Test_File — Text Document — 36 bytes
File name: Test_File
Files of type: All files (*.*)

Open
Cancel
Send File
CLEAR
Set Macros
M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12
M13 M14 M15 M16 M17 M18 M19 M20 M21 M22 M23 M24
+CR
-> Send

Figure 8: Uploading file



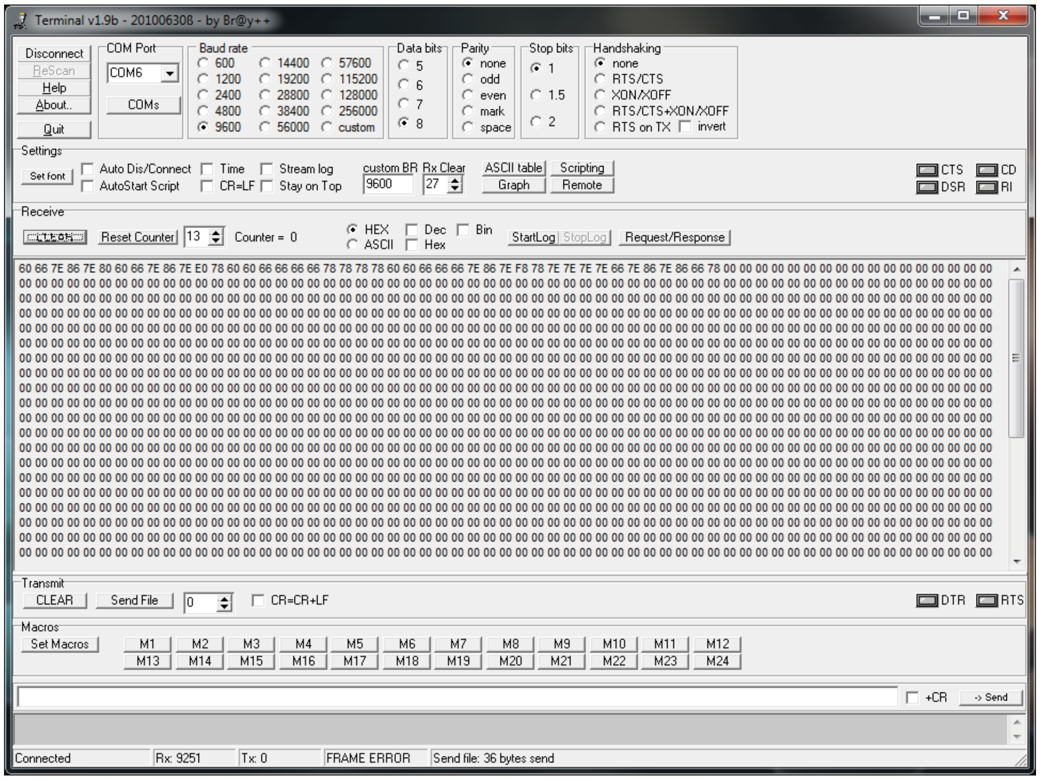
COM Port: COM6
Baud rate: 9600
Data bits: 8
Parity: none
Stop bits: 1
Handshaking: none
custom BR: 9600
Rx Clear: 27
Receive: ASCII, 13, Counter = 0
Transmit: 0

Connected

Send file: 36 bytes send

Figure 9: Uploading file Done

- Select which PUF design you want to perform the calculations. Use `SW7 SW6 SW5` to select the design. For example, setting switches `SW7 SW6 SW5` to "101" respectively will enable the fifth PUF design.
- Start the calculation by pressing `BTNL`, after the calculation is done, the led will light up.
- To view the calculated values, press `BTNC`. This will re-dump the memory contents to the terminal. To view the previous uploaded values, set `SW1` to "1" and press `BTNC`



COM Port: COM6
Baud rate: [✓ 9600]
Data bits: [✓ 8]
Parity: [✓ none]
Stop bits: [✓ 1]
Handshaking: [✓ none]
custom BR: 9600 Rx Clear: 27
Receive: 13 Counter = 0 [✓ HEX] ASCII

