Fall 2021

California State University, Northridge

Department of Electrical and Computer Engineering

Project - Mandelbrot Zoom

November 17, 2021

ECE 551/ECE 524

Professor: Shahnam Mirzaei

Written By: Morris Blaustein

Table of Contents

Introduction

The Mandelbrot set is the set of complex numbers for which the function $f\_c\ (z)$=z^2+c does not converge. A point c belongs in the Mandelbrot set if and only if $|z\_n\ |$ ≤ 2 for all n ≥ 0. The escape time algorithm was used to determine the n at which the magnitude of z_n exceeded 2. The mandelbrot set was visualized in this project by assignining the color intensity of a pixel to the maximum number of iterations in which it did not diverge. If a point did not diverge after n=64, then it was assigned the highest pixel intensity. The following examples show how to determine if a complex number is contained in the mandelbrot set.

Example (c = -1 + 0i)
f_c (0)= 0^2−1
f_c (−1)= (−1)^2−1
f_c (0)=0^2−1
f_c (−1)= 〖(−1)〗 ^2−1
|z_n| does not exceed 2, thus -1 + 0i is within the Mandelbrot set.

Example (c = 1 + 0i)
f_c (0)=0^2+1
f_c (1)=1^2+1
f_c (2)=2^2+1
f_c (5)=5^2+1|z_n| exceeds 2 at n = 4, thus 1 + 0i is not within the Mandelbrot set.

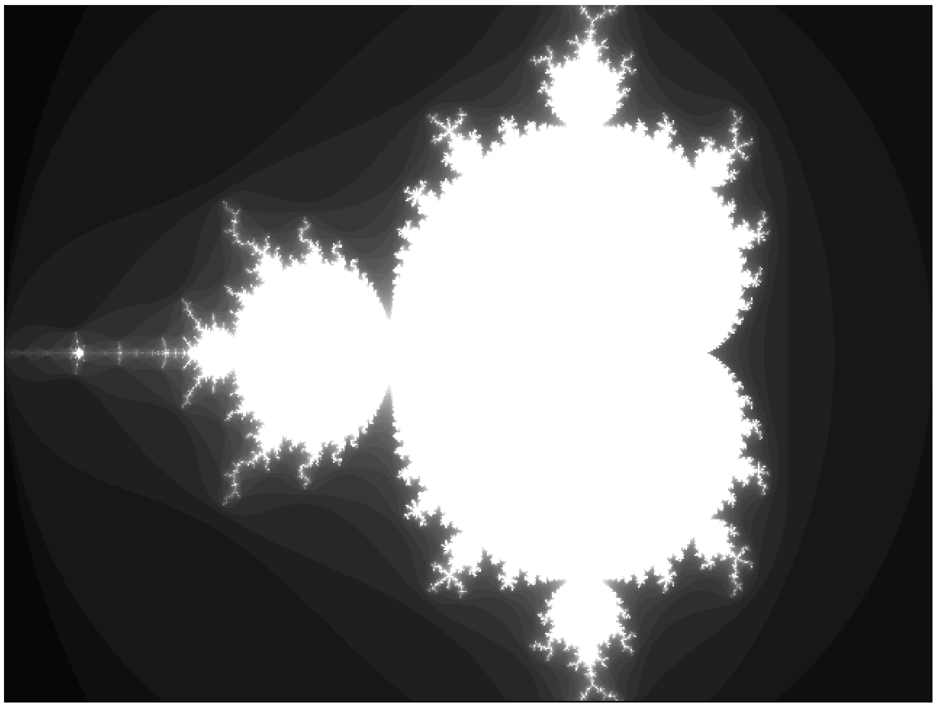


Figure 1 - Simulation results, max iterations = 32

On the FPGA, complex numbers were converted to real numbers. The following equations show a complex number z and the real and imaginary components of z^2.

