EE 271 VLSI DESIGN
FALL 2009

PROJECT REPORT
MICROPOLYGON RASTERIZATION

KARTHIKA PERIYATHAMBI
05590444
ADITI GOYAL
05591220

Electrical Engineering
Stanford University

INDEX

ABSTRACT

Modifications implemented:

- Reduction in number of samples:

1. Back Face Culling: (Triangles and Rhombus type Quadrilaterals)
2. Optimal Bounding Box: (Ceiling)
3. Triangle Merging

- Reduction in idle states:

1. Deleting idle HALT in FSM
2. Bubble Smashing
3. Scanning Valid Polygons only

- Synthesis improvements (from previous part)

1. Bitwise Comparator
2. Modified Booth with parallelism in partial product generation

Design Statistics

DESIGN TARGET:
Power : 300mW
Area: 1mm^2
Throughput>500million uPoly/sec

ORIGINAL DESIGN VALUES(per instance) :
Clock 1.05nS
Design Power is 12.44 mW
Design Area is .034 mm^2
Design Throuput is .083 uPoly/cycle or 79 Million uPoly/Sec

OUR DESIGN VALUES(per instance):
Clock 0.8Ns
Design Power is 15.0729 mW
Design Area is 0.027 mm^2
Design Throughput is 0.175 uPoly/cycle or 219 Million uPoly/Sec

THE THOUGHT PROCESS

We have tested all our improvements on a data of 145 polygons as that simulates faster which has been included in the $USER.tar.gz as sample_test_145.dat. The verif/tests/ folder also contains the same data under the name sample_test.dat. This is because the 200000 poly file is not suitable for debugging and quick checks.

Initial output:

Cycles: 1025 uPoly: 145 sampleTests: 870 sampleHits: 129

After understanding the code, we started with backface culling for triangles to remove the invalid triangles before they consume their time in sampletest.v. This would be a great improvement in number of samples.

This led us to implement back face culling for quadrilaterals; the plain rhombus types. Though we figured out the logic for bowtype and convex, keeping in mind the smaller number of these cases occurring, we moved onto next step.

Next step was to remove the cycles by removing the Halt idle cycle as that would cut down the cycles by no. of polygons, a huge improvement for bigger samples.

Having done BackFace Culling, we thought it would be more effective to remove bubble smashing. So, we tried playing with the halt signals of starting flops. But, then we realized that input is still not changing implying change implying change in rast_driver.v

So, we shifted backface culling from validPoly_R12H stage to validPoly_R11H stage, so that we can export the signal to rast_driver.v and use it there to input a new Polygon when needed.

Having entered the domain of rast_driver.v, we thought why not sample till you get a valid Polygon. First we tried to sample two polygons at a time to eliminate the case of 101. Later we used a while loop to eliminate cases of continuous zeros.

On viewing the vector.dat generated, we found there were lot of triangle cases whose edges were common. So, we decided to use the previous logic of two samples to merge triangles with common edge. That helped a lot in bigger cases.

Having done the above, we wanted to optimize the bounding box, by modifying the flooring done already. So, we implemented the ceiling function.

Moving onto synthesis, we used our previous bitwise comparator and modified booth files to improve synthesis timings.

IMPROVEMENTS

- Reducing number of Samples

1. ***Back Face Culling***:

There are a number of polygons which are facing backwards. The current design lets all of them through and only culls them at the end of the sampletest.v file. This means we unnecessarily calculate the time consuming partial distances values for all the samples for these kind of polygons also. Hence we modified the code to declare the polygon as invalid if it was backfacing at the end of bounding box stage.

To find out if its back facing we calculate the normal to the polygon.

a) ***TRIANGLE***: In case of a triangle, we reject the polygon if the z coordinate of the normal is greater than 0.

b)***QUADRILATERAL***: For a quadrilateral, we have implemented culling of rhombus kind of quadrilaterals. For other types of quadrilateral, we can implement using same strategy.

Rhombus(Plain): All 4 backfacing triangles

Convex: 3 Backfacing triangle and 1 front facing triangle.

BowType:2 Backfacing triangle and 2 front facing ones.

Here we check if all the four normals for the 4 triangled possible, are greater than 0. Here implementing for convex and bow tie type quads is also a possibility but we figured it would not make too much of a difference to the number of hits as these kind of polygons will be very few in number.

Calculating the surface normal is a computation intensive step and first when we implemented it we found the clock time shooting up to 1.86 ns. This was dealt with by breaking the subtraction and multiplication into two stages. We flop the required subtractions to the 11th stage, where a modified booth encoder is used to perform the multiplication. This improvement restored the timing to 1 ns.