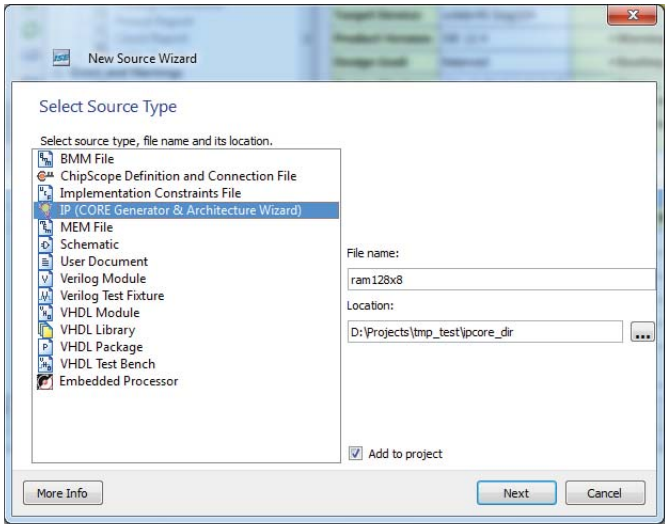
```
60 66 7E 86 7E 80 60 66 7E 86 7E E0 78 60 60 66 66 66 66 78 78 78 78 60 60 66 66 66 7E 86 7E F8 78 7E 7E 7E 7E 66 7E 86 7E 86 66 78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
Connected    Rx: 9251    Tx: 0    FRAME ERROR    Send file: 36 bytes send
```

Figure 10: Memory dump

10 Appendix

10.1 Generating memory IP Core



Select source type, file name and its location.

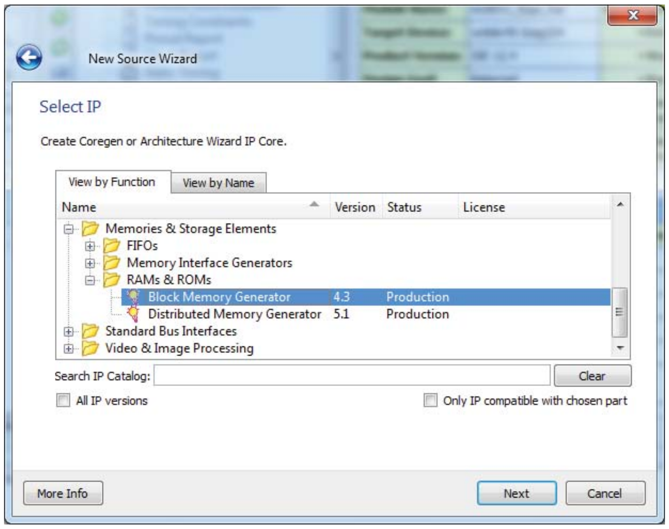
- BMM File
- ChipScope Definition and Connection File
- Implementation Constraints File
- IP (CORE Generator & Architecture Wizard)
- MEM File
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

File name: ram128x8

Location: D:\Projects\tmp_test\ipcore_dir

[✓] Add to project

Figure 11: RAM 2kx8



Create Coregen or Architecture Wizard IP Core.

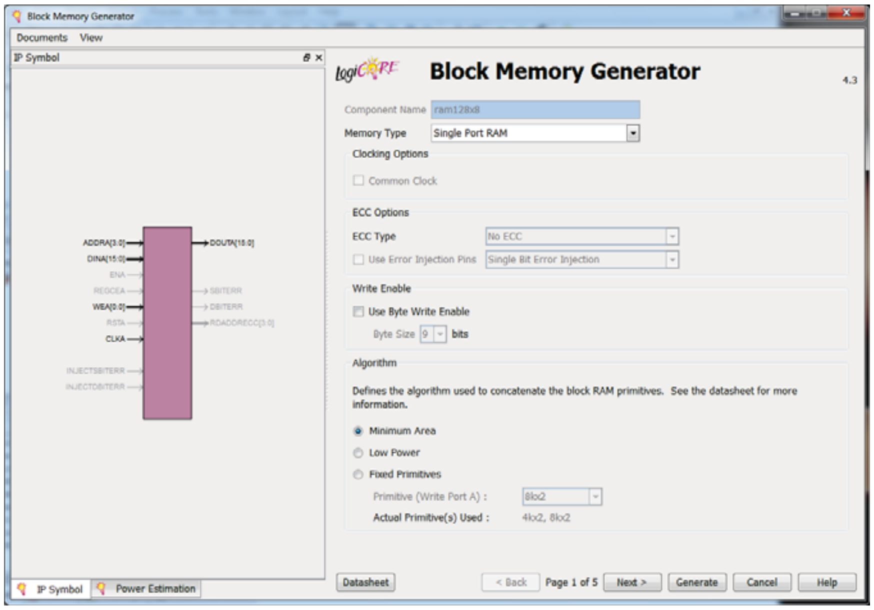
Name	Version	Status	License
Memories & Storage Elements			
FIFOs			
Memory Interface Generators			
RAMs & ROMs			
Block Memory Generator	4.3	Production	
Distributed Memory Generator	5.1	Production	
Standard Bus Interfaces			
Video & Image Processing			

Search IP Catalog:

