PROJECT REPORT

By:-
Vishal Reddy Burri
Naga Sreevatsava M

Faculty Mentor:-
Dr.Suresh Purini

OVERVIEW

To create an Approximate Drum Multiplier for various Data Type and to integrate this with Harris Corner Detection algorithm to see the difference in Power, Area and resources consumed.

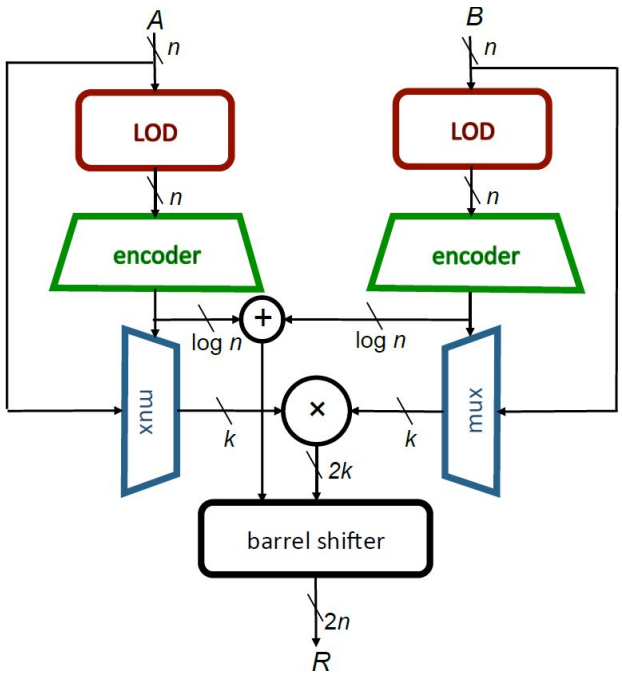
Objectives

1. To integrate DRUM in Harris Corner Detector
2. Compare the results with varying k.

DRUM Multiplier

The Drum multiplier exploits the fact that not all bits of a number are important in multiplication. Therefore we put a limit to the number of the two operand of the multiplier by selecting the range of bits for each of the two operands of the multiplier. By doing so we can reduce the hardware cost, power and also reduces the time taken to multiply giving us an increase in throughput.

The basic design of the DRUM consists of 3 basic parts namely LOD(leading one detector), Core Multiplier, Barrel shifter along with a few encoders. Assuming each of the operand is ‘n’ bits the LOD dynamically locate the most significant ‘1’ in the operand and we select k-2 consecutive bits based on the accuracy required. Here K is a designer-defined value which is related to the resources of the core multiplier and accuracy of the overall DRUM multiplier.



SPECIFICATIONS

We have implemented the multiplier in many versions such as combinational , multi-cycle and pipelined with both the inbuilt and our own implemented multiplier and have compared the results which are almost similar to those of what we expected. Apart from this we have extended the multiplier to unsigned and signed integers successfully by using the inbuilt unsignedMul(function name). We have also extended this multiplier for Fixed Point Multiplication for approximating the fractional part i.e. take k bits from the decimal point for multiplication.

To test as to how much of an impact the multiplier would make, we took the Harris Corner Detector(HCD) and integrated the multiplier into the HCD design. Since HCD is one of the benchmarks in image processing we took it up. Now coming to the integration the HCD consisted of 2 different files namely ‘Mac.bsv’ and ‘HarrisCornerMultirate.bsv’. Now each file called the multiplier quite a few times. Here as ‘Mac.bsv’ didn’t need to require as good of an accuracy we implemented with a K value being low(K2 mentioned in Results) and as ‘HarrisCornerMultirate.bsv’ required a better accuracy we took the K in this to be a bit high(mentioned as K1 in the result).

Now we tested the entire system with varying the variables and looked into what were the trends in the results which we obtained and will be discussed in detail at a later section. Also we used a pipelined version so that we would get the best savings possible. Also we are exploring different LOD implementations along the side, since our LOD may or may not be the best possible implementation of it and as it would have a great effect on the design of the DRUM. So in case we stumbled on a better implementation than the one we currently have we would be including that in the future which would lead to even better results than the ones currently obtained.

Challenges Faced

Implementing the LOD

The implementation of an efficient LOD was a big task as it is the core of the DRUM multiplier. Initially due to a bad design the timing of the circuit was getting affected a lot leading to bad results. Later with the implementation of a better design, we were able to see the results we expected.

Pipelining the circuit

Due to the various conditions imposed on the implemented LOD, it was a slightly trickier task to make the circuit pipelined

Implementing for Fixed point

For Fixed point multiplication we faced a lot of issues , such as how to carry on the approximation in case of fixed points, and also consider sign of the numbers in this case.