$z=x+iy$

$z^2=x^2+2ixy-y^2$

$x=Re(z)=x^2-y^2$

$y=Im(z)=2xy$

$x^2+y^2\le 4$

The escape time algorithm determined the iteration at which a point diverged from the Mandelbrot set:

```
while ( $x^2+y^2 \le 4$ and iteration < max_iteration )
  $y=2xy+y_0$
  $x=x^2-y^2+x_o$
  iteration += 1
```

Implementation

The video zoomed in on a point on the edge of the Mandelbrot set. The geometry shown in Figure 2 shows how the zooming parameters were calculated.

x_start, y_start: point at the top left of the current frame

x_inc, y_inc: distance between pixels in horizontal and vertical directions

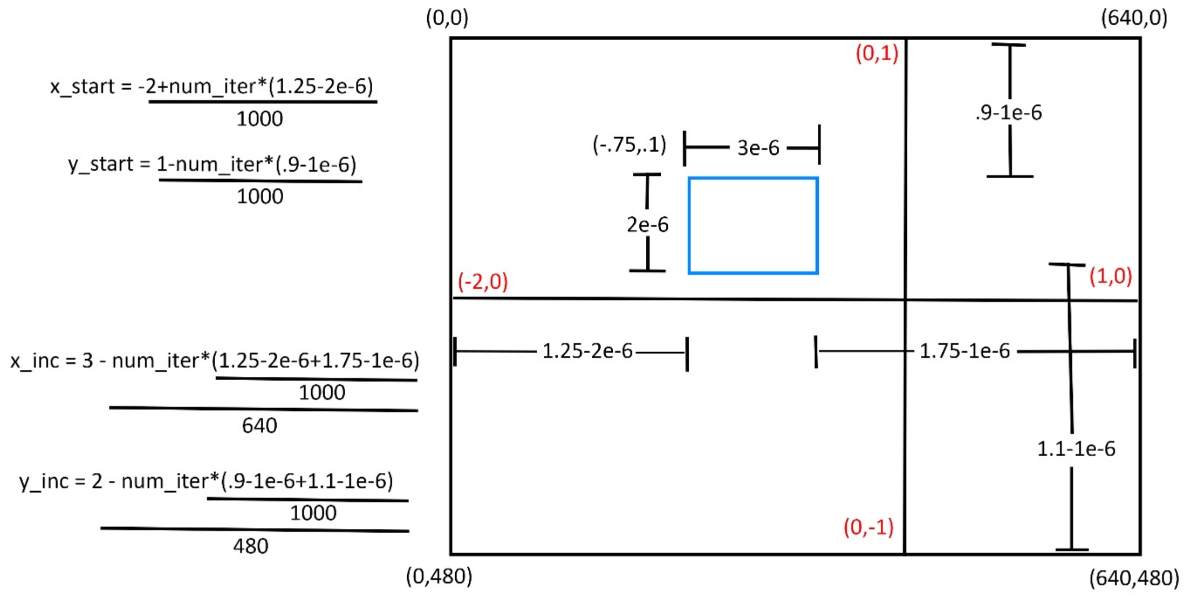

x_start = -2+num_iter*(1.25-2e-6) / 1000
y_start = 1-num_iter*(.9-1e-6) / 1000
x_inc = 3 - num_iter*(1.25-2e-6+1.75-1e-6) / 1000 / 640
y_inc = 2 - num_iter*(.9-1e-6+1.1-1e-6) / 1000 / 480
(0,0)
(640,0)
(0,1)
.9-1e-6
(-.75,.1)
3e-6
2e-6
(-2,0)
(1,0)
1.25-2e-6
1.75-1e-6
1.1-1e-6
(0,-1)
(0,480)
(640,480)

Figure 2 - Geometry of zooming algorithm and equations to determine Zooming Parameters

The pixel locations were calculated with the VHDL fixed point library. The following is an example of how fractional numbers were used to calculate pixel locations:

```
signal a, b, c : sfixed(10 downto -37);
…
```

a <= resize(To_sfixed(1,2,-4)/To_sfixed(2,2,-4),a); …1/2
b <= resize(To_sfixed(1,2,-4)/To_sfixed(4,2,-4),a); …1/4
c <= resize(a+b,c) … 3/4
Binary equivalent of 3/4: 000.1100

The state diagram in Figure 3 shows the state diagram that was used. A pixel location was mapped to a pixel on the VGA screen. Then, it was assigned a color from the result of the escape time algorithm. When all pixel colors were assigned, a delay occurred to increase the time each zoom iteration was displayed.

