CSCE 313: Embedded Systems

Scaling Multiprocessors

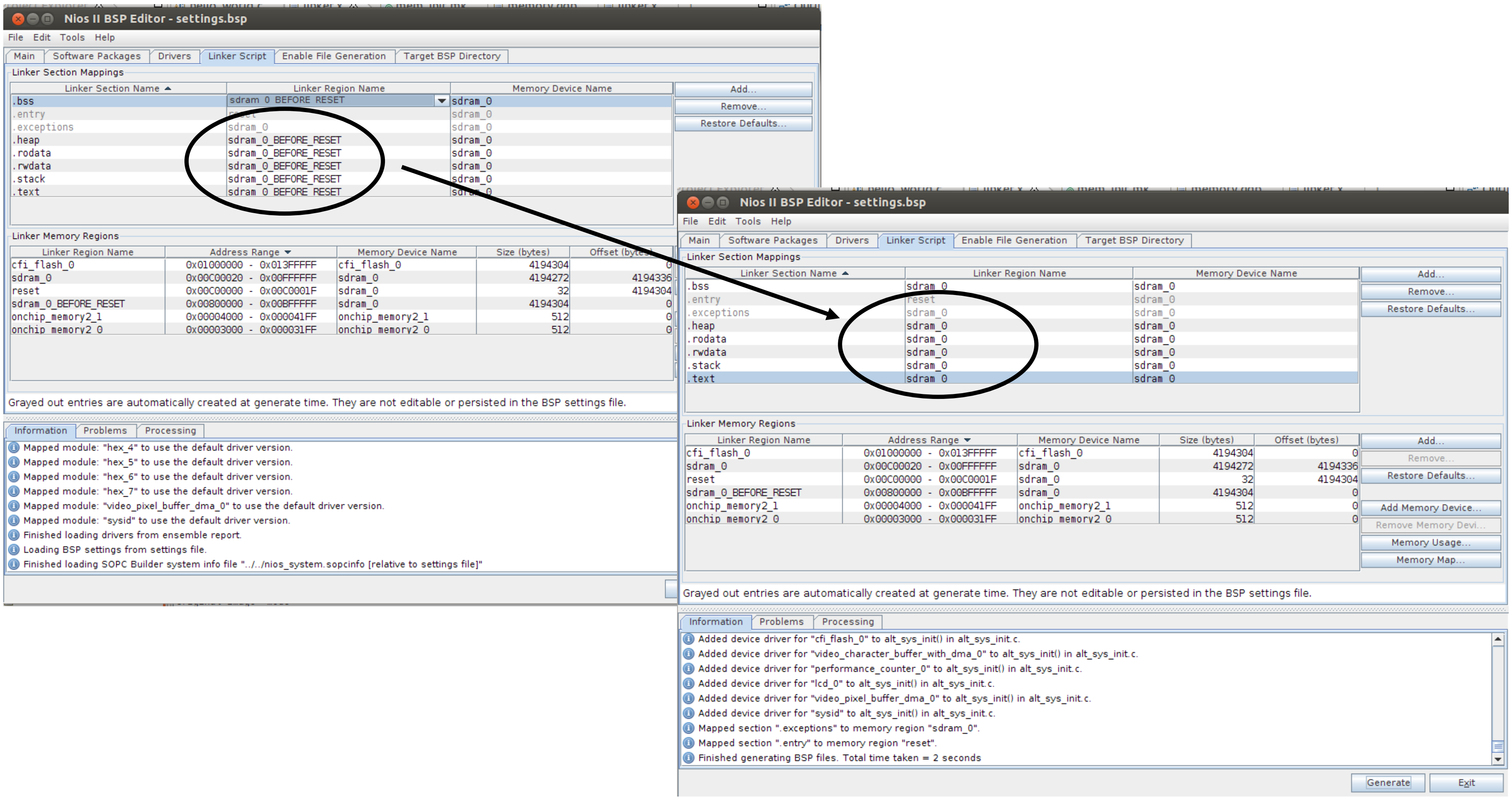
Instructor: Jason D. Bakos

UNIVERSITY OF SOUTH CAROLINA

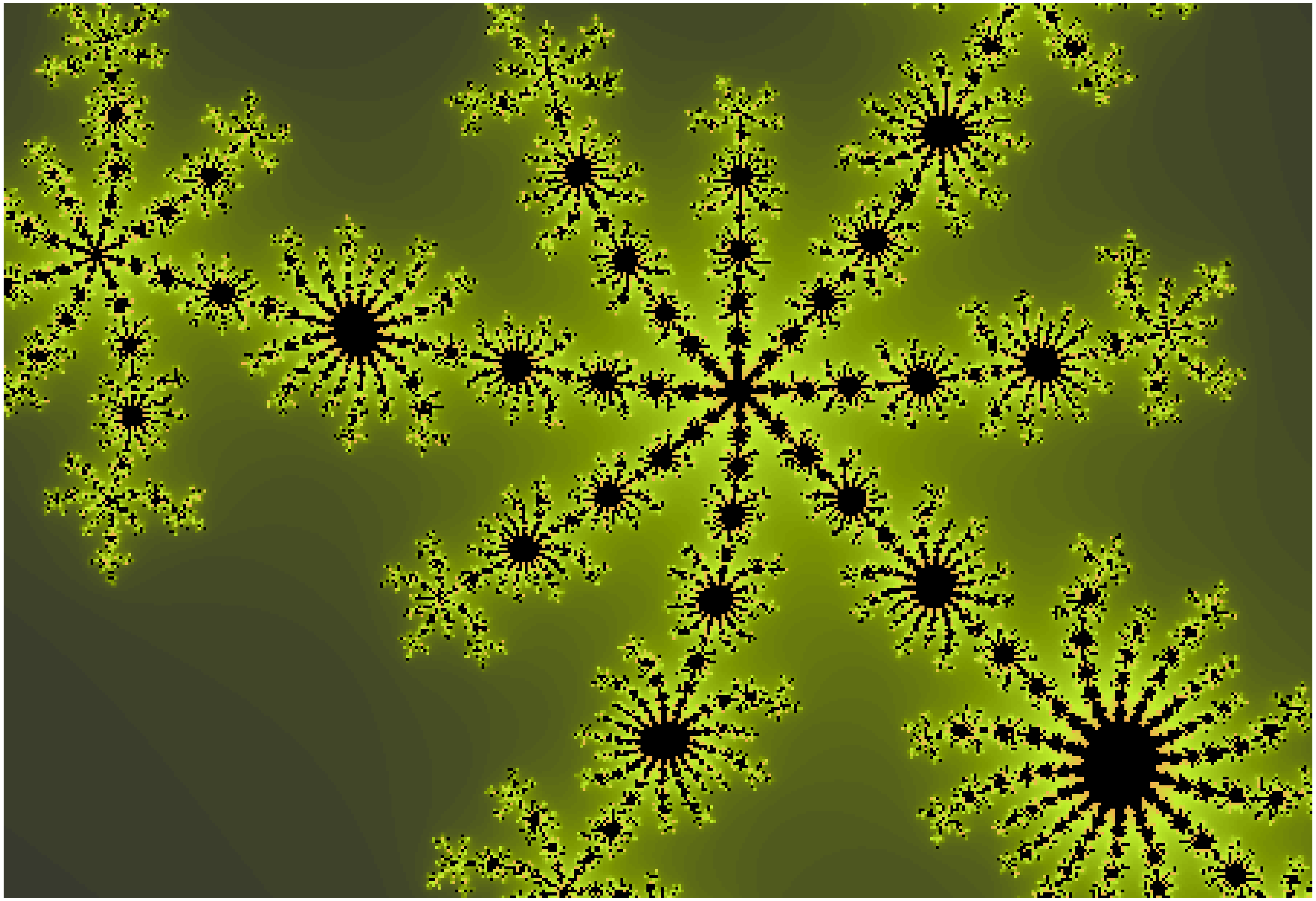
Barrier Code

```
void barrier () {
    volatile alt_u32 *counters = ONCHIP_MEMORY2_0_BASE;
    int i,count;
    alt_mutex_dev *my_mutex;
    alt_u32 cpuid;

    cpuid = __builtin_rdctl(5);

    my_mutex =
altera_avalon_mutex_open("/dev/mutex_0");
    if (!my_mutex) {
        perror("error opening mutex");
        return 0;
    }

    for (i=0;i<NUM_CPUS;i++) {
        if (i!=cpuid) {
                altera_avalon_mutex_lock(my_mutex, 1);
                count = IORD_32DIRECT(&counters[i],0);
                count++;
                IOWR_32DIRECT(&counters[i],0,count);
                altera_avalon_mutex_unlock(my_mutex);
        }
    }

    do {
        count = IORD_32DIRECT(&counters[cpuid],0);
    } while (count != NUM_CPUS-1);

    altera_avalon_mutex_lock(my_mutex, 1);
    IOWR_32DIRECT(&counters[cpuid],0,0);
    altera_avalon_mutex_unlock(my_mutex);

    altera_avalon_mutex_close(my_mutex);
}
```

