Advanced Computer-Aided VLSI System Design
Homework 1: Run-Length Encoder with Low Power Design

TA: 葉星宏 r12943125@ntu.edu.tw **Due Tuesday, Apr. 1st, 13:59**

TA: 蔡岳峰 f12943014@ntu.edu.tw

Data Preparation

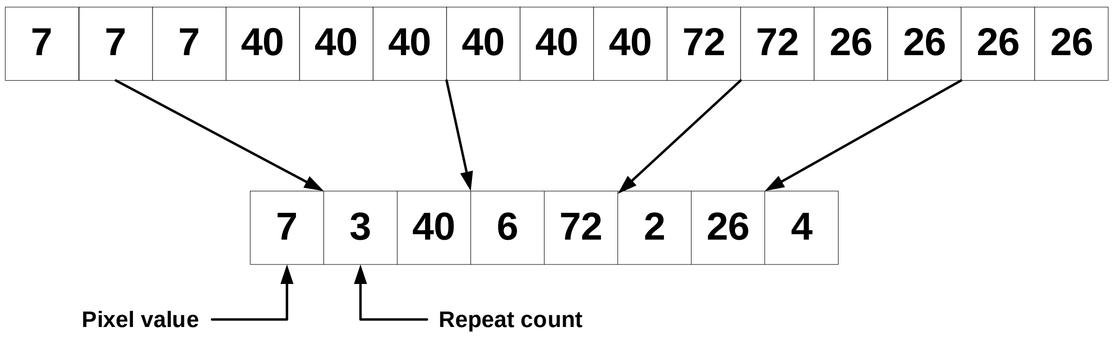
1. Unpack 1132_hw1.tar with the following command

```
tar -xvf 1132_hw1.tar
```

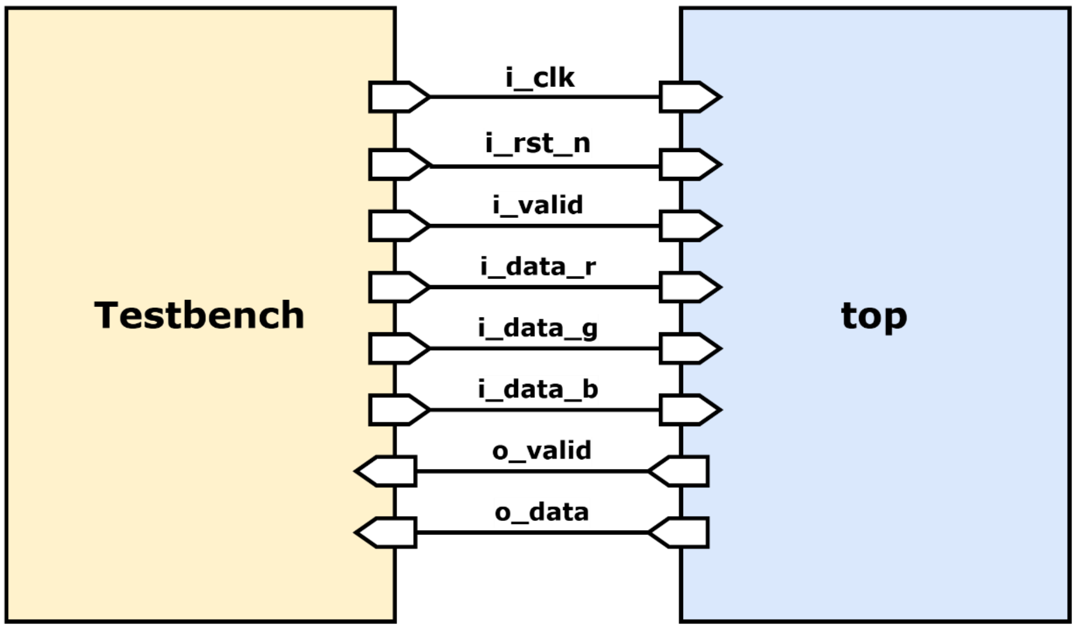
Folder	File	Description
00_TESTBED	testbench.v	File to test your design
00_TESTBED /p0~p4	pat*.dat	Input instruction patterns
	gold*.dat	Output golden patterns
01_RTL	top.v	Your design
	rtl_01.f	File list for RTL simulation
	01_run	VCS command for simulation
	02_run_UPF	VCS command for simulation
02_GATE	03_gatesim	VCS command for simulation
	04_gatesim_UPF	VCS command for simulation
	rtl_02.f	File list for RTL simulation

