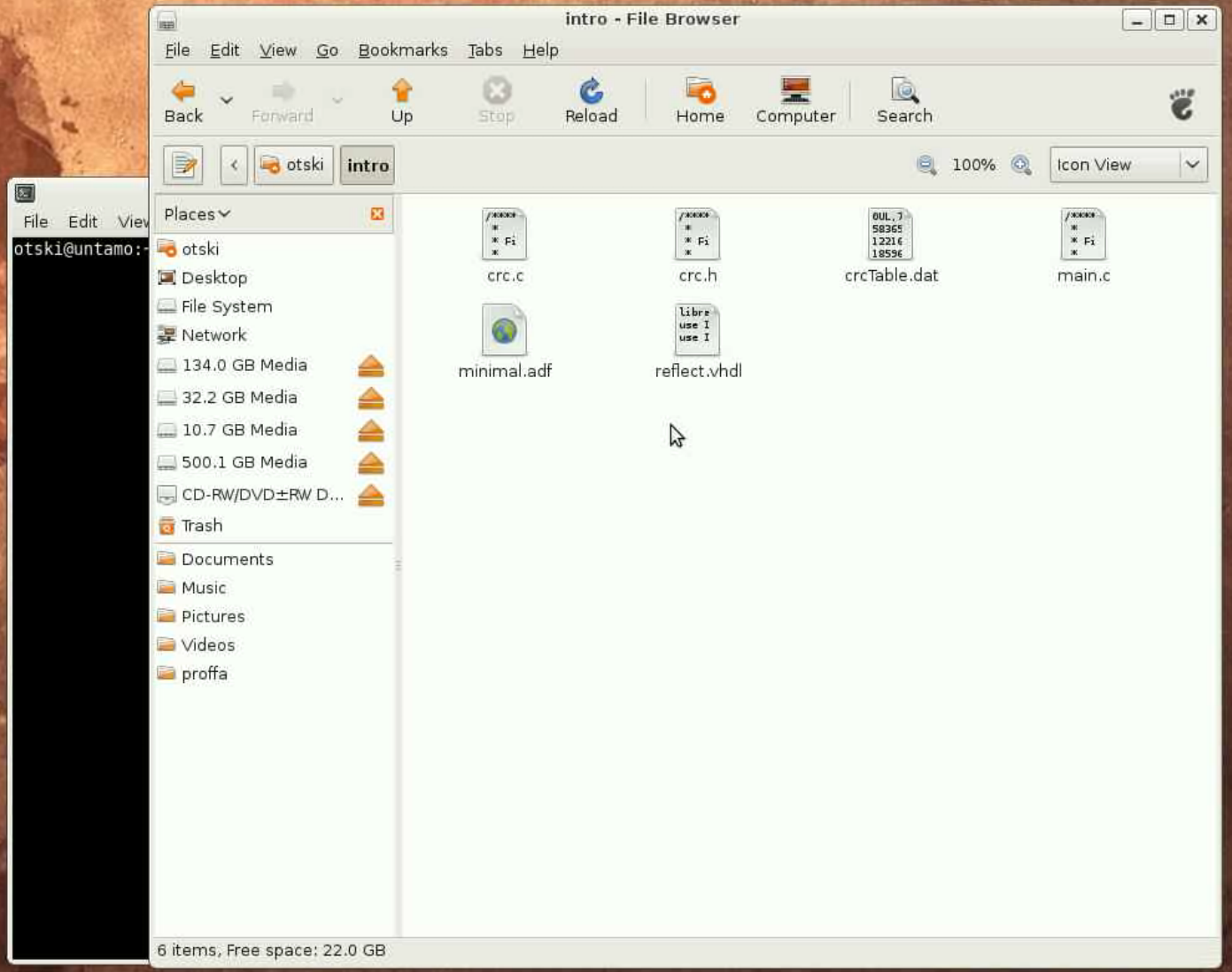
TCE Tour: Screencast

- These are accompanying slides for a set of TCE screencast clips available at http://tce.cs.tut.fi/index.php/home/screencasts
- The set of videos goes through the most important tools in TCE by means of a simple CRC example application
- Starts from C, ends with a TTA+program running on an FPGA board

