02132 Assignment 3 report

Implementation of an FSMD-based hardware accelerator for the image erosion in Chisel

Group: 22

Niclas Juul Schæffer **s224744**
Rasmus Kronborg Finnemann Wiuff **s163977**
github.com/rwiuff/02132Assignment3

December 1st

1 Work distribution

Table 1 shows the work distribution for this project.

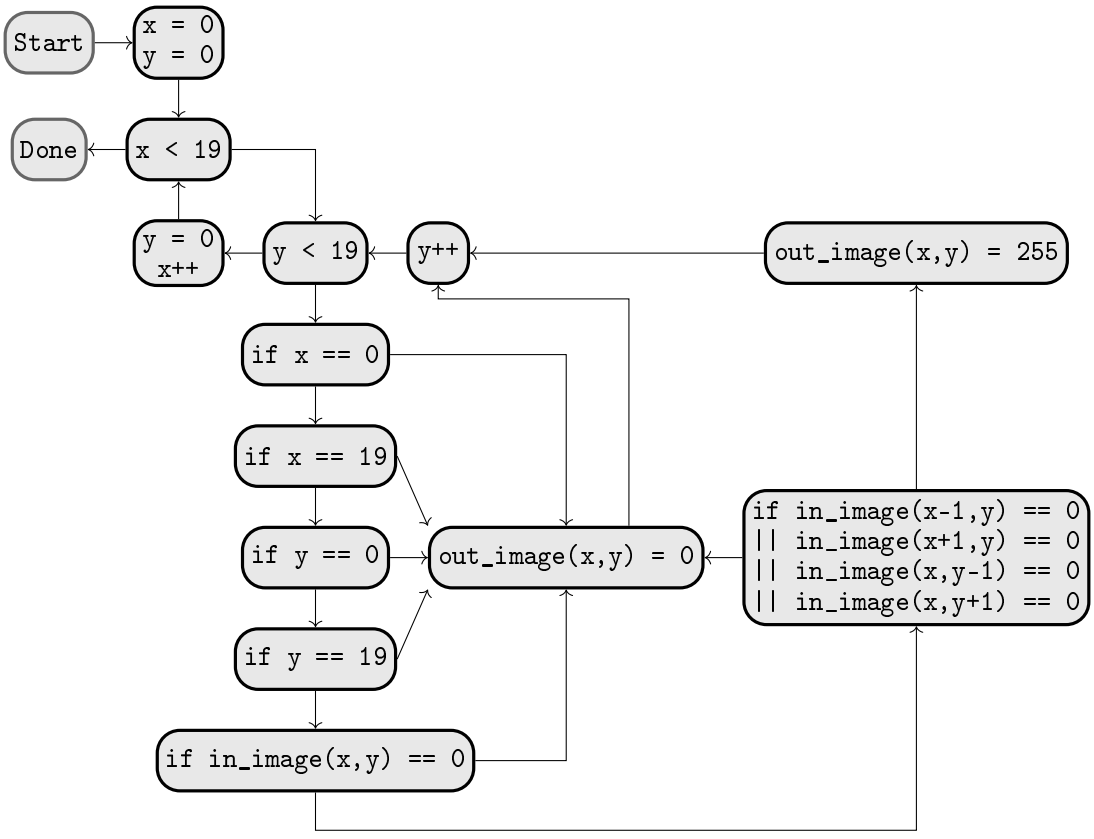
Table 1: *Work distribution on the project*