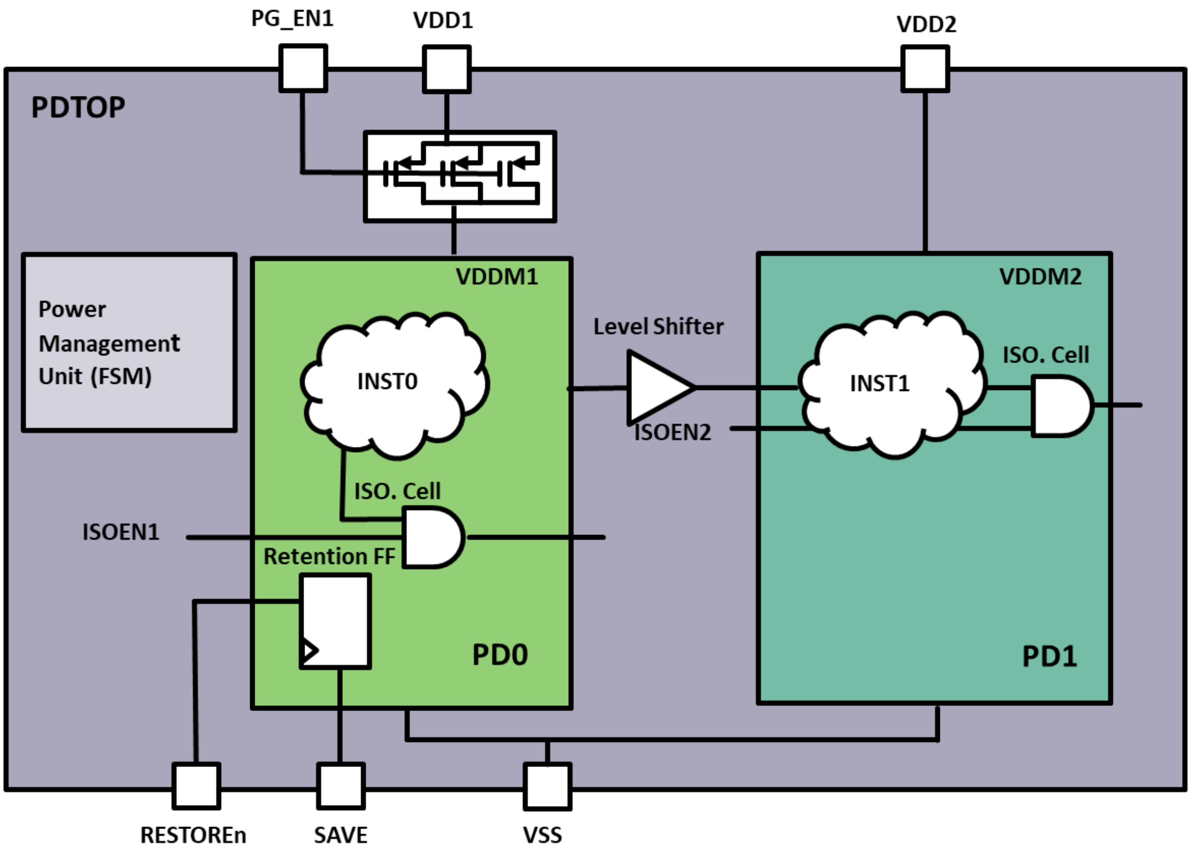
Introduction

Run-length encoding (RLE) is a form of data compression in which runs of data (consecutive occurrences of the same data value) are stored as a single occurrence of that data value and a count of its consecutive occurrences, rather than as the original run. In this homework, you are going to design a simplified Run-length Encoder and implement it with low-power design constraint through UPF flow.



