MASTER OF SCIENCE IN ENGINEERING

Final Exam Michaelmass 2011/12

Modul / *Module*: <*Design of Embedded Hardware and Firmwares*>
Datum / *Date*: <*Michaelmass 2011*>
Dozierende/n / *Teacher/s*: <*Doran, Hans Dermot* >
Teil <1 > von <1 > *(oder Aufgaben x-y von z)*

Name / Last name: _____

Vorname / First name: _____

FH / UAS: _____

Question 1: Hardware Optimizations:

300 bytes have to be transferred between address 0x3004 and 0x6003 on a 32-bit bus. In terms of efficiency, and if necessary referring to the Avalon bus, analyse the issues in using standard DMA (7P). Suggest an optimal solution (3P). Assess whether a CI or a purpose built DMA controller or possibly hardware acceleration would be useful in implementing the operation (3P).

The transfer is between a 32-bit aligned address and a MSb(yte) aligned address (1P) One transfer is required to collect the data (1P) and, depending on slave capability (1P), a minimum of two transfers required to write the data (1P). This makes for 900 bus cycles (1P) not including time for initialising the DMA (1P) or servicing the interrupt when DMA is finished (1P).

If the first byte is written in a byte cycle then words thereafter can be written as long words (1P). Some solution using shift registers or simple wiring (2P).

Some assessment of CI, f.i.: A custom instruction would probably take too long and hold up the processor – DMA is not supposed to affect processor operation (1P).

Some assessment of HA f.i.: Hardware acceleration is generally understood to be used for implementing more complex operations on the data than a simple shift function. (1P)

Some assessment of modified DMA f.i.: a purpose built DMA controller would be the solution based on negating the other two options (1P)

Name *Last name* ..... Vorname *First name* .....

Question 2: Software Optimisations:

Here is some code that is supposed to multiply 2 matrices:

```
matrixmult(a,b,c)
float a[4][4], b[4][4], c[4][4];
{
    int x, y;
    float temp[4][4];

    for(y=0; y<4 ; y++)
        for(x=0 ; x<4 ; x++) {
            temp[y][x] = b[y][0] * a[0][x]
                       + b[y][1] * a[1][x]
                       + b[y][2] * a[2][x]
                       + b[y][3] * a[3][x];
        }
    for(y=0; y<4; y++)
        for(x=0; x<4; x++)
            c[y][x] = temp[y][x];
}
```

Comment on this code and the application of loop unrolling (3P), in-lining (1P), custom instructions (3P) and cache-aware programming (3P). (Bonus points possible)

Comments such as:

Unrolling the inner loop would increase the instruction space by roughly 4, unrolling the outer loop would increase the instruction space by 16 (1P). Its not obvious what the second assignment loops are for (1P).

In-lining doesn't come to terms because no function is called (1P)

Custom instruction: in theory possible but due to floating point calculations four parallel calculations are probably unfeasible (1P), one FPU might be feasible but then at least 4 FPU clocks are required to complete calculations (1P) assuming that additions take one clock (1P). It might be feasible to implement a custom instruction for the additions only. (1P)

Cache awareness depends on how large the code is – that is whether the whole unrolled loop can be fitted into a cache block or whether just the inner loop (1P). Cache awareness also has data relevance given that the b matrix is required 4 times and in the outer loop the a matrix as well. (1P) Blocking techniques could be used here (2P)

Name
Last name .....

Vorname
First name .....

Question 3: Varia:

Read the docs