Advanced Computer-Aided VLSI System Design
Homework 2: Local Binary Pattern

TA: 江承恩 r13943008@ntu.edu.tw

Due Tuesday, Apr. 15, 13:59

TA: 蔡岳峰 f12943014@ntu.edu.tw

Data Preparation

1. Decompress 1132_hw2.tar with following command

```
tar -xvf 1132_hw2.tar
```

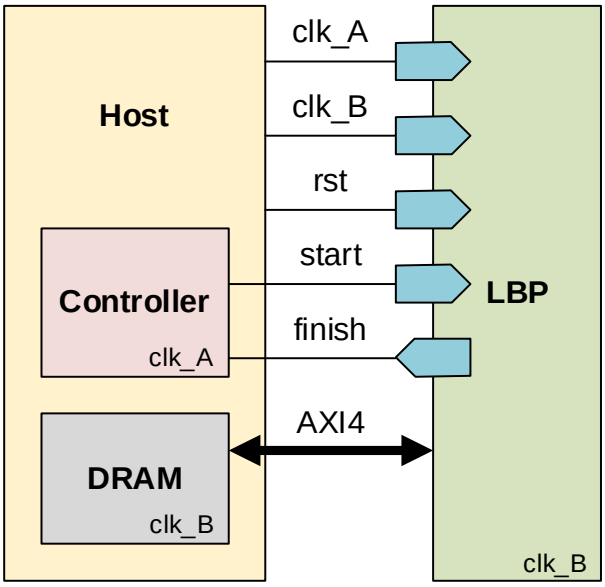
Folder	File	Description
00_TESTBED	testfixture.v	Testbench for LBP
00_TESTBED/pattern	pattern1.dat	Input image data
00_TESTBED/pattern	golden.dat	Output golden
00_TESTBED/axi	*	AXI related file
01_RTL	filelist.f	File list for RTL simulation
01_RTL	01_run	RTL simulation bash file
01_RTL	LBP.v	Your design
02_SYN	filelist.v	File list for synthesis
02_SYN	02_run	Synthesis bash file
02_SYN	syn.tcl	TCL script for synthesis
02_SYN	LBP.sdc	Constraint for synthesis
03_GATE	filelist.f	File list for gate level simulation
03_GATE	03_run	Gate level simulation bash file

All libraries needed for synthesis, simulation can be found in previous homework. **Only worst-case library is used in this homework.**

Introduction

In this homework, you are asked to design a **Local Binary Patterns (LBP)**, which can be used to describe local texture features. The input is a grayscale image stored in the memory on the Host side. The LBP must fetch the grayscale image data from the memory via AXI4 and then perform independent calculations for each pixel in the grayscale image. The computation results are then written back to the memory via AXI4. The start and finish signals are cross-domain signals, and after **the finish signal is asserted high for 3 cycles**, the system will automatically compare the correctness of the entire image data.

Block Diagram



Specifications

1. Top module name: **LBP**
2. Input/output description:

Signal Name	I/O	Width	Simple Description
clk_A	I	1	Clock for control signal (positive edge trigger). Inputs delays **positive** edge clock by **1ns**. Outputs should be synchronized at clock **rising** edge.
clk_B	I	1	Clock for design and axi channel (positive edge trigger). Inputs are synchronized with the **positive** edge clock. Outputs should be synchronized at clock **rising** edge.
rst	I	1	Active **high** asynchronous reset.
start	I	1	Input start signal.
finish	O	1	Output finish signal. The signal should **be asserted high for three cycles** to indicate the end of computation.
data_awaddr data_araddr	O	15	AXI Write / Read request channel. Address of first transfer in a transaction.
data_awlen data_arlen	O	8	AXI Write / Read request channel. Total number of transfers in a transaction.
data_awsize data_arsize	O	3	AXI Write / Read request channel. Maximum number of bytes in each data transfer within a transaction.

data_awburst data_arburst	O	2	AXI Write / Read request channel. How the address increments between transfers in a transaction.
data_awvalid data_arvalid	O	1	AXI Write / Read request channel. Write / Read request valid indicator.
data_awready data_arready	I	1	AXI Write / Read request channel. Write / Read request ready indicator.
data_wdata data_rdata	O/I	8	AXI Write / Read data channel. The actual data signal.
data_wlast data_rlast	O/I	1	AXI Write / Read data channel. Indicates the last write data transfer of a transaction.
data_wvalid data_rvalid	O/I	1	AXI Write / Read data channel. Write / Read data valid indicator.
data_wready data_rready	I/O	1	AXI Write / Read data channel. Write / Read data ready indicator.
data_wstrb	O	1	AXI Write data channel. Indicates which byte lanes of write data contain valid data in a write transaction.
data_rresp	I	2	AXI Read data channel. Response for transactions on the read channels.