We also made corresponding changes to the gold file so that the two models match and no assertions are thrown.

Improvement after Backface only :

Cycles: 795 uPoly: 145 sampleTests: 639 sampleHits: 128

Relevant file:

bbox_backfaceonly.v

2. ***OPTIMAL BOUNDING BOX*** - Ceiling the lower left corner:

The current design floors the coordinates of both the bounding box top-right and bottom left corners. Clearly we could save some computation by ceiling the bottom right corner. But we need to ceil only if the fractional part has at least 1 one in it. This is to ensure that we dont ceil a coordinate which is right on a sample location. Also the integral part should have at least 1 zero in it from the 23d position onward so that adding a 1 does not cause it to overflow.

Improvement after ceil only:

Cycles: 923 uPoly: 145 sampleTests: 768 sampleHits: 127

After ceil + back face:

Cycles: 753 uPoly: 145 sampleTests: 597 sampleHits: 127

Relevant Files:

bbox_ceilonly.v

bbox_backface_ceil.v

We found that on large data, the ceiling function does not give a very large improvement in number of sampletests performed. On the other hand it reduces number of samplehits significantly. Hence for larger data it would make sense to probably avoid the ceiling function altogether.

3) ***Triangle merging***:

Since we have the ability to process quadrilaterals, hence if we can merge two common sided triangles into quadrilateral, it will achieve the purpose in lesser time. But before merging, the concept of front facing or backfacing has to be checked. Only 'anti-parallel edges' can coincide and according to which edge has merged, the corresponding vertices are rearranged to yield a front facing quadrilateral. In case of no merge, the sampled data is resampled at the next cycle using $fseek and $ftell functions.

Improvements observed:
Cycles: 981 uPoly: 145 sampleTests: 869 sampleHits: 127

Relevant Files:
rast_driver.v

- Number of cycles consumed

1) REMOVING IDLE STATE IN FSM:

Currently, total no. of cycles=Number Of sample tests + number of Upolys. This is because of 1 Waste HALT cycle that exists during every wait state. To make it active, i have used it to calculate the data. The box_R13S[0] is passed twice, but once with validSamp=0. If we can instead pass the next to be sample point and make validSamp=1 correspondingly, then we can cut the idle halt cycle. But, next valid sample has to be calculated keeping in mind; long and thin polygons and just 1 point polygon.

next_sample_R13S = (chk_thin&&chk_short) ? box_R13S[0] : (chk_thin) ? start_sample_up : start_sample_rt;

Also, passing data if polygon is invalid is avoided by incorporating the case into validSamp calculation.

validSamp_R13H = (current_state_R14H == `R_TI_TEST_STATE )||((current_state_R14H == `R_TI_WAIT_STATE) && (validPoly_R13H)&&!(chk_thin&&chk_short));

Improvement in number of cycles after test iterator change only:

Cycles: 860 uPoly: 108 sampleTests: 850 sampleHits: 136

Improvement after backface culling and bounding box optimization and FSM :

Cycles: 679 uPoly: 145 sampleTests: 597 sampleHits: 134

Relevant files:

test_iterator_fsm.v

2) ***BUBBLE SMASHING***:

While pipe flow is halted during iteration for samples of valid Polygon, we can check if the previous flop holds inValid polygon (next in queue). Though invalid Polygons have been removed by rast_driver, backface culling might generate them. Hence, parallely updating the front pipeline needs outsourcing validPoly_R12H from bbox.v through rast.v to rast_driver.v using bench.v By modifying halt condition in bbox.v (using pseudoHalt=halt_RnnnnL | ~validPoly_R12H) and in rast_driver.v; we can smash the bubble.

Relevant Files:

bbox.v

rast.v

rast_driver.v

bench.v

3. ***Scan Valid polygons only***:

We figured that the current code passes all polygons to the bounding box function even if they are invalid. Clearly we can reduce some clock cycles by culling invalid polygons as soon as we read them from the sample_test.dat file. Thus we made changes to the rast_driver file. Here in the while loop we first scan only the isvalidPoly variable. If its valid we scan the rest of the line. If its not valid, we scan the line into fake variables which are not passed to the computation pipeline. Then the next line is scanned and the same process is repeated till isvalidPoly is found to be 1.

