

# Labrapport 4

TSEA44

Jonathan Karlsson, Niclas Olofsson, Paul Nedstrand  
jonka293, nicol271, paune  
Grupp 2

24 januari 2014

## 1 Inledning

Det fjärde och sista miniprojektet i kursen, lab 4, handlar om att skapa en egen assembler-instruktion för skrivning till minnet. Syftet med denna instruktion är att snabba upp koden för att Huffman-koda den resulterade bilden från vår jpeg-accelerator.

Instruktionen tar en längd samt ett data som argument, och sparar all inkommen data i en buffer tills minst 8 bitar erhållits. I detta fall skrivs den buffrade datan till minnet; en byte åt gången och så många bytes som möjligt (maximalt två).

## 2 Design

- How does your hardware work?

## 3 Resultat

### 3.1 Verifiering av hårdvarans funktion

För att kontrollera att vår hårdvara fungerade, började vi med att anropa vår instruktion från det monitor-program som körs när datorn startar. Efter en del felsökning och justering övergick vi till att testa koden på en FPGA med samma monitor-program. Vi använde monitorns inbyggda kommando för att visa minnesadresser för att verifiera att rätt data skrevs till minnet. Ganska snabbt insåg vi behovet av att kunna felsöka även i denna miljö utan att behöva syntetisera om koden varje gång, och skrev därför testprogrammet `asm.c` som vi kunne ladda in i minnet och köra via monitorn.

Det största problem vi hade under denna lab, och även det svåraste vi fått under labkursen, var att sista biten i varje byte vi skrev till minnet blev fel. Till skillnad från de tidigare fel vi fått under kursen så fick vi varken några varningar av värde vid syntetiseringen, konstiga odefinierade signaler eller märkliga läs/skrivcykler vid simulering.

Till sist testades även hårdvaran genom att instruktionen användes i jpegtest.

jchuff.c modifierades, för att använda set bit-instruktionen för skrivning till minnet, och en korrekt bild genererades.

### 3.2 Prestanda och FPGA-användning

Vi använde vårt testprogram och den prestandaräknare som inkluderades i jpegtest för att mäta prestandan på några olika slags anrop till vår set bit-instruktion. Dels varierade vi storleken, dels testade vi även att enbart skriva 0xFF, vilket i vår implementation gör två minnesskrivningar.

Storlek	Data	Antal klockcykler
2	0x00	10
8	0x00	13
8	0xFF	17
16	0x00	17

Tabell 1: Antal klockcykler för anrop av vår sbit-instruktion (medelvärde av 100 försök)

Vid användning i JPEG-acceleratorn jämförde vi utskriften av prestandamätningen för en version av programmet som kompilerades utan set bit-instruktionen (2), mot resultatet med denna instruktion påslagen (3).

Beskrivning	Antal klockcykler
Main program	25 847 242
Init	6 600 044
Encode_image	19 247 198
Forward_DCT	6 489 189
Copy	0
DCT kernel	0
Quantization	6 489 189
Huffman encoding	12 226 950
Emit_bits	4 983 905

Tabell 2: Prestanda för jpegtest utan sbit-instruktionen

Beskrivning	Antal klockcykler
Main program	21 754 287
Init	6 653 818
Encode_image	15 100 469
Forward_DCT	6 396 157
Copy	0
DCT kernel	0
Quantization	6 396 157
Huffman encoding	8 126 873
Emit_bits	1 306 129

Tabell 3: Prestanda för jpegtest med sbit-instruktionen

- What is the performance with and without the set bit hardware? This should include measurements of both the entire application and the set bit instruction by itself, assuming good code in a software implementation (take a look at how the software solution in jpegfiles).
- How much of the FPGA does your hardware use?
- What is the performance of your final system?

## 4 Slutsats

## 5 Appendix: Källkod

## 6 What to Include in the Lab Report 4

The lab report should contain all source code that you have written. (The source code should of course be commented.) We would also like you to include a block diagram of your hardware. If you have written any FSM you should include a state graph of the FSM. We would also like you to discuss the following questions in detail somewhere in your lab report:

- How would your design change if you had to achieve even higher speed using more hardware?
- How would your design change if you had to use less hardware at the cost of a slower solution?
- What are the problems with using your new hardware in a multitasking operating system? How can the problem(s) be solved?

And of course, the normal parts of a lab report such as a table of contents, an introduction, a conclusion, etc. The source code that you have written should be included in appendices and referred to from the main document.