Block Diagram





Specifications

1. Top module name: top
2. Input/output description:

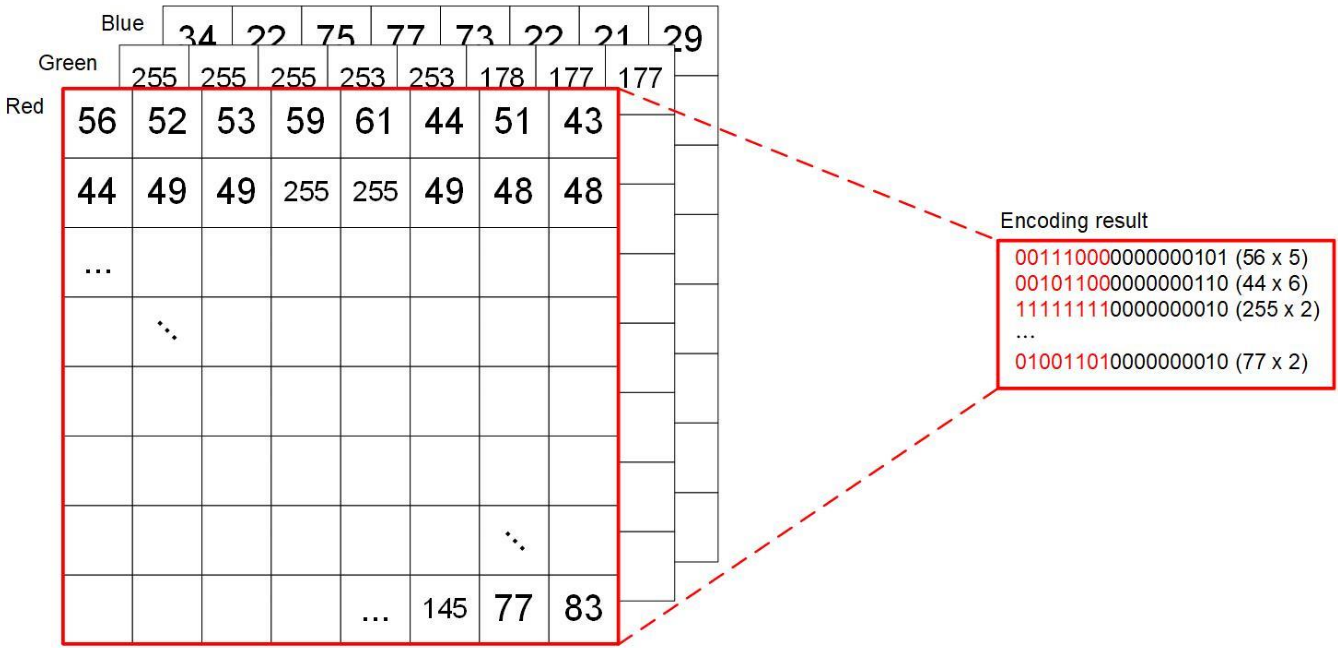
Signal Name	I/O	Width	Simple Description
i_clk	I	1	Clock signal in the system All inputs are synchronized with the negative edge clock All outputs should be synchronized at clock rising edge
i_rst_n	I	1	Active **low** asynchronous reset
i_valid	I	1	The signal is **high** if input data is ready
i_data_r	I	8	Unsigned integer input data which represent the red, green, blue value of each pixel from 0 to 255
i_data_g	I	8	
i_data_b	I	8	
o_valid	O	1	Set **high** if ready to output result
o_data	O	18	Unsigned output data after run-length encoding 1. For o_data[17:10], output the pixel value of each run 2. For o_data[9:0], output the length of the value

3. Active low asynchronous reset is used only once.
4. All inputs are synchronized with the **negative** clock edge.
5. All outputs should be synchronized with the **positive** clock edge. That is, flip-flops should be added before all outputs.
6. New pattern (i_data_r, i_data_g and i_data_b) is ready only when i_valid is high.
7. o_valid should be pulled high for only one cycle for each o_data.
8. The testbench will sample o_data at **negative** clock edge if o_valid is high.
9. You can raise o_valid at any moment **but only once**.