BSP Editor Linker Script

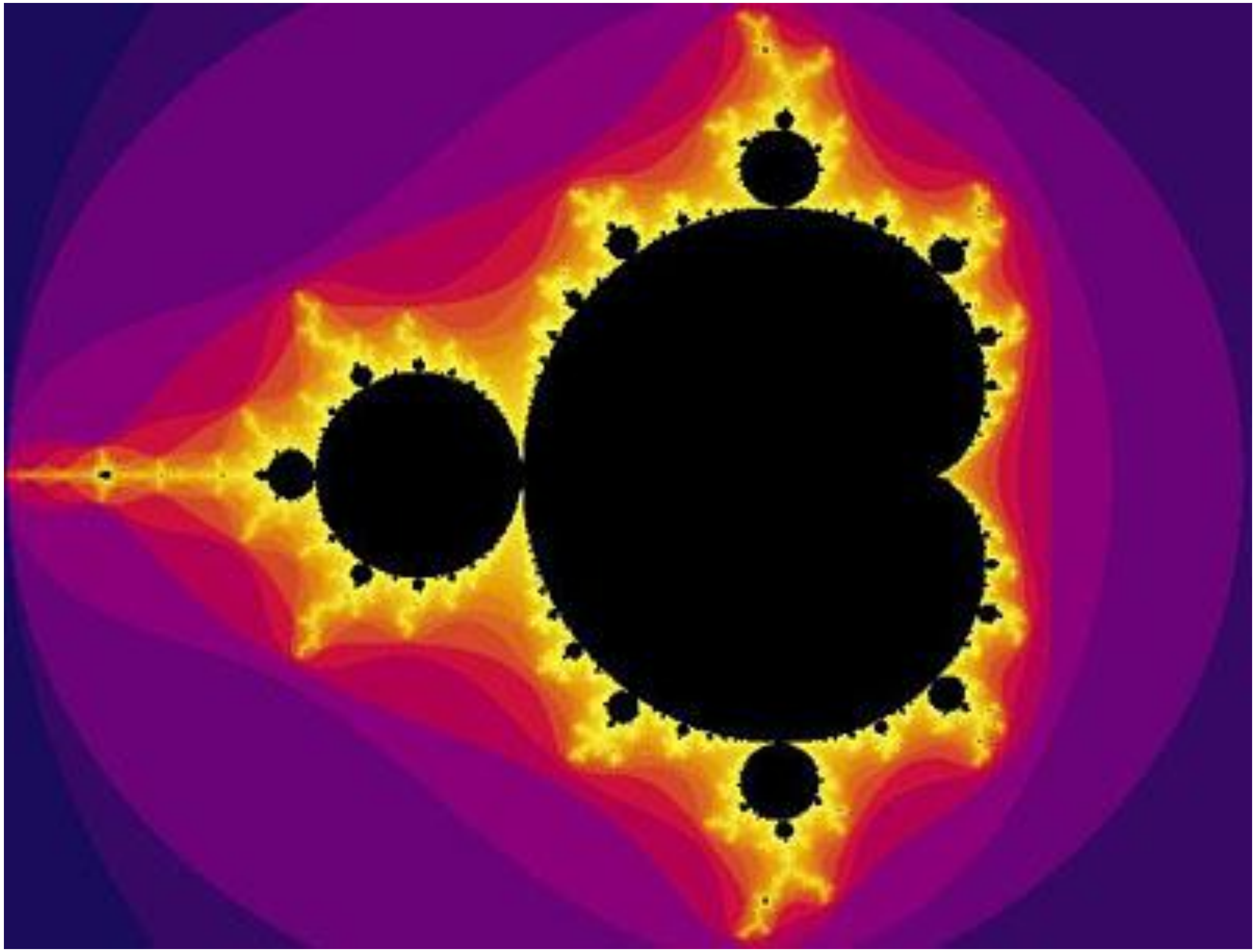


Fractals



Fractals

Mandelbrot Set



Mandelbrot Set

- Basic idea:
  - Any arbitrary complex number c is either in the set or not
    - Recall complex numbers have a real and imaginary part
    - e.g. 3 + 2i
    - i = sqrt(-1)
  - Plot all c's in the set, set x=Real(c), y=Imag(c)
    - Black represents points in the set
    - Colored points according to how "close" that point was to being in set