Improvement:

Cycles: 642 uPoly: 110 sampleTests: 597 sampleHits: 135

Relevant file:

rast_driver_invalidcull.v

- Synthesis Improvements

1) Modified Booth Encoding

The critical path bottle neck in multiplier was resolved by our MODIFIED BOOTH ALGORITHM (using 3 bits) .Combining three bits at a time for Partial Product Generation and using TREE structure in Partial Product Multiplication, the RTL code resides in "modifbooth.v"

2) Comparator

Modified as a Tree structure from Daisy chain and also built a BITWISE COMPARATOR based on MORRIS MANO Digital Design Book Logic, with the inverted logic for first signed bit. The corresponding COMPARATOR module resides in "cmpkar.v"; though it has lot of wires, but effectively it is Order(one) time calculation.

VERIFICATION

We picked a sample test file consisting of 145 polygons which was included in the baseline directory. All our modification were tested against this file. Later we tested against the bigger databases.
The basic idea behind verifying was to give a test vector which contained only a small number of test cases which we knew should produce a change in output variables like samplehits, sampletests,cycles etc.
1. Backface culling:
In this we modifed the bbox.v file and then modified sample_test.file to hold only one backfacing triangle. Then we observed the output to see if the variable SampleTests went down. Similarly for a quadrilateral. We also inputted cases of backfacing triangles to compare the cycles consumed to know where it is invalidated. One case is
1 3 00020 00020 00050 00020 00030 00030
2. Ceiling:
In this we modified bbox.v and observed the sampletests to go down. Simultaneously the samplehits should not go down. Further we checked for cases of point already coinciding at the sample space and infinite loops.

3. FSM:
Here the number of cycles should go down by the number of polygons processed and the samplehits should remain same.
Initially, CYCLE=uPoly+SampleTests
Now CYCLE=SampleTests + 3(to fill the pipeline initially)

4.Invalid polygon culling:
This code should reduce the number of cycles by the number of invalid polygons but there should be no change to either sample tests or samplehits.
Can be seen only in the small dataset, as other have valid signal ==1 always.
By using at bbox.v just after signal declaration:
$monitor("Inputstream %b %b %6h %6h %6h %6h %6h %6h %6h %6h", validPoly_R10H, isQuad_R10H, poly_R10S[0][0], poly_R10S[0][1], poly_R10S[1][0], poly_R10S[1][1], poly_R10S[2][0], poly_R10S[2][1], poly_R10S[3][0], poly_R10S[3][1]);

5.Bubble smashing:
This code reduces the number of cycles keeping samplehits and sampletests. After scanning only valid polygon or for bigger test cases, if you comment back facing then this code does not produce any change. Again using abpve mentioned $display command in always@(posedge clk) we can know the polygon inputs at each cycle and hence detect whether our halt is currently overpassed in case of bubble or not.

6.Triangle merging:
Running it on the bigger samples, and using same display command, by inspecting the isQuad_R10H , we can detect the conversion cases.
Also, the number of uPoly will be reduced by a factor of half for only triangle cases. For debugging we printed the vertices to check if order of front facing or back facing have been done.

DISCREPANCY

We have commented a few lines in gold folder to avoid errors arising due to differences from Back face culling that invalidates the signal earlier and Ceiling which reduces the number if sample points.
Furthermore, as observed in the RESULTS section, there exists a minor difference in the number of sample Hits in original and our model. We first encountered it during FSM modification. For the specific test case:

80000 80000 4
1 4 0000a2 fffe08 fffdd5 0001b5 fffe5a 000232 000625 fffff5
1 4 fffe0c fffd15 000895 0000fc 00043c 000116 00056f 0003ec

We then printed the sample point, validSample signal and hit result.

Original Model:

```
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000000000000 0 0
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000000000000 0 0
Input Sample Points ValidSignal HitSignal 00000000000000010000000 00000000000000000000000 1 1
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000000000000 1 1
Input Sample Points ValidSignal HitSignal 00000000000000110000000 00000000000000000000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000010000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000110000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000000000000 1 1
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000000000000 0 0
Input Sample Points ValidSignal HitSignal 00000000000000010000000 00000000000000000000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000000000000 1 1
Input Sample Points ValidSignal HitSignal 00000000000000110000000 00000000000000000000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000000000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000010000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000010000000 1 1
Input Sample Points ValidSignal HitSignal 00000000000000110000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000000000000 0 0
time=          45000          **************          Driver          Is          Done          ******************
time=                                45000                                **********************FINISH************************
Cycles:      31 uPoly:      3 sampleTests:      18 sampleHits:      4
```

