**FPGA Implementation Document**

**1. Higher level overview and Connections:**

FPGA\_Random\_Walk\_Module can are defined in bram\_random\_walk.v file (Verilog to be found in github https://github.com/lixali/FPGA\_PPR/tree/master/FPGA/ultra96\_random\_walk), and it can be instantiated into multiple instances depending on how many processing is needed for the application (the following picture shows the 2 FPGA\_Random\_Walk\_Module instantiation)

Diagram

Description automatically generated

The connections to ZCU102 PS is described in section 6.

**2. What and how Data are stored in each BRAM Block:**

The following diagram shows what stores in each BRAM Block. In each BRAM block, seed node is stored from the 0x0 to 0x(K-1) address (K is the number of seed node). From 0xK to 0x(K+2\*N), it stores the first and last neighbour’s address for each node in the subgraph (N is number of nodes in the subgraph). Starting from Ox(K+2\*N + 1), it stores the neighbours for each node in the subgraph.

The following picture shows 3 rectangular boxes and the first box is ranging from 0x0 to 0x(K-1), and the second box ranging is from 0xK to 0x(K+2\*N), and the third box is starting from Ox(K+2\*N+1)

Graphical user interface, table

Description automatically generated

**3. Implementation of random walk block module in Verilog**

The verilog random walk module is responsible for reading out the data stored in the bram (the data stored in the bram is illustrated in section 2; The data consist of 3 part, seed node, neighbours address, neighbour nodes). The always block (starting at line 76 in <https://github.com/lixali/FPGA_PPR/blob/master/FPGA/ultra96_random_walk/bram_random_walk.v>) is a finite state machine that reads/writes BRAM (the state is determined by the variable “read\_write\_count”).

Please note that the “seed nodes number”, “number of random walks”, “number of steps” are pre-defined as “parameter” in Verilog module so that the module will know number of seed nodes & seeds nodes’ address that it is required to read from BRAM. In order to access the “neighbour address” and “neighbour nodes” table in the BRAM, 2 more variable “nei\_addr\_table\_offset” are “nei\_table\_offset” are also pre-defined as parameter in the Verilog module (these 2 variable are determined by number of seed nodes and number of nodes in the sub-graph).

**4. Other verilog files and sub-modules:**

Under https://github.com/lixali/FPGA\_PPR/tree/master/FPGA/ultra96\_random\_walk/, the following Verilog files are created.

(1)BRAM.v(module definition of BRAM block, used in the test bench bram\_random\_walk\_testbech.v)

(2) Clock\_divider.v (clock divider modulethat make main clocks into various clocks with various frequency)

(3) lfsr.v (linear feedback shift register to mimic generating random numbers in FPGA)

**5. Simulations Waveform**

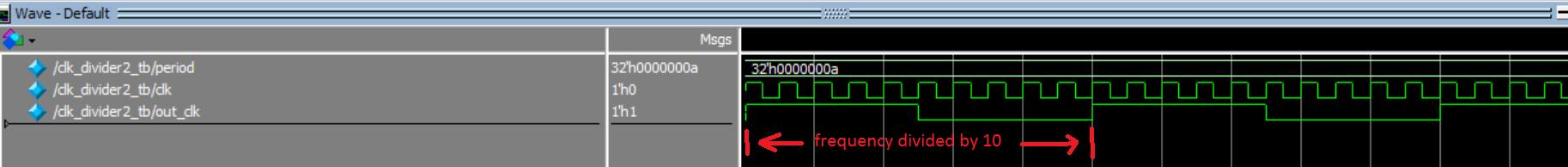
All the test bench files can also be found in the same github folder. The following waveforms show the simulations result for various block. Because modelsim is running faster in simulation than vivado, modelsim is used in here. But synthesis/implementation/bitstream generation are still done in vivado

BRAM basic read and write simulation (bram\_basic\_testbench.v)

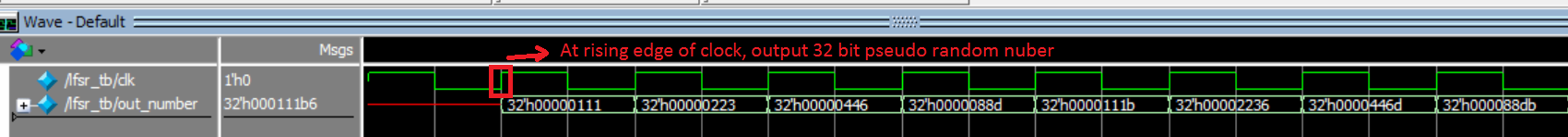
A picture containing graphical user interface

Description automatically generated

Clock frequency divider (clock\_divider\_tb.v)



Linear Feedback Shift Register (lfsr\_tb.v)



Modelsim simulation project (bram\_random\_walk\_testbench.v)

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Simulation waveform of BRAM\_random\_walk module (bram\_random\_walk.v)

A screenshot of a computer

Description automatically generated

**6. Connections to zcu102’s PS**