Name	Development tasks	Report tasks
Niclas Juul Schæffer		
Rasmus Wiuff		

2 Design

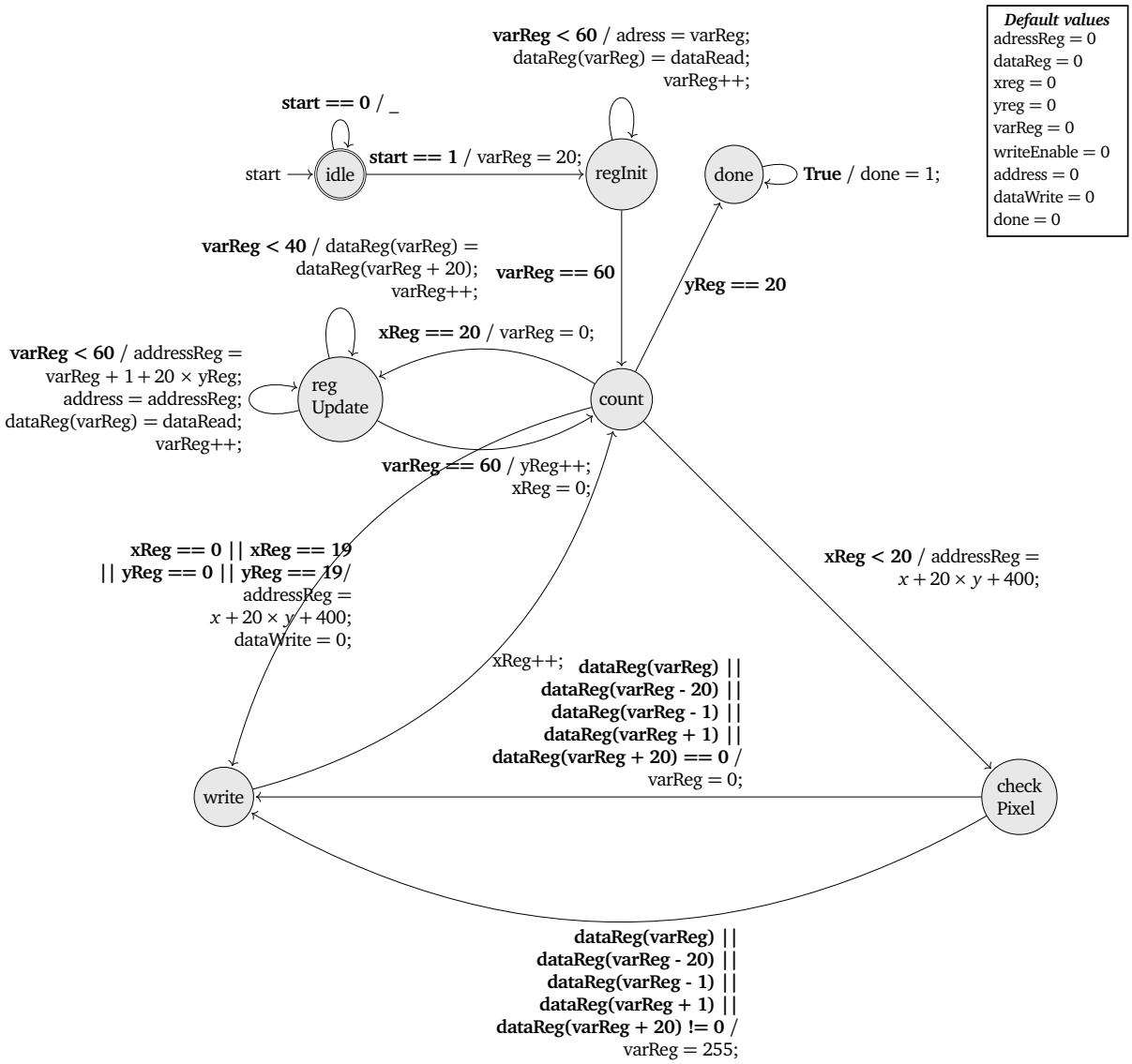
First a CFG was generated from the pseudocode in the assignment material, yielding the graph in Fig. 1.