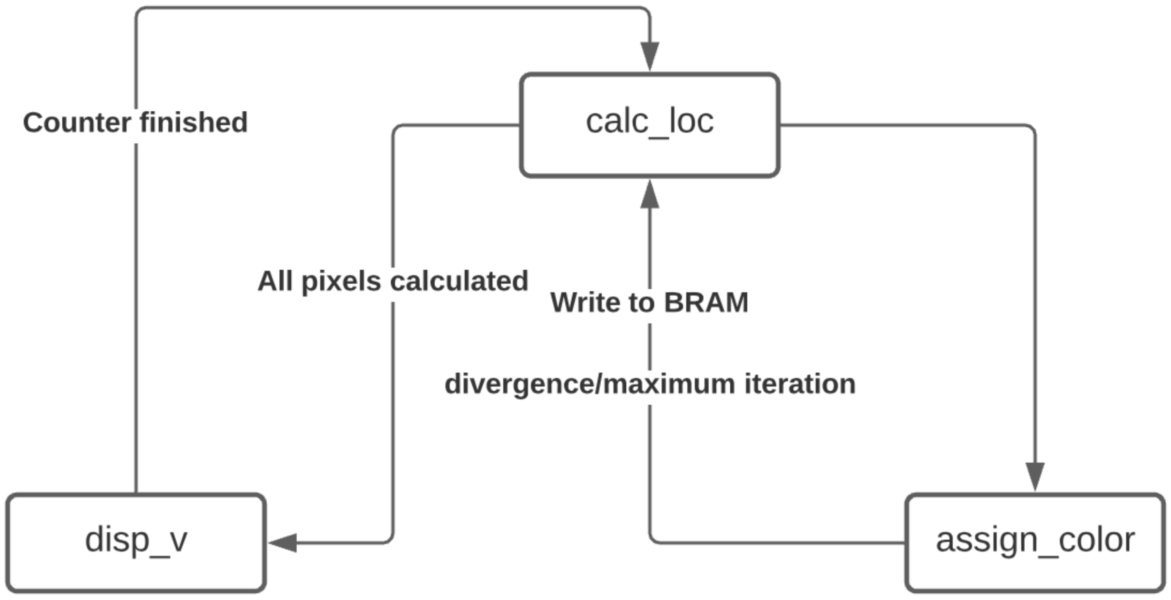


Figure 3 – State diagram of Mandelbrot zoom

A simple dual port block RAM was used to store the color assignments of each pixel so they could be continuously displayed to the screen. The port width was 6 bits, and the port depth was 307,200 (480*640). The 6 bit width allowed for 64 iterations of the escape time algorithm to be used. The numbers 0-63 were mapped to 0-255 with the following transformation:

Iter2col <= std_logic_vector(to_unsigned(to_integer(unsigned(doubt))*4-1, iter2col’length));
Color Intensity = iterations∗4−1 , (0−255)

An HDMI-out library was used to output VGA on the HDMI TX of the Zybo-Z7. The library generated timing signals and converted video data to TDMS (transition-minimized differential signaling) which transmitted the serial data.

Simulation and Testing

The contents of the BRAM were written to a file in Vivado simulation. The data was imported into MATLAB and reconstructed as an image with the same dimensions as VGA, 640x480:

```
for I = 1:480
    for j = 1:640
        img(i,j) = data(640*(i-1)+j);
    end
end
```