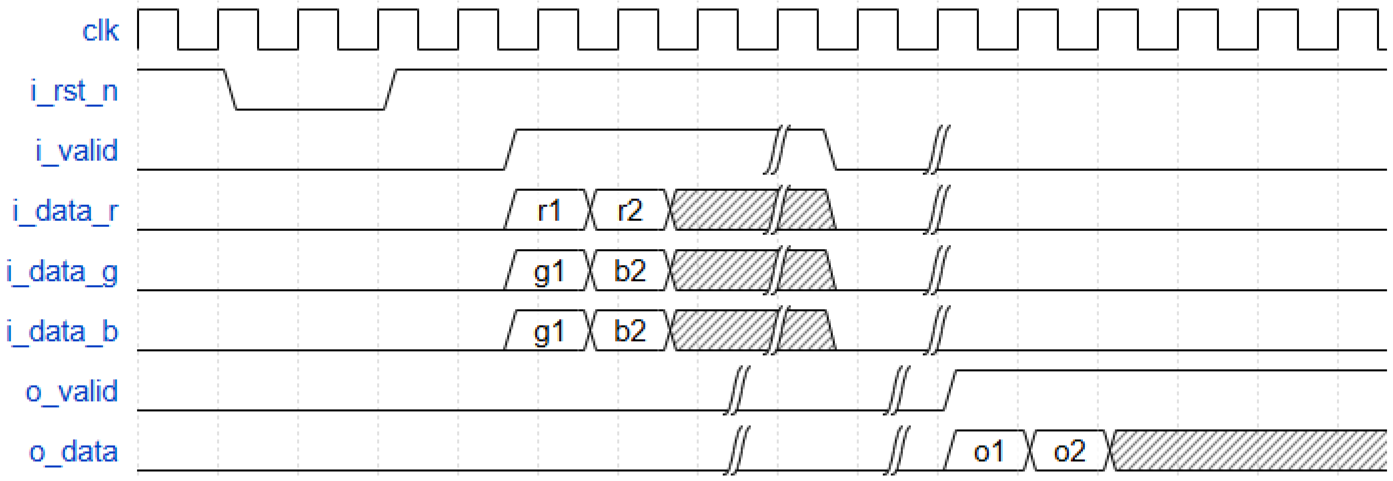
Design Description

1. Lossy RLE, a value is first picked as the values reference value, and values between a certain range, instead of the exact same value, are stored in one run.
2. The sizes of all input images are 64x64x3 pixels.
3. The values in each channel should be considered separately. For example, the last pixel value in the red channel and the first pixel value in green channel cannot be merged into the same run.
4. Input data are separated by channel, while output data are serial with the order of red, green, and blue.
5. The number of encoded data in each channel won't exceed 1536.

6. The range threshold value is set to **10**.
7. **At least one SRAM** is implemented in your design.
8. At least **two power domains** (including top) are used in your design.
9. At least **one power switch** is used in your design.
10. Power top should be set in the UPF file.
11. You are **NOT** allowed to use DesignWare.

