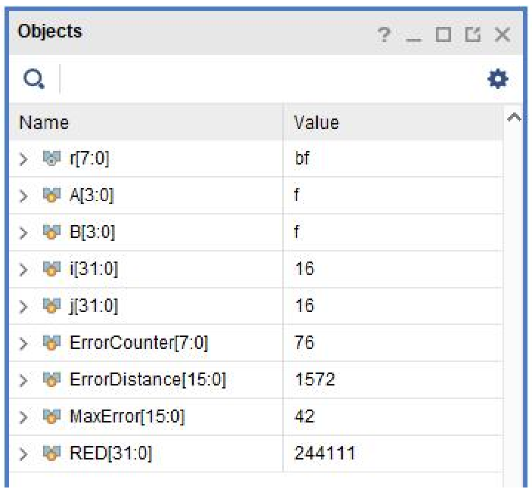
This paper proposed FPGA-based approximate multipliers via modifying LUT primitives.

For AE, there are three main points:

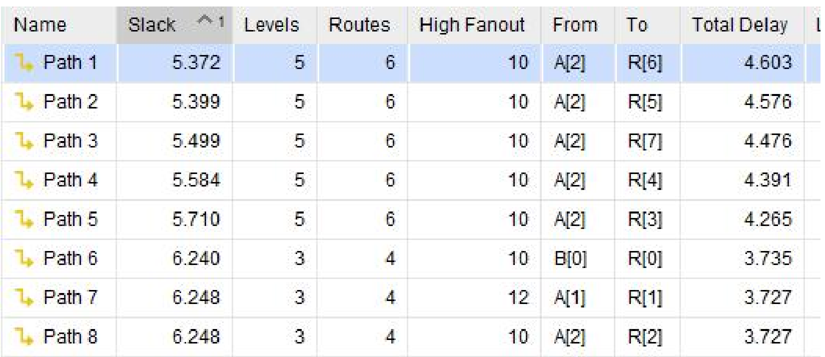
First is the error analysis, the testbench for evaluation is included in my code. The sizes of approximate multipliers have 4x4 and 8x8. The testbenches for the two sizes are different. For example, when running 4x4 multiplier testbench, the object box may like this:



Name	Value
r[7:0]	bf
A[3:0]	f
B[3:0]	f
i[31:0]	16
j[31:0]	16
ErrorCounter[7:0]	76
ErrorDistance[15:0]	1572
MaxError[15:0]	42
RED[31:0]	244111

There are four different error matrix of approximate multipliers, such as ErrorCounter, ErrorDistance, etc. The 8X8 size is the same.

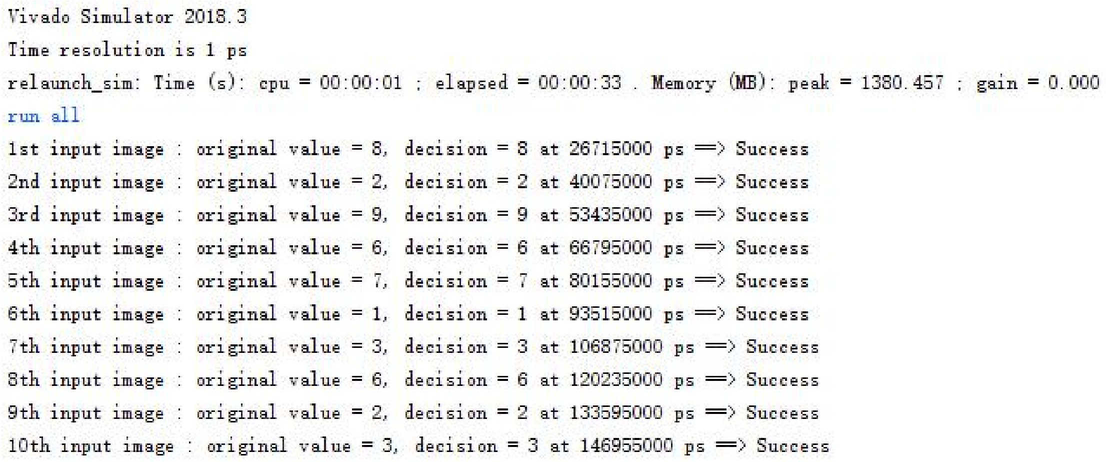
Second is performance, it is completed by Vivado automatically. Just set the multiplier as top and run synthesis. When finish synthesis, it can be seen like this:



Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	L
Path 1	5.372	5	6	10	A[2]	R[6]	4.603	
Path 2	5.399	5	6	10	A[2]	R[5]	4.576	
Path 3	5.499	5	6	10	A[2]	R[7]	4.476	
Path 4	5.584	5	6	10	A[2]	R[4]	4.391	
Path 5	5.710	5	6	10	A[2]	R[3]	4.265	
Path 6	6.240	3	4	10	B[0]	R[0]	3.735	
Path 7	6.248	3	4	12	A[1]	R[1]	3.727	
Path 8	6.248	3	4	10	A[2]	R[2]	3.727	

Last is deployment in application, there are two application, image blending and CNN. Image blending is completed in Matlab, and CNN is implemented in Verilog. The code is also open-source. When running image blending, the preporcessing and comparation of image is in Matlab and pixel multiplication is in Vivado using proposed approximate multipliers. For example, the pictures may like this:

Then is CNN. The weight and bias have been trained in advance. The MNIST is preprocessed in Python for 1000 images. When running CNN in Vivado simulation, it can be seen like this:



```
Vivado Simulator 2018.3
Time resolution is 1 ps
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:33 . Memory (MB): peak = 1380.457 ; gain = 0.000
run all
1st input image : original value = 8, decision = 8 at 26715000 ps ==> Success
2nd input image : original value = 2, decision = 2 at 40075000 ps ==> Success
3rd input image : original value = 9, decision = 9 at 53435000 ps ==> Success
4th input image : original value = 6, decision = 6 at 66795000 ps ==> Success
5th input image : original value = 7, decision = 7 at 80155000 ps ==> Success
6th input image : original value = 1, decision = 1 at 93515000 ps ==> Success
7th input image : original value = 3, decision = 3 at 106875000 ps ==> Success
8th input image : original value = 6, decision = 6 at 120235000 ps ==> Success
9th input image : original value = 2, decision = 2 at 133595000 ps ==> Success
10th input image : original value = 3, decision = 3 at 146955000 ps ==> Success
```

It will run until testing 1000 images finished.

For implement details, please email: Yaoshangshang96@outlook.com