Mandelbrot Set

- Definition:
  - Consider complex polynomial $P_c(z) = z^2 + c$
  - c is in the set if the sequence:
    $P_c(0)$, $P_c(P_c(0))$, $P_c(P_c(P_c(0)))$, $P_c(P_c(P_c(P_c(0))))$, …
  - …does NOT diverge to infinity
  - Guarantee: all points in the set are contained radius = 2 around (0,0)

Mandelbrot Set

- How do you square a complex number?
  - Answer: treat it as polynomial, but keep in mind that $i^2 = -1$
  - Example:
    - $(3 + 2i)^2$
      $= (3 + 2i)(3 + 2i)$
      $= 9 + 6i + 6i + 4i^2$
      $= 9 + 12i - 4$
      $= 5 + 12i$

    - $(x + yi)^2$
      $= (x + yi)(x + yi)$
      $= x^2 + 2xyi - y^2$
      $= (x^2 - y^2) + 2xyi$

Example 1

- Is (.5 + .75i) in the Mandelbrot set?
  - $P_{(.5\ +\ .75i)}(0) = 0^2+(.5 + .75i) = .5 + .75i$
  - $P_{(.5\ +\ .75i)}(P_{(.5\ +\ .75i)}(0)) = (.5 + .75i)^2 + (.5 + .75i) = 0.1875 + 1.5i$
  - $P_{(.5\ +\ .75i)}(P_{(.5\ +\ .75i)}(P_{(.5\ +\ .75i)}(0))) = (0.1875 + 1.5i)^2 + (.5 + .75i)$
    $= -1.7148 + 1.3125i$ (outside)
  - … $= 1.7179 - 3.7514i$ (outside)
  - … $= -10.6218 -12.1391i$ (outside)

  Color should reflect 4 iterations

Example 2

- Is (.25 + .5i) in the Mandelbrot set?

  - Iteration 1 => 0.2500 + 0.5000i
  - Iteration 2 => 0.0625 + 0.7500i
  - Iteration 3 => -0.3086 + 0.5938i
  - Iteration 4 => -0.0073 + 0.1335i
  - Iteration 5 => 0.2322 + 0.4980i
  - Iteration 6 => 0.0559 + 0.7313i
  - Iteration 7 => -0.2817 + 0.5817i
  - Iteration 8 => -0.0090 + 0.1723i
  - …
  - Iteration 1000 => -0.0095 + 0.3988

  - (appears to be in the set)

Mandelbrot Set

- Goal of next lab:
  - Use the DE2 board to plot a Mandelbrot fractal over VGA and zoom in as far as possible to “reveal” infinitely repeating structures
  - Problems to solve:
    1. How to descretize a complex space onto a 320x240 discrete pixel display
    2. How to determine if a discretized point (pixel) is in the set
    3. How to zoom in
    4. What happens numerically as we zoom in?
    5. How to parallelize the algorithm for multiple processors

Plotting the Space

- Keep track of a "zoom" window in complex space using the four floating point variables:
  - min_x, max_x, min_y, max_y
- Keep a flattened 240x320 pixel array, as before
  - Each pixel [col,row] can be mapped to a point in complex space [x,y] using:
    - x = col / 320 x (max_x – min_x) + min_x
    - y = (239-row) / 240 x (max_y – min_y) + min_y
- Keep track of your zoom origin in complex space, or target point:
  - target_x, target_y

Algorithm

- For each pixel (row,col):
  - Check to see if the corresponding point in complex space is in Modelbrot set:

```
Transform (row,col) into c = (x0,y0)
initialize z = 0 => x=0, y=0  // recall that series begins with Pc(0)
set iteration = 0
while ((x*x + y*y) <= 4) and (iteration < 500)
// while (x,y) is inside radius=2 (otherwise we know the series has diverged)
        – xtemp = x*x – y*y + x0
        – y = 2*x*y + y0
        – x = xtemp
        – iteration++
if iteration == 500 then
   color=black,
else
   color=(some function of iteration)
```