3. Only worst-case library is used for synthesis.
4. The synthesis result of data type should NOT include any latch.
5. The slack for setup time should be non-negative.
6. No timing violations or glitches in the gate-level simulation after reset.

Design Description

1. Image Memory Mapping

The image has a fixed size of 128×128 pixels, with each pixel represented by an 8-bit value (ranging from 0 to 255). This image is stored in the provided RAM that can be communicated through AXI4 protocol at address 0 to 16383. The mapping between the image and RAM is illustrated in Figure 1.

2. Local Binary Pattern (LBP) Memory Mapping

The output Local Binary Pattern image also has a fixed size of 128×128 pixels, with each pixel represented by an 8-bit value. You should store the image back to the provided RAM with AXI4 protocol at address 16384 to 32767. According to the specification, **the outermost layer of pixels must be set to 0**. The mapping between LBP processing result and RAM s illustrated in Figure 2.

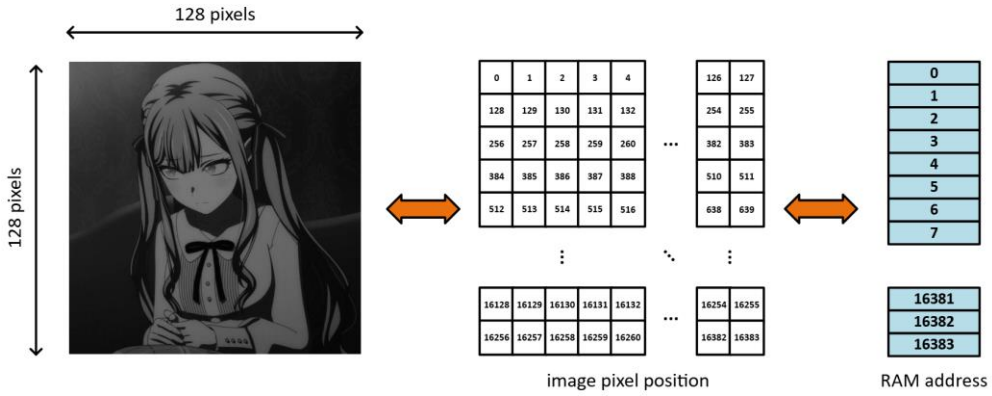


Figure 1: Image Memory Mapping

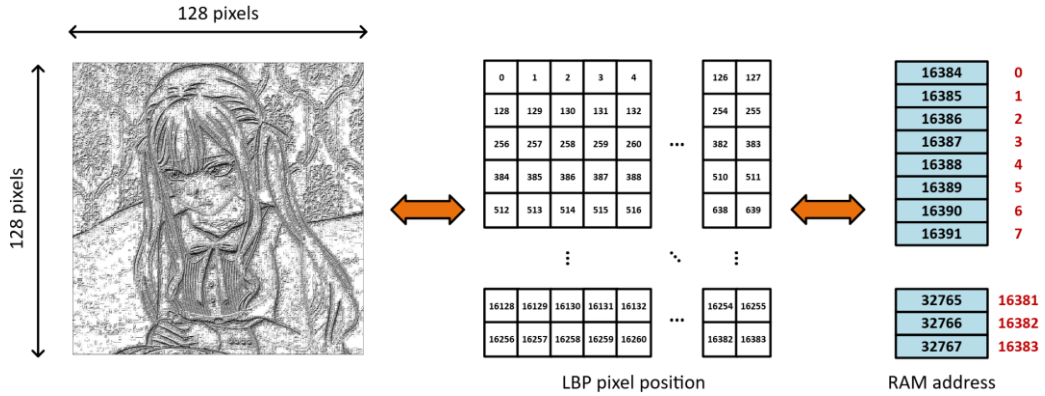


Figure 2: LBP Memory Mapping

3. LBP introduction

The LBP encoding method calculates the relationship between each pixel and its neighboring pixels. As shown in Figure 3, the value of the center pixel in the region is g_c, with the value of the neighboring pixels being g_p, where p = 0,1,...,P-1. In this homework, the region is defined as a 3 × 3 square area expanding one pixel outward from the center (as shown by the green box in Figure 3). Therefore, each g_c has 8 neighboring pixels (P = 8).