[ ] All IP versions

[ ] Only IP compatible with chosen part

Figure 12: RAM 2kx8



Block Memory Generator 4.3

Component Name: ram128x8
Memory Type: Single Port RAM

Clocking Options
- [ ] Common Clock

ECC Options
- ECC Type: No ECC
- [ ] Use Error Injection Pins: Single Bit Error Injection

Write Enable
- [ ] Use Byte Write Enable
- Byte Size: 9 bits

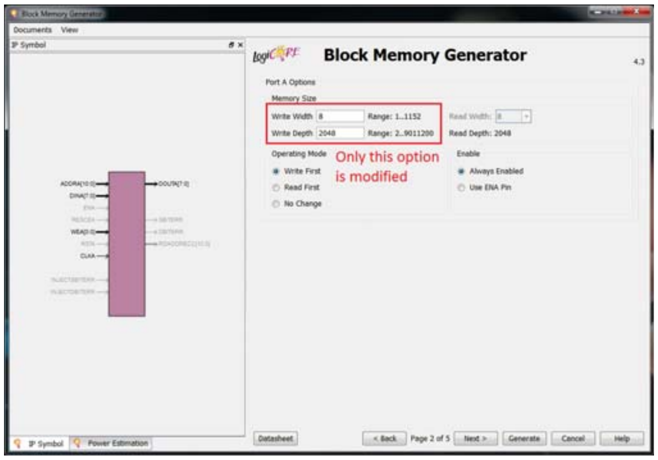
Algorithm

Defines the algorithm used to concatenate the block RAM primitives. See the datasheet for more information.

[✓ Minimum Area]
Low Power
Fixed Primitives

Primitive (Write Port A) : 8kx2
Actual Primitive(s) Used : 4kx2, 8kx2

Figure 13: RAM 2kx8



Block Memory Generator 4.3

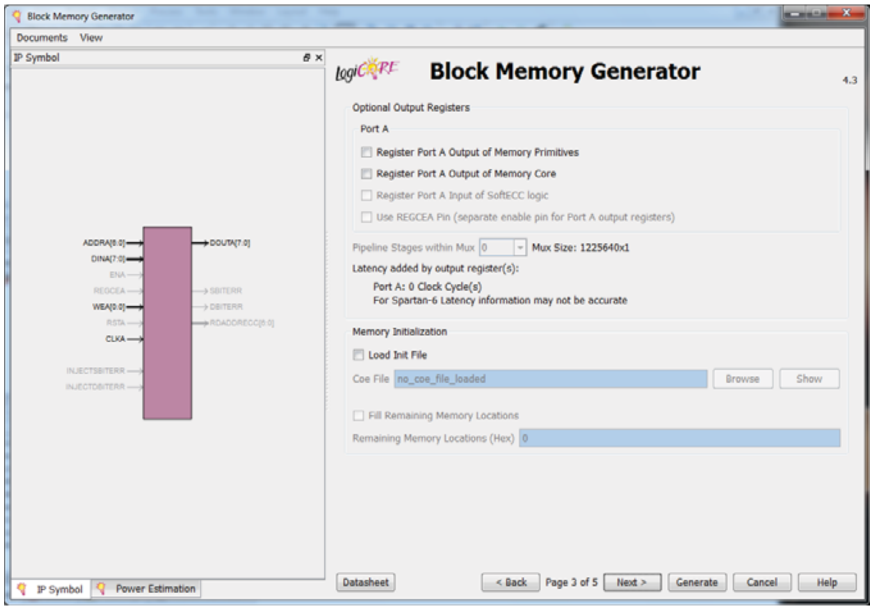
Port A Options

Memory Size
- Write Width: 8 Range: 1..1152
- Write Depth: 2048 Range: 2..9011200
- Read Width: 8
- Read Depth: 2048

Operating Mode: [✓ Write First] Read First No Change

Enable: [✓ Always Enabled] Use ENA Pin

Figure 14: RAM 2kx8



Block Memory Generator

Optional Output Registers

Port A

- [ ] Register Port A Output of Memory Primitives
- [ ] Register Port A Output of Memory Core
- [ ] Register Port A Input of SoftECC logic
- [ ] Use REGCEA Pin (separate enable pin for Port A output registers)

Pipeline Stages within Mux 0 Mux Size: 1225640x1

Latency added by output register(s):

Port A: 0 Clock Cycle(s)