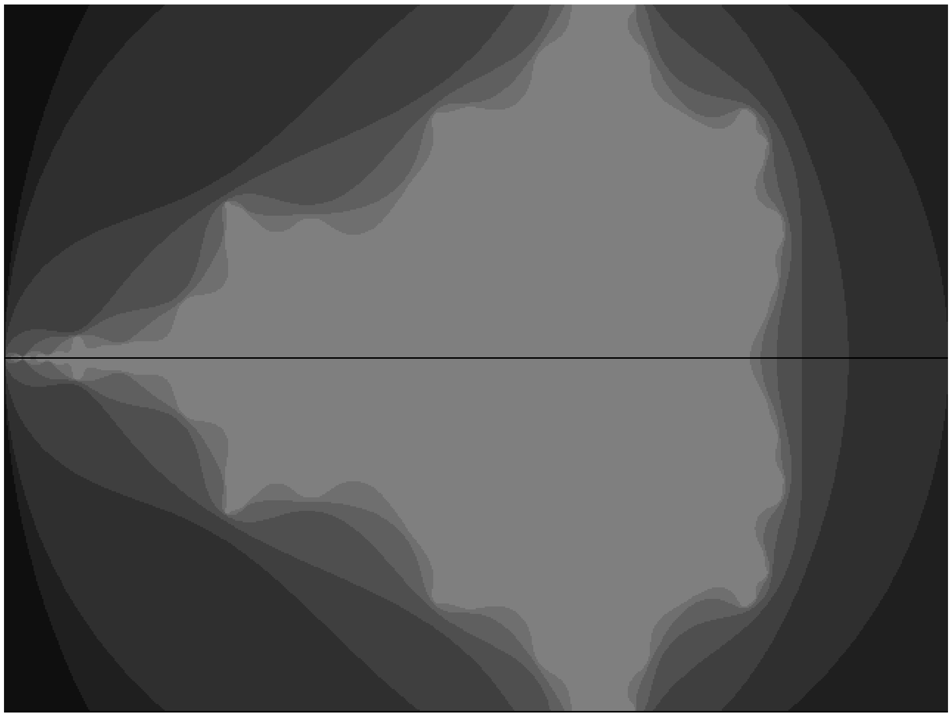


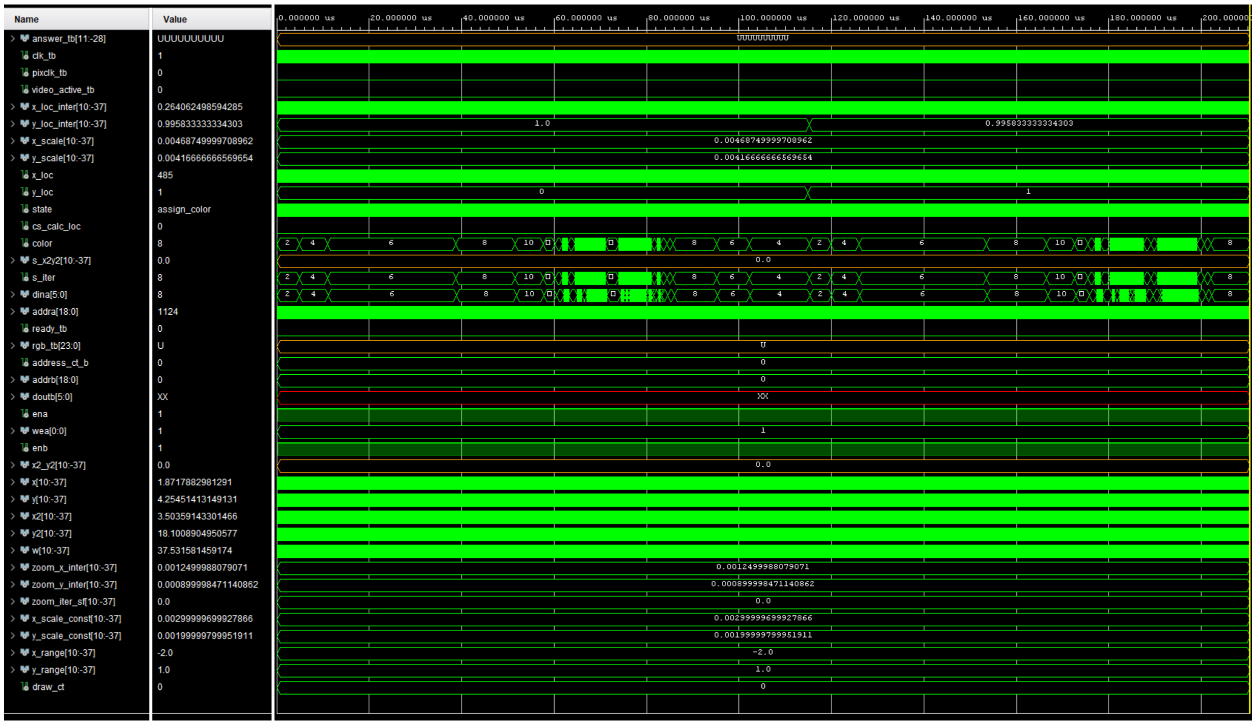
Figure 4 - Simulation results, max iterations = 8