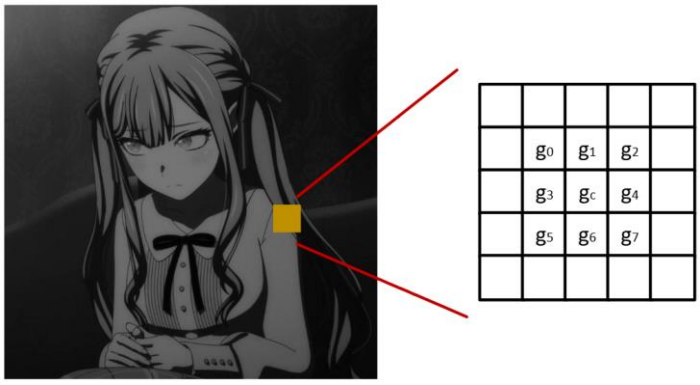


Figure 3: LBP process region

If the coordinate of g_c is (x, y), the LBP value is calculated as follows:

$$\mathrm{LBP}(x,y) = \sum_{p=0}^{P-1} s(g_p - g_c) \cdot 2^p, \quad s(z) = \begin{cases} 1, & z \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

For example, as shown in Figure 4, the center pixel g_c is located at the black circle position within the green box. By applying the formula above, the threshold values s(z) for each g_p are obtained (as shown in the red box in Figure 4).
Multiplying each s(z) value by its corresponding weight 2^p (as shown in the blue box in Figure 4) results in the values in the purple box. Finally, summing all the values in the purple box gives the LBP result for that region.

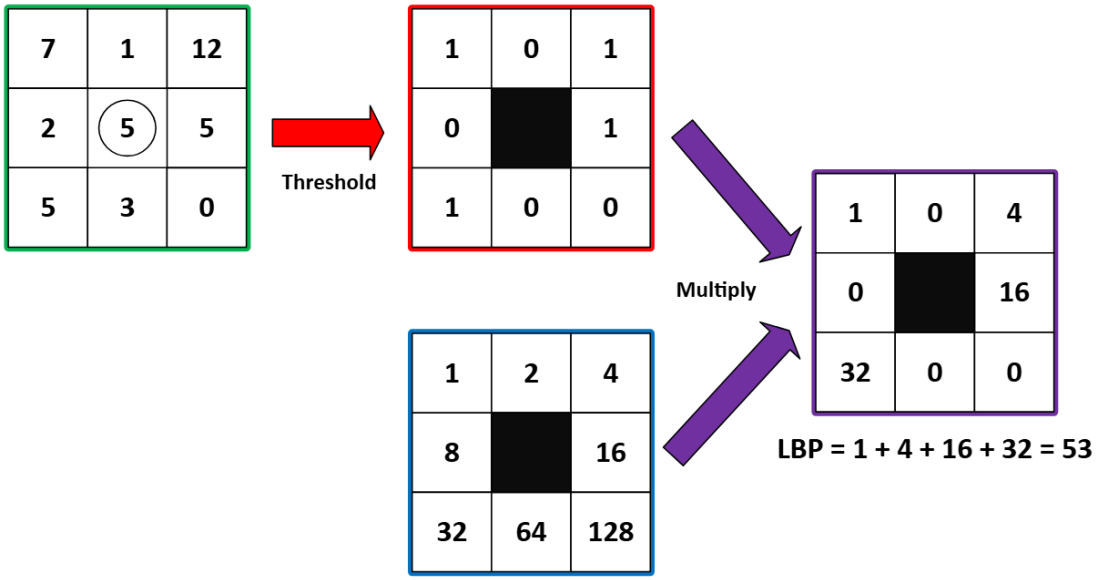
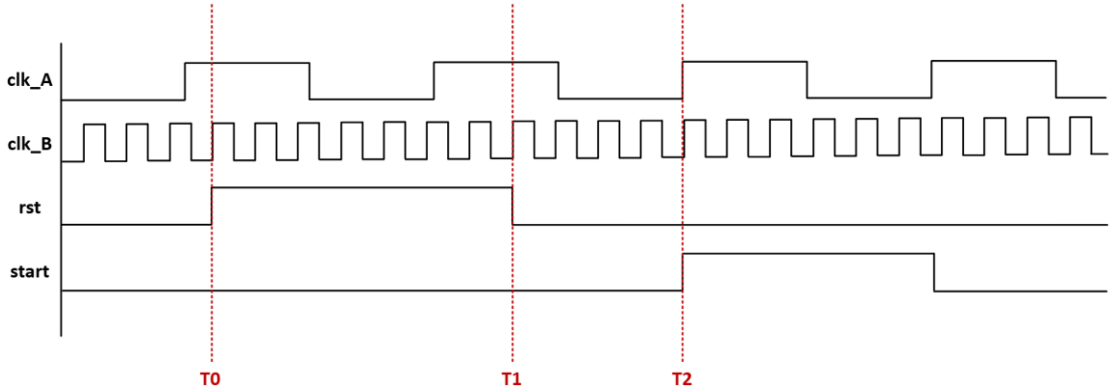