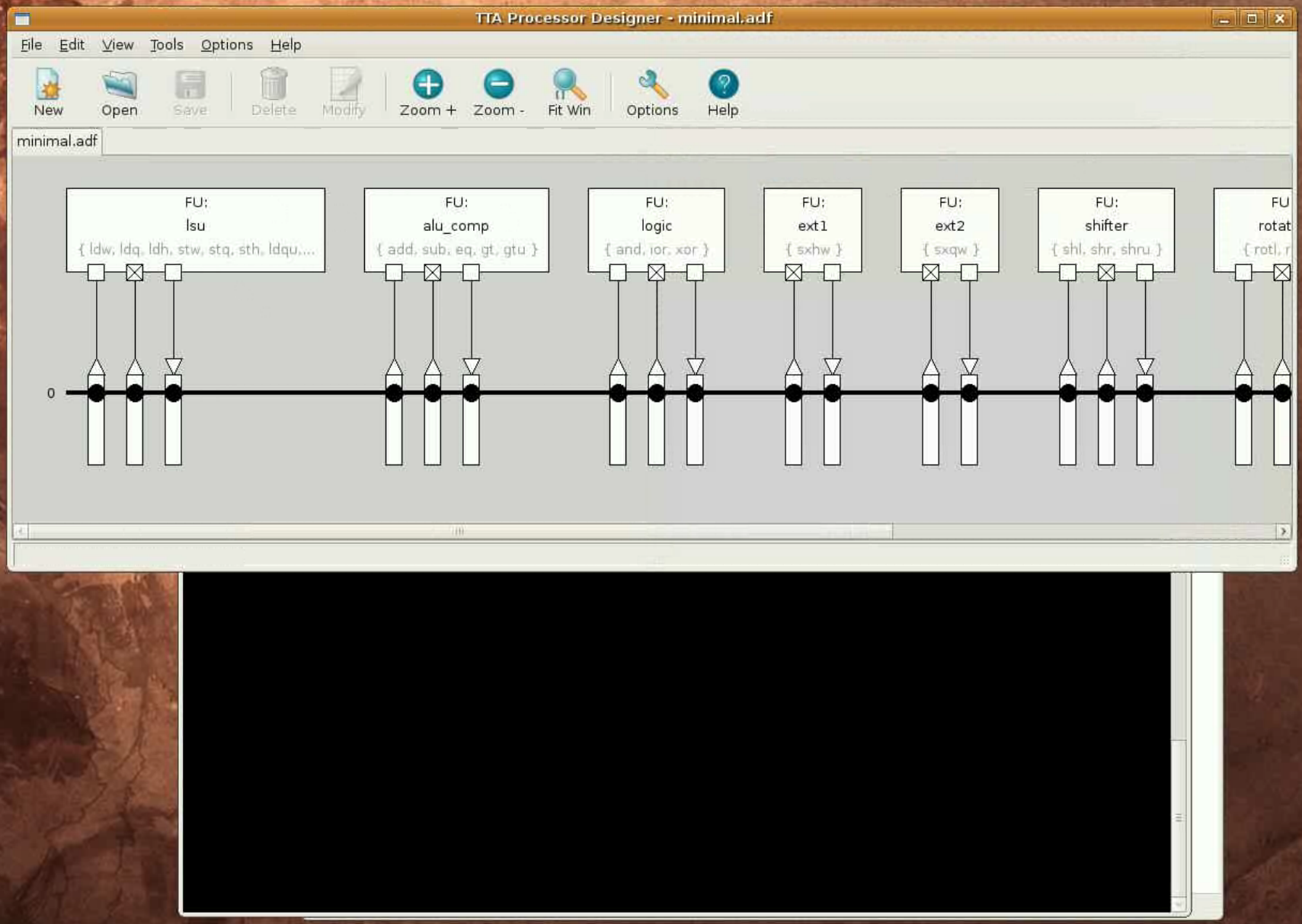
Intro and Exploration (about 7 minutes)

- Start with the application code in C
- minimal.adf is shipped in TCE:
  - a minimal set of resources in TTAs supported by the tcecc compiler
- reflect.vhdl is a VHDL implementation of a custom operation which will be added later to the design


intro - File Browser
File Edit View Go Bookmarks Tabs Help
Back Forward Up Stop Reload Home Computer Search
otski intro
100%
Icon View
Places
otski
Desktop
File System
Network
134.0 GB Media
32.2 GB Media
10.7 GB Media
500.1 GB Media
CD-RW/DVD±RW D...
Trash
Documents
Music
Pictures
Videos
proffa
crc.c
crc.h
crcTable.dat
main.c
minimal.adf
reflect.vhdl
6 items, Free space: 22.0 GB
File Edit
otski@untamo:

