**FPGA Implementation Document**

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

The PL’s goal is to **process subgraph and calculate “scores” in parallel**. The document will explain how PL works when **number of parallel run is “2”** (same parallel mechanism can apply to 4,8, etc.). The PL’s block diagram is shown as followed.

Diagram

Description automatically generated

(1) Firstly, a subgraph is split and stored into 2 BRAM blocks (“BRAM\_subgraph\_first\_half” and “BRAM\_subgraph\_second\_half”; how a subgraph is stored is explained is in section 2).

(2) The final score table is stored in 2 BRAM blocks (“BRAM\_score\_table\_first\_half” and “BRAM\_score\_table\_second\_half”; how are data stored in score table is explained in section3).

(3) M1 is responsible for reading out “node’s previous score”, “node’s neighbours current score” and update “node’s neighbours current score”. Details of how M1 is written in verilog is explained section 4.

(4) Scheduler block implementation detail is explained in section 5.

(5) Sync\_control implementation detail is explained in section 6.

**2. How are data stored in each “BRAM\_subgraph”:**

The following diagram shows what stores in each BRAM Block. Supposedly there are N nodes in one subgraph. In each BRAM block, 0x0 to 0x(N\*2) addresses are to store the first and last neighbour address of the node. Starting from Ox(2\*N + 1), it stores the neighbours for each node in the subgraph.

A picture containing graphical user interface

Description automatically generated

**3. How are data stored in each “BRAM\_score\_table”:**

Both previous and current scores need to be stored. The following picture shows how the “previous” and “current” score are organized in the BRAM\_score\_table. Please note that in each new run, current score will become previous score.

**Text

Description automatically generated**

**4. Implementation of M1 Module**

The following is the functionality of M1 (same functionality applies to M2).

The M1 module is responsible for doing the following.

(1)read out node’s previous score stored in “subgraph\_score\_table\_first\_half” (S1).

(2)read out node’s neighbours (will calculate node’s degree)

(3)read out node’s neighbours’ latest score (s2)

(4)write to neighbour nodes with a new score (s2+s1/degree)

The following timing diagram shows M1’s “state” in the Verilog (5 states in total; all 5 states are inside the “always” starting at line 35 in https://github.com/lixali/FPGA\_PPR/blob/master/FPGA/diffusion\_rw/diffusion\_rw.v).

Diagram

Description automatically generated

**5. Implementation of Scheduler**

The “scheduler” is to **push back operation of M2 in case of “M1, M2 conflict”**(the Verilog implementation is in “https://github.com/lixali/FPGA\_PPR/tree/master/FPGA/diffusion\_rw/schedular\_dual.v“). The basic logic building block of “scheduler” is “https://github.com/lixali/FPGA\_PPR/tree/master/FPGA/diffusion\_rw/conflict\_block.v“ and its functionality is shown in the following picture.

Table

Description automatically generated

“schedular\_dual.v” is built on top of “conflict\_block.v”. Its block diagram with M1&M2 is shown in the following picture.

Diagram

Description automatically generated

**5. Implementation of sync\_control**

The sync\_control is to make sure that **M1 wait for M2 to finish before both of them starts a new run** (the Verilog file is “https://github.com/lixali/FPGA\_PPR/blob/master/FPGA/diffusion\_rw/sync\_control\_dual.v”). The block diagram of sync\_control with connection to M1 & M2 is shown in the following picture.

Graphical user interface, application, table

Description automatically generated

**6. Simulations:**

The top level test bench can be found in <https://github.com/lixali/FPGA_PPR/blob/master/FPGA/diffusion_rw/top_dual_tb.v> . If we are given a following simple graph, we can get the correct simulations using the test bench. The graph txt file in “https://github.com/lixali/FPGA\_PPR/tree/master/FPGA/diffusion\_rw/input\_bram\_table\_testbench” are exactly in the format discussed in section 2 and 3.