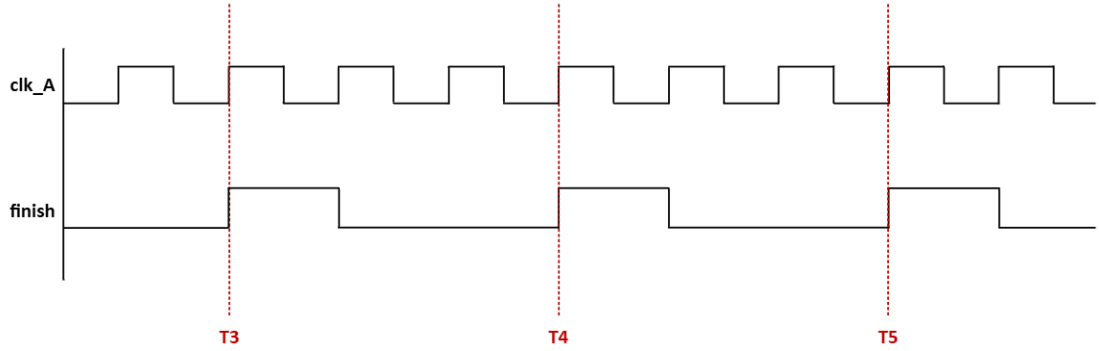
Figure 4: LBP computation example

Notice that the outermost ring of pixels in the grayscale image would not undergo LBP computation, and these pixels must be stored as 0 in the memory. To simplify the problem, the host will initialize all values in memory to 0 before processing begins.

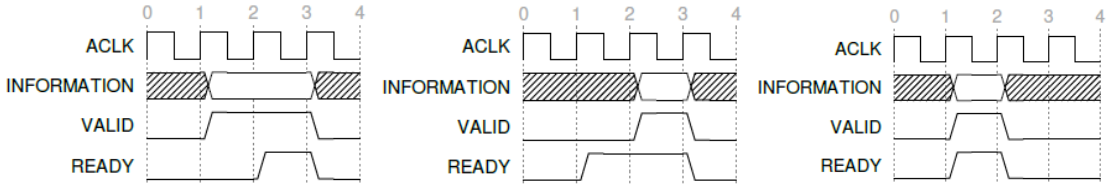
Timing Diagram



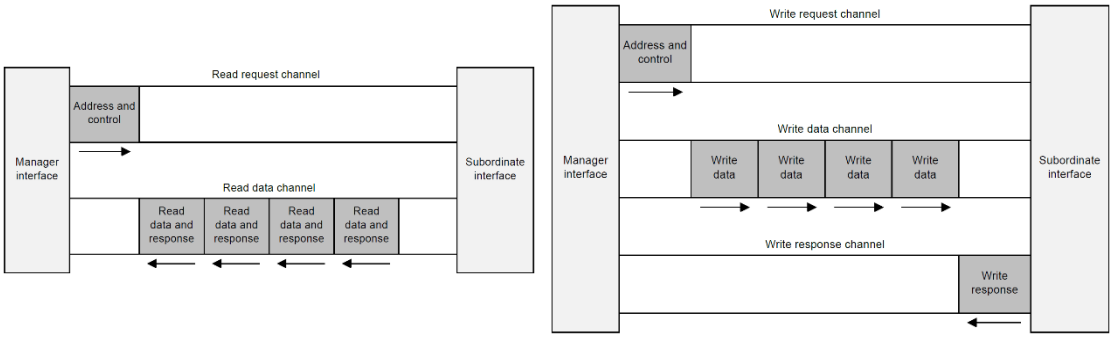
1. LBP is initialized between T0~T1. Registers for both clk_A and clk_B domain can share the same reset signal. No need to consider reset synchronizer.
2. Start is set to high at T2. Notice that this signal is synchronized with clk_A. You should perform CDC to run under clk_B in your design.