Integrating the Multiplier

Initially we went with a combinational multiplier to test a few basic things about the code but while integrating the fractional multiplier we had come across various interface issues due to which we had to rewrite a pieces of codes so that the inputs given by the HCD were correctly taken up by the multiplier.

FILES

1. **Signedcombdrum:- (Signed Combinational DRUM multiplier)**
    a. Drum.bsv (Main file which contains implementation of DRUM)
    b. Multiplier.bsv (Interface file)
    c. MultiTb.bsv (Testbench file)
    d. Test.py (test for accuracy and max error)

2. **SignPipelineddrum :- (Signed Pipelined DRUM multiplier)**

    a.Drum.bsv (Main file which contains implementation of DRUM)

    b.Multiplier.bsv (Interface file)

    c. MultiTb.bsv (Testbench file)

    d. Test.py (test for accuracy and max error)

3. **Fixedpointsignedcombdrum :-(Signed FixedPoint DRUM multiplier)**

    a.Drum.bsv (Main file which contains implementation of DRUM)

    b.Multiplier.bsv (Interface file)

    c.MultiTb.bsv (Testbench file)

    d.FixedPoint.bsv (Fixedpoint library file)

    e.Test.py (test for accuracy and max error)

4. **HCD :- (Original HCD with accurate multiplier)**

    a.HarrisCornerMultirate.bsv (Main file and used fxpt multiplier)

b.MAC.bsv (fxpt multiplier used in this file)

5. DRUMHCD :- (HCD with Drum multiplier)

a.HarrisCornerMultirate.bsv (Main file and Drum integrated in this file)

b.MAC.bsv(Drum multiplier is integrated in this file)

RESULTS:-

Now we have tested the HCD after integrating our pipelined multiplier with it. And as we have expected we got a significant decrease in power and area. The details of the analysis are given in the table below. Now on the table we can see that at some points there is a decrease in power where generally we would be expecting an increase such as when $K_1 = 10$ and $K_2 = 8$. This is because the Board we simulated the code on, starts using the DSP blocks present which doesn't happen unless the bit length is long enough otherwise it would use LUT's to create the multiplier. This is the reason for such an anomaly. Also we can see that the accuracy results are quite good enough and comparable to that of the 'accurate HCD'.

Also we found that integrating the pipelined version of the multiplier yielded great results as to the 7-8% power savings we got when we integrated the combinational version of the DRUM. Also apart from this if we get a better version of the LOD in the future as mentioned before we would also be seeing a significant difference in the results.

For Harris Corner Detector:-

1) **Power Analysis:-**

Accurate HCD power : 0.945

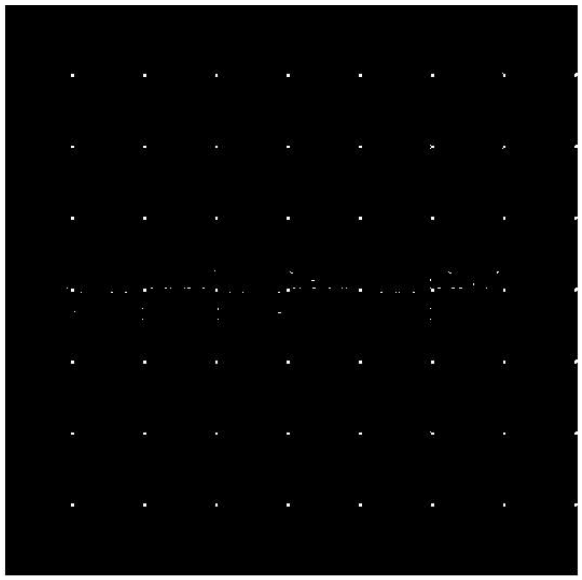
K_1	K_2	Total Power taken	Total Power reduced
8	6	0.658	31%
10	6	0.644	32%
12	6	0.679	29%
15	6	0.691	27%

K_1	K_2	Total Power taken	Total Power reduced
8	8	0.683	28%
10	8	0.668	30%
12	8	0.700	26%
15	8	0.717	25%

K_1	K_2	Total Power taken	Total Power reduced
8	10	0.718	24%
10	10	0.696	26%
12	10	0.716	25%
15	10	0.727	24%