The output of the simulation was compared to the expected results from the following MATLAB code:

```
xpixel=100;
ypixel=100;
num_iter = 64;
mandelbrot_check(-2+.0046879999708962*xpixel+1i*(1-.00416666666569654*ypixel),num_iter)
function [counter] = mandelbrot_check(c,maxIter)
z=0;
counter = 0;
while abs(z) < 2 && counter < maxIter
   z = z^2 + c;
```

```
    counter = counter + 1;
end

if counter == maxIter
    inSet = true;
else
    inSet = false;
end
end
```



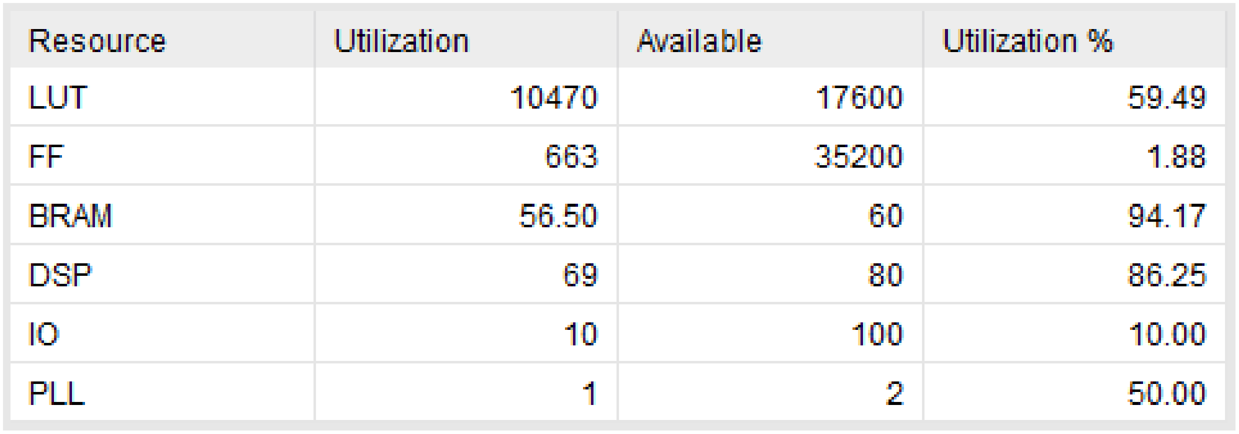
Name	Value
answer_tb[11:-28]	UUUUUUUUUU
clk_tb	1
pixclk_tb	0
video_active_tb	0
x_loc_inter[10:-37]	0.264062498594285
y_loc_inter[10:-37]	0.995833333334303
x_scale[10:-37]	0.00468749999708962
y_scale[10:-37]	0.00416666666569654
x_loc	485
y_loc	1
state	assign_color
cs_calc_loc	0
color	8
s_x2y2[10:-37]	0.0
s_iter	8
dina[5:0]	8
addra[18:0]	1124
ready_tb	0
rgb_tb[23:0]	U
address_ct_b	0
addrb[18:0]	0
doutb[5:0]	XX
ena	1
wea[0:0]	1
enb	1
x2_y2[10:-37]	0.0
x[10:-37]	1.8717882981291
y[10:-37]	4.25451413149131
x2[10:-37]	3.50359143301466
y2[10:-37]	18.1008904950577
w[10:-37]	37.531581459174
zoom_x_inter[10:-37]	0.0012499988079071
zoom_y_inter[10:-37]	0.000899998471140862
zoom_iter_sf[10:-37]	0.0
x_scale_const[10:-37]	0.00299999699927866
y_scale_const[10:-37]	0.00199999799951911
x_range[10:-37]	-2.0
y_range[10:-37]	1.0
draw_ct	0

Figure 4 – Vivado Simulation Configuration

Results

https://youtu.be/Am59BlNa_mY

FPGA Utilization



Resource	Utilization	Available	Utilization %
LUT	10470	17600	59.49
FF	663	35200	1.88
BRAM	56.50	60	94.17
DSP	69	80	86.25
IO	10	100	10.00
PLL	1	2	50.00