For Spartan-6 Latency information may not be accurate

Memory Initialization

- [ ] Load Init File

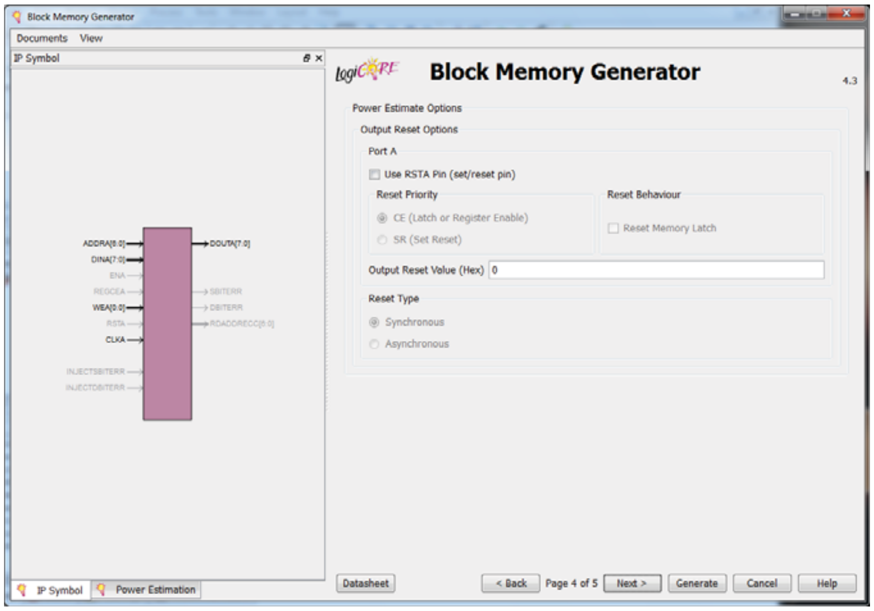
Coe File no_coe_file_loaded

- [ ] Fill Remaining Memory Locations

Remaining Memory Locations (Hex) 0

Page 3 of 5

Figure 15: RAM 2kx8



Block Memory Generator

Power Estimate Options

Output Reset Options

Port A

- [ ] Use RSTA Pin (set/reset pin)

Reset Priority

[✓ CE (Latch or Register Enable)]
SR (Set Reset)

Reset Behaviour

- [ ] Reset Memory Latch

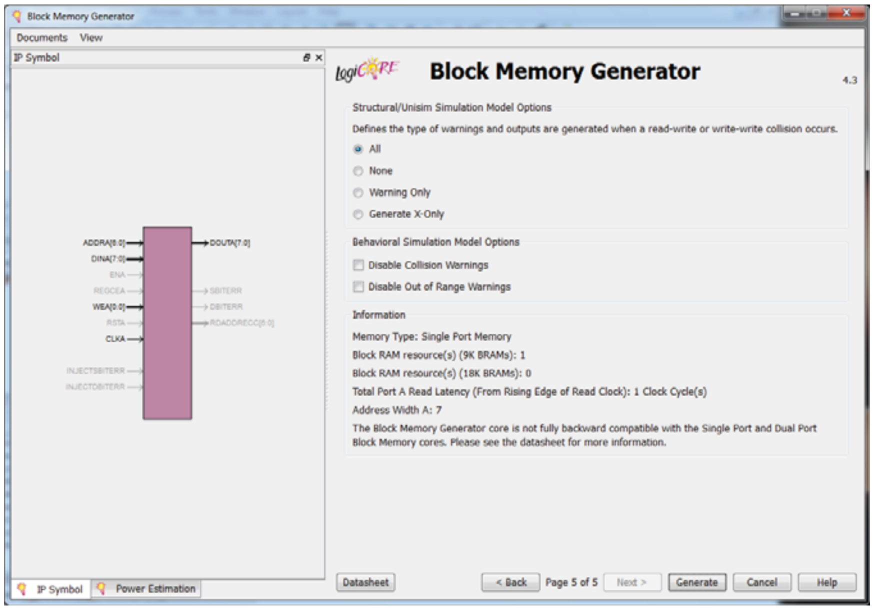
Output Reset Value (Hex) 0

Reset Type

[✓ Synchronous]
Asynchronous

Page 4 of 5

Figure 16: RAM 2kx8



Block Memory Generator

Structural/Unisim Simulation Model Options

Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.

[✓ All] None Warning Only Generate X-Only

Behavioral Simulation Model Options

☐ Disable Collision Warnings

☐ Disable Out of Range Warnings

Information

Memory Type: Single Port Memory

Block RAM resource(s) (9K BRAMs): 1

Block RAM resource(s) (18K BRAMs): 0

Total Port A Read Latency (From Rising Edge of Read Clock): 1 Clock Cycle(s)

Address Width A: 7

The Block Memory Generator core is not fully backward compatible with the Single Port and Dual Port Block Memory cores. Please see the datasheet for more information.

Figure 17: RAM 2kx8