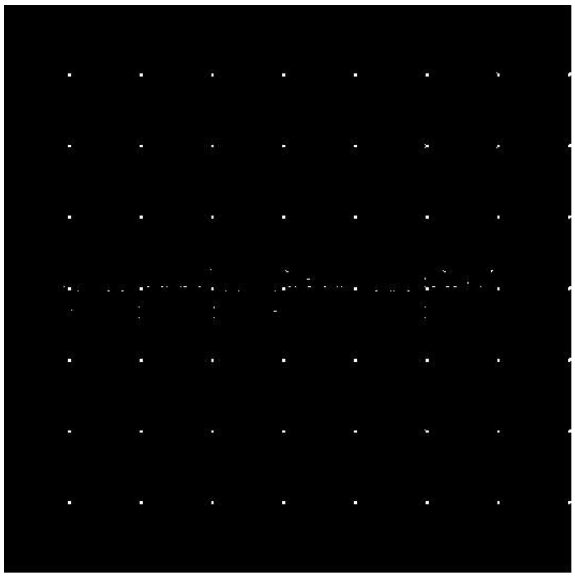
2) Accuracy Analysis :-

HCD Accurate



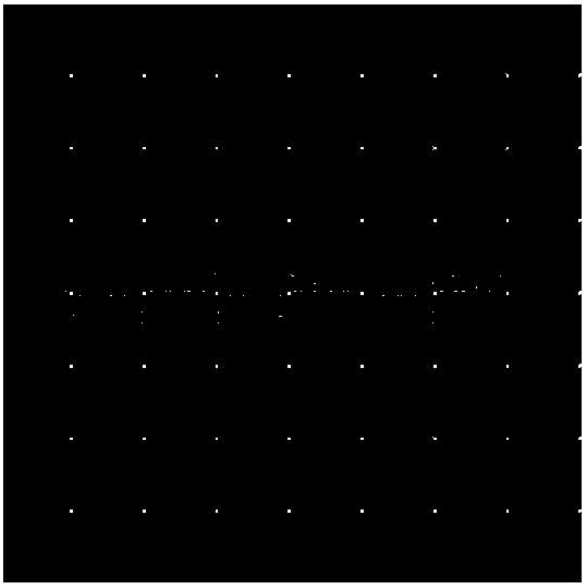
k1=10,k2=10

Accuracy is 99.9993687028 %



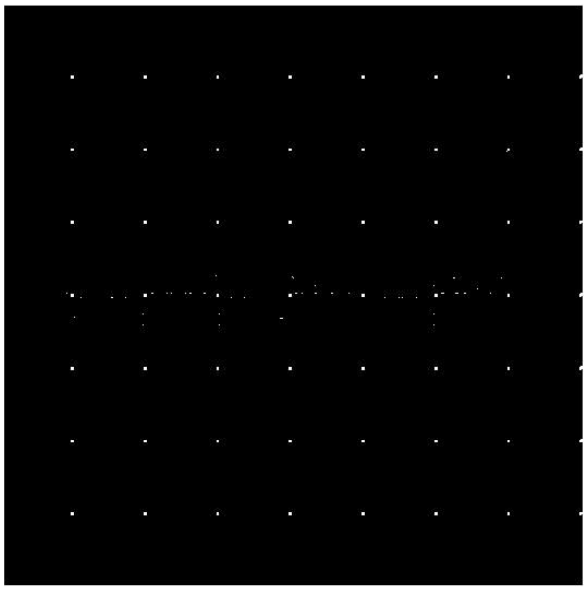
k1=8,k2=8

Accuracy= 99.9966856898 %



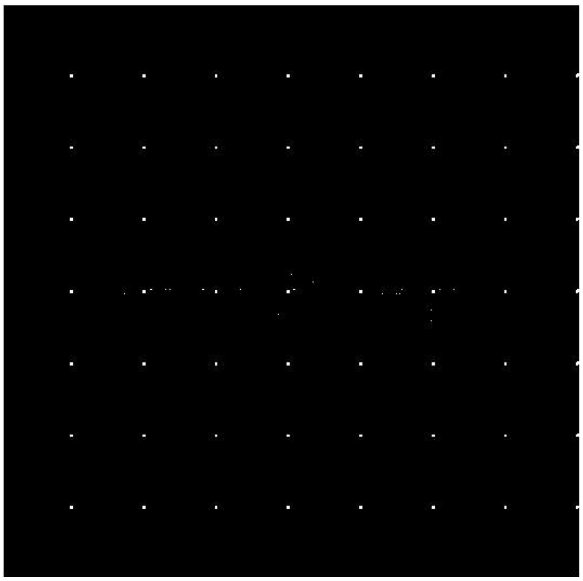
k1=6,k2=6

Accuracy= 99.9933713795 %



k1=4,k2=4

Accuracy= 99.9820080301 %



k1=2,k2=2

Accuracy= 99.9767998283 %