Zooming

- Goal:
  - We want to zoom in to show the details on the fractal
  - Problem: on which point to we zoom?
  - Set the initial frame to encompass:
    - -2.5 <= x <= 1
    - -1 <= y <= 1
    - (This is the typical window from which the Mandelbrot fractal is shown)
  - During the first frame rendering, find the first pixel that has greater than 450 iterations
    - Set this only once!
    - How to set when there's >1 processors?
  - This will identify a colorful and featureful area
  - Set this point (in complex space) as your target_x and target_y

Zooming

- To zoom in:
  - min_x=target_x – 1/(1.5^zoom)
  - max_x=target_x + 1/(1.5^zoom)
  - min_y=target_y - .75/(1.5^zoom)
  - max_y=target_y + .75/(1.5^zoom)
- In the outer loop, increment zoom from 1 to 100 (or more)

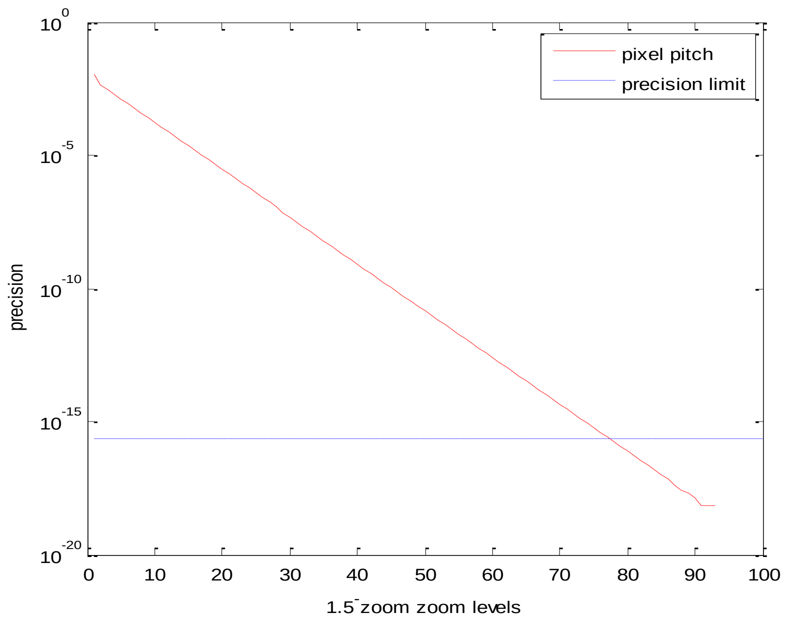
- Fractals are deliberately made colorful, but the way you set the colors is arbitrary
  - Here's one sample technique:
  - color [R,G,B] =
    - [iteration*8/zoom, iteration*4/zoom, iteration*2/zoom]
  - This creates a yellowish brown hue that dampens as you zoom in
  - Make sure you saturate the colors

Numerical Precision

- You'll notice as you zoom in that picture definition quickly degrades

- This is because double precision values have a precision of 2^{-52} and zooming in at a quadratic rate reaches this quickly
  - In other words, the difference in the complex space between pixels approaches this value
  - Note: $2^{-52} = 2.2 \times 10^{-16}$

- Inter-pixel distance from 0 to 200 iterations using specified zoom



Precision

- Interesting note:
  - Size of observable universe
    - 93 billion light years
    - $= 8.8 \times 10^{26}$ meters
  - Smallest constant in physics, Plank’s constant:
    - 6.0×10^{-34} m^2kg/s
  - Ratio is 6.8×10^{-61}
  - At the rate we’re zooming, we achieve that ratio in ~341 iterations

Parallelizing

- The frame rate will depend on how many of the pixels in the frame are in the Mandelbrot set, since these pixels are expensive (requires 1000 loop iterations each)
  - Lighter-colored pixels are also expensive, though less so

- To speed things up, use multiple processors

- Use the data parallel approach and write to the frame buffer in line

Notes

- Make sure you add hardware floating point on each processor
- Use at least 4KB instruction and data cache per processor
- Implement on one processor first