Algorithms to Architectures

In this exercise, we transform an algorithm into an architecture, focusing on the design of the datapath. We explore different datapath options through architectural transformations and analyze them in terms of timing and complexity.

Hand-in instructions: Prepare a small report with your solutions (detailed). Submit your report as a PDF through the lecture moodle per the moodle submission deadline.

Algorithm Description

We consider the iterative algorithm for computing the Mandelbrot set which yields beautiful pictures like the one in Figure 1. A complex number c is a member of the Mandelbrot set if the recurrence $z_{n+1} = z_n^2 + c$, with $z_0 = c$, remains bounded as $n \to \infty$. As an example, $c = -1$ is a member of the Mandelbrot set as the recurrence only returns 0 and -1, i.e., starting from $z_0 = -1$ we get $z_1 = (-1)^2 - 1 = 0, z_2 = -1, z_3 = 0$ and so on. As c is a complex number, we can use the real- and imaginary-parts as image coordinates, and depending on how fast the recursion crosses an arbitrary threshold we can color the pixel at this coordinate.

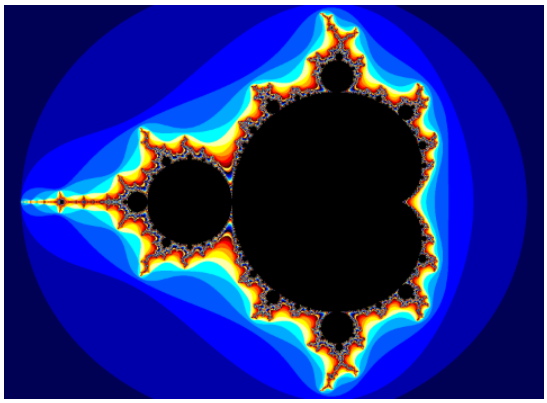

Figure 1: The Mandelbrot set shown in black

In practice, these pictures are obtained by computing (and testing the outcome of) the following:

```
INPUTS: c_r, c_i, MAX_ITER;
OUTPUT: z_r, z_i;

n=0; z_r=c_r; z_i=c_i;
while((z_r*z_r + z_i*z_i)<2*2 && n<MAX_ITER) {
  z_r' = z_r*z_r - z_i*z_i + c_r;
  z_i  = 2*z_r*z_i + c_i;
  z_r  = z_r';
  n++;
}
```

In this case, 2^2 has been chosen as our arbitrary threshold. The counter n determines the number of iterations for testing the divergence of c, that is, finding when $|z_n|^2 \geq 2^2$. For

practical reasons, we have to put a limit on the number of iterations, which in this case is MAX_ITER. The pictures are obtained by plotting pixels based on n. If n reaches the maximum iteration number we color the pixel black and otherwise we use the lower bits for blue, middle bits for red and higher bits for green, which is why the colors are brighter closer to the mandelbrot set and dark blue further away from it.

Focus of this Exercise

For this exercise, we focus only on the datapath and ignore the termination condition. Instead, we simply compute z_r and z_i for a fixed (given) number of iterations MAX_ITER. We suppose that the number of iterations MAX_ITER is fixed at design time. Therefore, we consider the following slightly simplified algorithm specification:

```
GIVEN AT DESIGN TIME: MAX_ITER
INPUTS: c_r, c_i;
OUTPUT: z_r, z_i;

z_r=c_r; z_i=c_i;
for(n=0; n<MAX_ITER; n++) {
  z_r' = z_r*z_r - z_i*z_i + c_r;
  z_i  = 2*z_r*z_i + c_i;
  z_r  = z_r';
}
```

Note that the loop runs for exactly MAX_ITER iterations (from 0 to MAX_ITER $- 1$. **For the following tasks, we always assume MAX_ITER $= 4$.** Please note that in the above algorithm, `z_r'` (`z_r` prime) in line 1 of the for loop is the new value of `z_r` in the next iteration of the loop, and is not used for computing the new `z_i` in line 2.