3. You are required to finish all your computation for LBP and store the result back to AXI memory before T3. After that, the finish signal should be set to high for three times. The finish signal is synchronized with clk_A, that is, you should also perform CDC as your finish signal is given by LBP module who runs under clk_B.
4. Once finish signal is set to high for three times at T5, testbench will check the result in AXI memory with golden answer.
   Next, let's review how AXI protocol's timing diagram looks.



5. You have three ways to perform handshake with READY and VALID signal. However, you are **not permitted to wait until READY is asserted before asserting VALID**. On the other hand, you are permitted to wait until VALID is asserted before asserting READY. All in all, you cannot set VALID according to READY, but you can set READY according to VALID.



6. Above are the AXI transaction of read and write (response channel is not required in this homework, but TA will be happy if you do it). As shown in the graph, **you are required to transfer address and control through request channel, then you are able to transfer data through data channel**. In course, you should be taught that it's ok to transfer write data before transfer write address. This is true according to AXI protocol spec, but the AXI memory used in this homework requires to transfer write address first, so please follow the instruction above. Just keep in mind that what you've learned are correct.

Hint

1. You are recommended to separate your LBP design and AXI memory access control into different modules. By doing so, your hardware function and bus control can be written independently and such control module can still be used in future works that needs to communicate through AXI.
2. CDC should be perform at the topmost module so start and finish signal can be correctly input and output.
3. Sending multiple data at a time is possible using AXI protocol if you use **len** port correctly.
4. The sdc file is not completed. Remember to add constraint for CDC to get correct synthesis result.

Submission

1. Create a folder named **studentID_hw2**, and put all below files into the folder

```
r13943008_hw2
    ├── 01_RTL
    │       ├── LBP.v      (along with other verilog files)
    │       └── filelist.f (include all your verilog files)
    ├── 02_SYN
    │       ├── LBP.area
    │       └── LBP.timing
    ├── 03_GATE
    │       ├── LBP_syn.sdf
    │       └── LBP_syn.v
    └── report.txt
```

Note: Use **lower case** for the letter in your student ID. (Ex. r13943008_hw2)

2. Content of the report.txt
   Record the processing time and area of gate-level simulation.

```
StudentID: r13943008
Area: 1725.027870 (um^2)
Total Runtime: 3097640.00 ns
```

3. Pack the folder **studentID_hw2** into a **tar.gz file** named **acvsdxxx-hw2.tar.gz.** TA will only check the last version after the homework deadline.

```
tar -zcvf acvsdxxx-hw2.tar.gz [path to studentID_hw2]          (ADFP)
```

```
tar -zcvf acvsdxxx-hw2-vk.tar.gz [path to studentID_hw2]      (NTU COOL)
```

Note:

a. Use **lowercase** for all the letters. (e.g. acvsd000-hw2.tar.gz)
b. Remember to add vk (k is the number of version, k =1,2,…) for files submitted to NTU COOL. (e.g. acvsd000-hw2-v1.tar.gz)
c. Pack the folder on ADFP server to avoid OS related problems.

4. Submit to
   i. NTU cool.
   ii. Place the tar.gz file at the root of your ADFP account.

Grading Policy

1. TA will use **01_run** and **03_run** to run your code at RTL and gate-level simulation.

2. Simulation (70%)

	Score
RTL simulation	30%
Gate-level simulation	30%
Hidden pattern (Gate-level)	10%

3. Performance (30%)
   Score for performance to be considered:

   Score = Time*Area

   Unit: Time(ns), Area(um^2)

   Baseline = 10^{10}

	Score
Baseline	10%
Ranking (Need to pass Baseline)	20%

4. Precise explanation of the terms **Area and Time**:
   - Area: Cell area from synthesis report (ex. 7419.740190 um^2 below)

```
Number of ports:                          504
Number of nets:                          5413
Number of cells:                         5017
Number of combinational cells:           4431
Number of sequential cells:               574
Number of macros/black boxes:               0
Number of buf/inv:                       2557
Number of references:                       9

Combinational area:               1156.343059
Buf/Inv area:                      573.868803
Noncombinational area:             563.397131
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  1719.740190
Total area:                 undefined
```

   - Time: processing time from simulation (ex. 3097640.00ns below)

```
---------------------------------------------------

Congratulations! All data have been generated successfully!

Total cost time: 3097640.00 ns
-------------------------PASS-------------------------
```

Reference

[1] IC Design Contest, 2016.

[2] AMBA AXI Protocol Specification.

[3] R. Ginosar, Metastability and Synchronizers: A Tutorial, 2011.

[4] Clifford E. Cummings, Clock Domain Crossing (CDC) Design & Verification Techniques Using SystemVerilog, 2008.