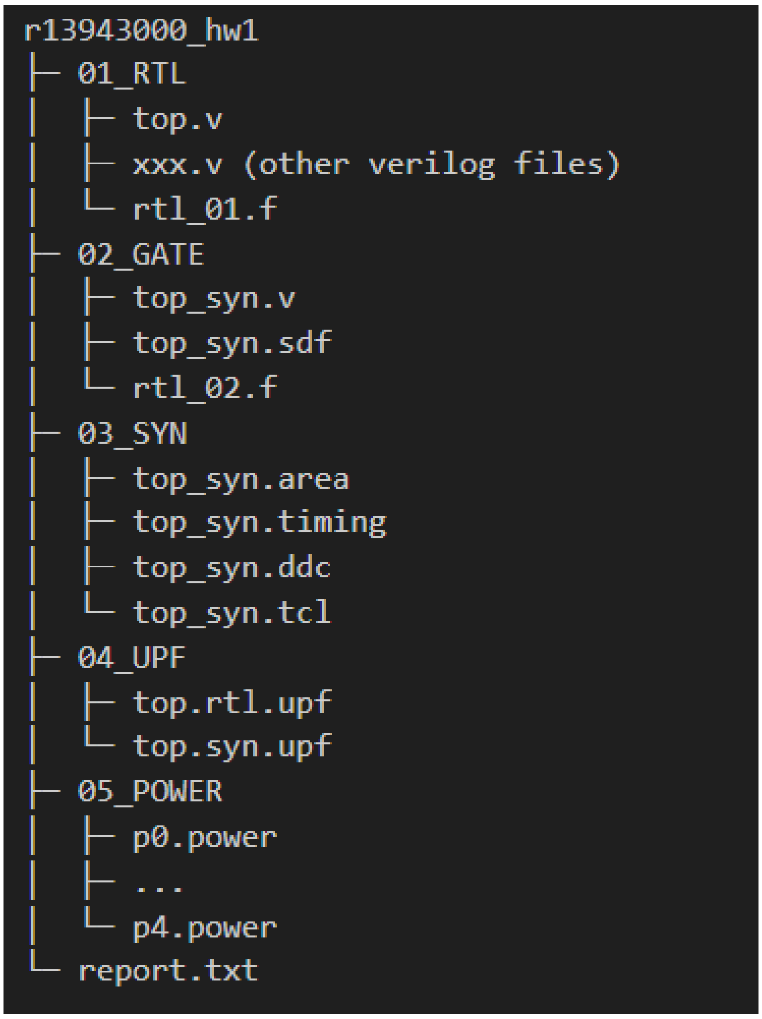


Sample Waveform



Submission

1. Create a folder named **studentID_hw1** and follow the hierarchy below.



```
r13943000_hw1
├ 01_RTL
│  ├ top.v
│  ├ xxx.v (other verilog files)
│  └ rtl_01.f
├ 02_GATE
│  ├ top_syn.v
│  ├ top_syn.sdf
│  └ rtl_02.f
├ 03_SYN
│  ├ top_syn.area
│  ├ top_syn.timing
│  ├ top_syn.ddc
│  └ top_syn.tcl
├ 04_UPF
│  ├ top.rtl.upf
│  └ top.syn.upf
├ 05_POWER
│  ├ p0.power
│  ├ ...
│  └ p4.power
└ report.txt
```

Note: Use **lowercase** for all the letters. (e.g. r13943000_hw1)

2. Pack the folder **studentID_hw1** into a **tar.gz file** named **acvsdxxx-hw1.tar.gz**. TA will only check the last version after the homework deadline.

```
tar -zcvf acvsdxxx-hw1.tar.gz [path to studentID_hw1]            (ADFP)
```

```
tar -zcvf acvsdxxx-hw1-vk.tar.gz [path to studentID_hw1]      (NTU COOL)
```

Note:

a. Use **lowercase** for all the letters. (e.g. acvsd000-hw1.tar.gz)

b. Remember to add vk (k is the number of version, k =1,2,…) for files submitted to NTUCOOL. (e.g. acvsd000-hw1-v1.tar.gz)

c. Pack the folder on ADFP server to avoid OS related problems.

3. Submit to

i. NTU Cool

ii. Place the tar.gz file at the root of your ADFP account.

Grading Policy

1. TA will run your code with the following format of command. Make sure to run this command with no error message on ADFP server.
   RTL simulation:

```
vcs -full64 -R +v2k -sverilog -f rtl.f -debug_access+all +define+$1
```

   Gate-level simulation:

```
vcs -full64 -R +define+SDF +v2k -f rtl.f -sverilog -v2005 -
debug_access+all +maxdelays -negdelay +neg_tchk +define+$1
```

   For VCS NLP, additional arguments will be added

```
-upf [path to UPF file] +define+UPF
```

2. Pass all the test pattern (5 public and 5 private) to get full score.
   - Simulation: **70%**

	Score
RTL simulation	15%
RTL with UPF	15%
Gate-level simulation	20%
Gate-level with UPF	20%

   - Performance: **30%**
     - Score for performance to be considered:

$$\textbf{Score} = \sum(P^2 \times T) \times A$$

P: Power (mW), T: Simulation time (ns), A: Area (um^2)

     - Only if you pass all patterns will you get the score.
     - Baseline = 3×10^6

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

3. Lose **5 points** for any incorrect naming or format.
   - It is your responsibility to ensure that the files can be correctly unpacked and executed on IC Design LAB server.
4. No late submission
   - 0 point for this homework
5. No plagiarism
   - Plagiarism in any form, including copying from online sources, is strictly prohibited.

References

1. Reference for run-length encoding
   https://medium.com/@ishifoev/run-length-encoding-rle-algorithm-step-by-step-guide-b0b89f3a4a9f
2. Reference for UPF
   Synopsys ® Multivoltage Flow User Guide