Task 1: Isomorphic Datapath

We start by sketching the isomorphic datapath of the Mandelbrot iteration with a fixed iteration number and by analyzing its area and timing. To this end, perform the following tasks:

- Draw the isomorphic datapath (remember that loops are fully unrolled). Add a register at the beginning (input) and at the end (output) of the datapath to store the output after the last iteration. In addition to the usual registers, logic gates, and multiplexers, you can use the following components (combinational arithmetic circuits), also given in Table 1: 2-input multiplier, 2-input adder, shift-left by 1 bit (multiplication by 2).
- Analyze your datapath's area using the area of the available components given in Table 1.
- Analyze the minimum clock period **and the throuhgput** of your datapath. The delays of the available components are given in Table 1. Please remember to use the CLK-Q delay for the critical path.

Task 2: Pipelining

To increase the throughput, we use pipelining. We start from the isomorphic datapath. Consider the following tasks/questions:

- What is the latency of the isomorphic datapath (not counting the registers at the beginning and the end)? Remember that latency can be expressed as a length of time or, in synchronous circuits, as a certain number of clock cycles.

Table 1: Components with area and delay.

Function	Inputs	Bits	Area [μm^2]	Delay [ns]
Adder/Subtractor	2	16	3600	0.5
Multiplier	2	16	3600	2.3
Shift	1	16	0	0
MUX	2	16	450	0.1
FlipFlop	1	16	450	CLK-Q: 0.2
				Setup/Hold: 0.0
AND/OR/...	2	1	0	0

- Pipeline the isomorphic datapath once (two pipeline stages, i.e., latency increases by one clock cycle compared to the original architecture). Insert the pipeline registers in your drawing (just use vertical black rectangles ▌ as registers).
- What are the area and the minimum clock period of the pipelined circuit with the additional pipeline stage? Please remember to use the CLK-Q delay for the critical path.
- Pipeline the isomorphic datapath three times (four pipeline stages, i.e., latency increases by three clock cycles compared to the original architecture). Insert the pipeline registers in your drawing (just use vertical black rectangles ▌ as registers).
- What are the area and the minimum clock period of the pipelined circuit with the total four pipeline stages?
- Why is pipelining two times (three pipeline stages) difficult?

Task 3: Iterative Decomposition

We would like to save area. To this end, start again from the isomorphic architecture. We first attempt to cut the area into half. To this end, consider the tasks/questions below:

- In the isomorphic architecture, identify and mark (enclose with a line) the largest sub-circuit that occurs twice and which is re-usable.
- Apply iterative decomposition and draw the new datapath that performs the same computation in two clock cycles. You can simplify your drawing by hiding the details of the sub-circuit you re-use by drawing it as a box (without the details inside). However, you must mark clearly define/mark the re-used sub-circuit in the isomorphic datapath.
- Analyze your datapath's area using the area of the available components given in Table 1.
- Analyze the minimum clock period **and the throuhgput** of your datapath. The delays of the available components are given in Table 1. Please remember to use the CLK-Q delay for the critical path.

We then attempt to cut the area by four. To this end, consider the tasks/questions below:

- In the isomorphic architecture, identify and mark (enclose with a line) the largest sub-circuit that occurs four times and which is re-usable.
- Apply iterative decomposition and draw the new datapath that performs the same computation in four clock cycles. You can simplify your drawing by hiding the details of the sub-circuit you re-use by drawing it as a box (without the details inside). However, you must mark clearly define/mark the re-used sub-circuit in the isomorphic datapath.
- Analyze your datapath's area using the area of the available components given in Table 1.

- Analyze the minimum clock period **and the throughput** of your datapath. The delays of the available components are given in Table 1. Please remember to use the CLK-Q delay for the critical path.

Finally we compare all three implementations:

- Provide the AT-product for all the circuit variants you have designed.