Figure 1: *Control flow graph of erosion algorithm*



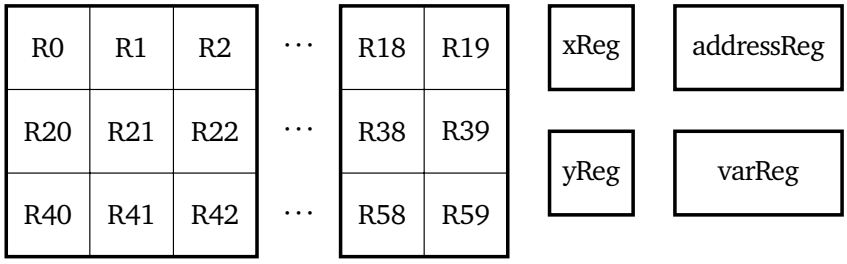
This gives a good idea of the needed states in the FSMD. On the getgo two obvious optimisations can be done here. The y incrementor is constant with both writing operations, and can be split into these. We also see that the bordercases could be replaced by a single state using a more complex logical ciruit. The inner pixel check can also be merged into this state as it is still on the same control path and leads to the zerorising of the output pixel or the check for neighbouring pixels. Fig. 2 shows the state diagram.

Figure 2: *State diagram of the erosion accelerator FSMD*



A list of registers are used to reduce read/write operations to/from the memory. The idea is to store three rows at a time, as to satisfy the erosion mask. On every iteration only one row is loaded into the register. The Register scheme is seen in Fig. 3.

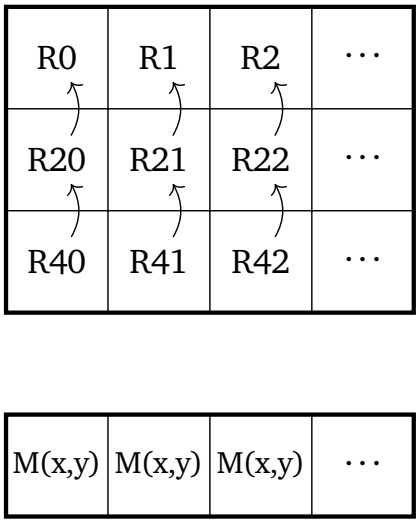
Figure 3: *The registers in the accelerator*



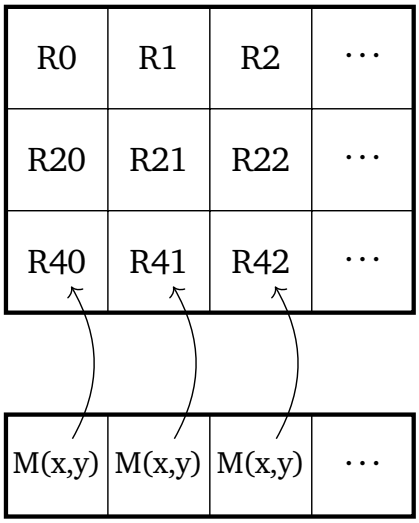
Furthermore registers are needed to store the memory address, the x and the y counters and a variable register to help with data register management. The state `regInit` initialises the data registers by filling R20-R59. The top row needs to be set to zero anyway. On every iteration the registers are updated as shown in Fig. 4.

Figure 4: *Data register update*

(a) *Image rows move up*



(b) *New image row from memory*



3 Implementation

Briefly discuss the implementation in Chisel of your design. You can include some code snippets if these are relevant to explain certain aspects of the implementation. In other words, try to answer the question "What does a reader need to know about your Chisel implementation?"
We implemented a feature where we could control the states with a Enum called stateReg which we make starts on idle. The way we use it its that all our functions is some when loops, that checks if the state is certain state, if so then run the specific function corresponds to that state. when everything have been ran, then the state becomes done.

```
// State enum and register
//     0 ::      1 ::     2 ::    3 ::         4 ::          5 ::     6 ::
val idle :: regInit :: count :: done :: regUpdate :: checkPixel :: write :: Nil =
  Enum(7)
val stateReg = RegInit(idle)
```

4 Test And Evaluation

Report here the results from the test you have carried out. Present how you have tested (paper and pencil testing) the FSMD you have designed. Present the tests you have developed (if any). Remember to discuss the relusts and the test you have carried out, do not just present them, but explain and argue their meaning. Adress the design evaluation questions in Task 6 in the Assignment 3 document.