Our Model:

```
Input Sample Points ValidSignal HitSignal 00000000000000010000000 00000000000000000000000 0 0
Input Sample Points ValidSignal HitSignal 00000000000000010000000 00000000000000000000000 1 1
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000000000000 1 1
Input Sample Points ValidSignal HitSignal 00000000000000110000000 00000000000000000000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000010000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000110000000 00000000000000010000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000000000000 1 1
Input Sample Points ValidSignal HitSignal 00000000000000010000000 00000000000000000000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000000000000 1 1
Input Sample Points ValidSignal HitSignal 00000000000000110000000 00000000000000000000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000100000000 00000000000000000000000 1 0
Input Sample Points ValidSignal HitSignal 00000000000000000000000 00000000000000010000000 1 0
```

Input	Sample	Points	ValidSignal	HitSignal	000000000000001000000000	000000000000001000000000	1	0
Input	Sample	Points	ValidSignal	HitSignal	000000000000010000000000	000000000000001000000000	1	1
Input	Sample	Points	ValidSignal	HitSignal	000000000000011000000000	000000000000001000000000	1	0
Input	Sample	Points	ValidSignal	HitSignal	000000000000100000000000	000000000000001000000000	1	0
Input	Sample	Points	ValidSignal	HitSignal	000000000000000000000000	000000000000000000000000	1	0
Input	Sample	Points	ValidSignal	HitSignal	000000000000001000000000	000000000000000000000000	0	0

time= 43000 ************** Driver Is Done *****************
time= 43000 ********************FINISH***********************
Cycles: 29 uPoly: 3 sampleTests: 18 sampleHits: 5

On counting the number of hits in original case it comes to be 5 and the counter wrongly reports it to be 4.

Hence, most of the cases error occurs due to clash with the counter. But, that is minor as compared to lakhs of sampleHits.

RESULTS

The following is a summary of improvements achieved in a step by step manner as we implemented the different design ideas. All tests were done on the 145 poly file.

Change done	Cycles	Poly	SampleTests	ThroughPut(poly/cycle)
Nothing	1025	145	870	0.14146
Backface culling and ceiling	753	145	597	0.19256
Backface culling,ceiling and idle state in FSM	679	145	597	0.21355
BackFace, ceiling , FSM, cull invalid polys	642	145	597	0.22586
With all the improvements (overall 6) included	598	145	583	0.2425

Following is the synthesis report for some of the design improvements.

Changes done	Power	Area(mm2)	Timing	Throughput(poly/sec)	Throughput(poly/cycle)
Nothing	12.44 mW	0.034	1.05	134 Million /sec	0.14146
Multiplication using Modified Booth + bitwise comparator	12.156 mW	0.019	0.75	---	-----
Backface culling , ceiling and culling idle state in FSM	13.6 mW	0.021	0.75	284 Million/sec	0.21355
Only triangle Merge	13.595 2 mW	0.023	0.75	293 Million/sec	0.21975
Everything	15.089 4 mW	0.0275	0.8	303 Million/sec	0.2425

THROUGHPUT RESULTS:

Here we have compared our design throughput with the baseline design based on a 200000 poly data which has been attached as sample_test.dat in the verif/tests folder. We have achieved an improvement of 300 % in throughput.

DESIGN	Numberofpoly/MSAA	Cycles	Sampletests	SampleHits	Throughput(Million poly/sec)
Original	200000/4	2189483	1989480	252364	79
Our Design	200000/4	1140326	1111137	252261	219.24
Original	200000/16	5824522	5624519	1009511	32.7
Our Design	200000/16	3240470	3218579	1009299	61.7

FINAL RESULTS:

Per Instance:

Clock 0.8nS

Design Power is 15.0729 mW

Design Area is 0.027 mm^2

Design Throughput is 0.2425 uPoly/cycle or 303 Million uPoly/Sec (on a data of 145 polygons)

Design Throughput is 0.175 uPoly/cycle or 219.24 Million uPoly/Sec (on a data of 200000 polygons)

Total:

Instantiating 3 instances will achieve the instance (while the original design needed 8 instances)

Clock 0.8ns

Power 45.2187 mW

Design Area is 0.081 mm^2

Design Throughput is 0.7275 uPoly/cycle or 909 Million uPoly/sec (on a data of 145 polygons)

Design Throughput is 0.525 uPoly/cycle or 657.72 Million uPoly/Sec (on a data of 200000 polygons)

REFERENCES

- cplusplus.com : functions regarding file pointers
- Digital Design, Morris Mano : bitwise comparator
- Computer Organization and Design Hardware and Software Interface, Henessey and Patterson: pipeline logic.