Table 1 - FPGA Resource Utilization

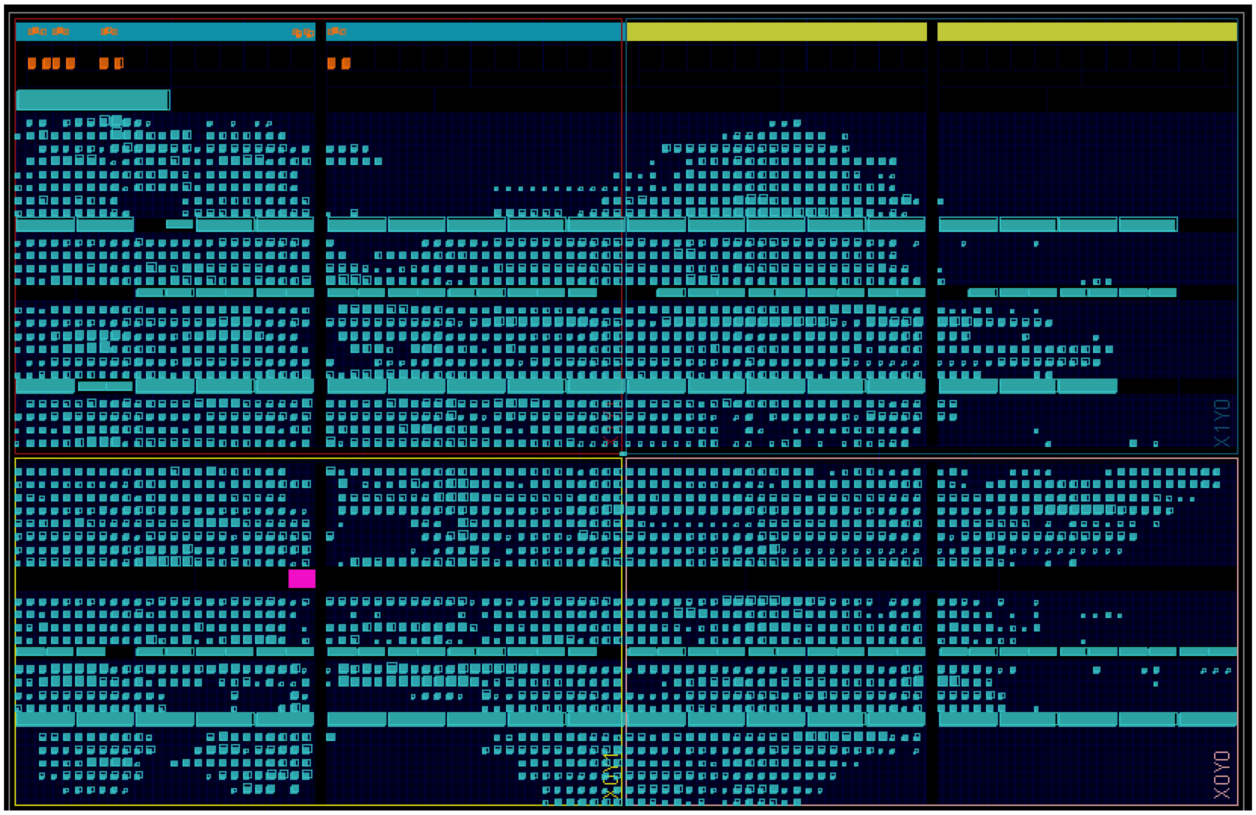


Figure 5 - FPGA Device Utilization

Conclusion

This project helped me become more comfortable working with VHDL and Vivado. I practiced the syntax of VHDL so I can write complete code without using external resources. The main challenges of this project were:

- Understanding the Mandelbrot set
- Using the VHDL fixed library
- Meeting resource requirements for the Zybo Z7
- Creating a plan for the zooming algorithm
- Understanding the HDMI out library
- Waiting for slow simulation, synthesis and implementation

A known bug in this program is that the screen is slightly misaligned on the left side. This is probably because the BRAM memory address counter is slightly misaligned from the VGA sync signals.

Future improvements of this project could be:

- Parameterize the center of the zoom
- Pipeline calculations to increase speed and fluidity of zoom
- Joystick to manually control zooming area
- Increase number of iterations, currently 64
- Increase the amount of zoom, currently 1,000,000x magnification

If a device with a larger BRAM was used, the Mandelbrot set could be plotted with more detail because the number of iterations could increase. The program would be able to zoom in further.

There were a lot of steps to completing this project including initial research and planning, testing, and synthesizing. I initially hoped that the video could zoom in further than 1,000,000x, but the constraints of memory did not allow it. It was rewarding to get this project working and I look forward to continuing to combine programming and art in the future.