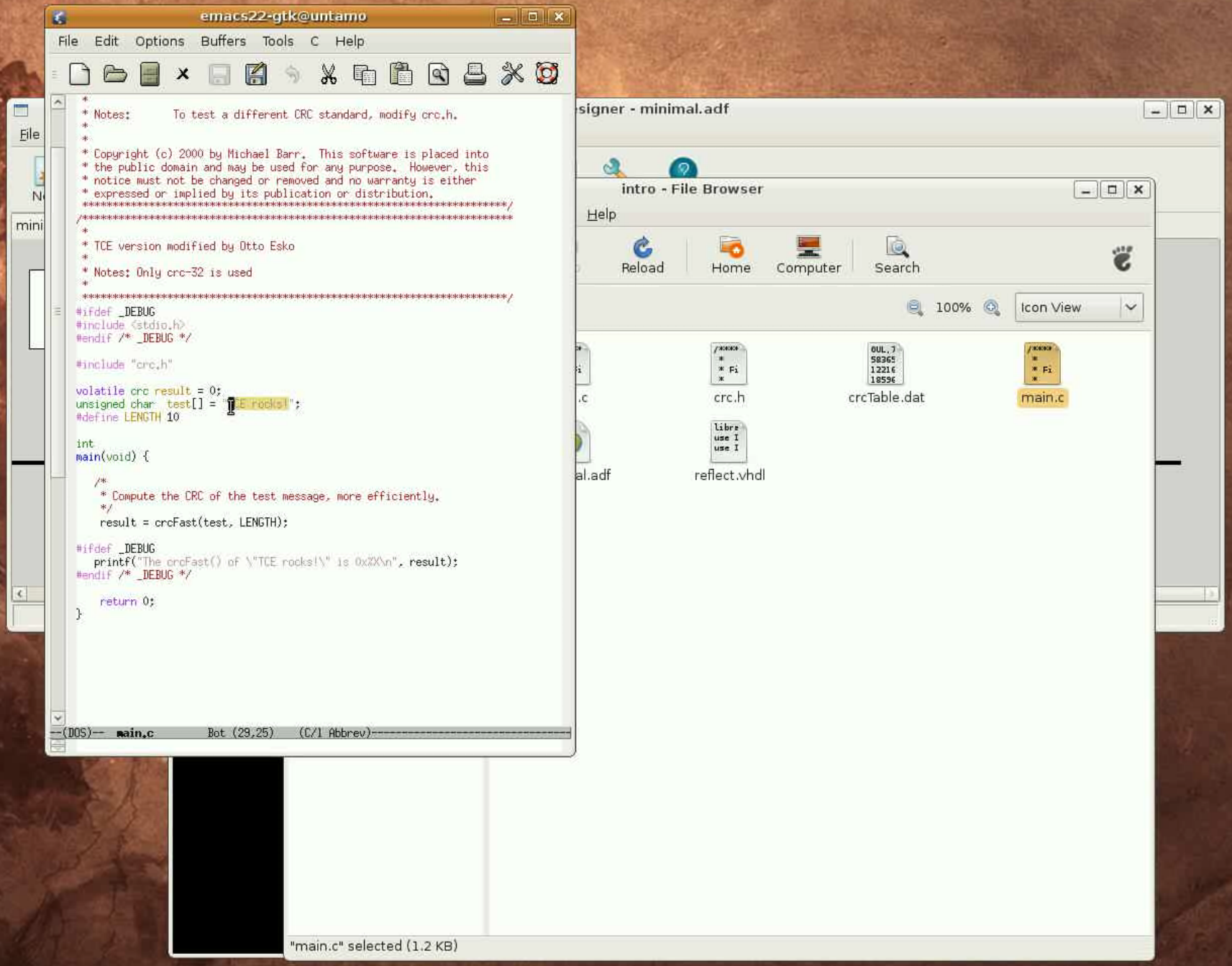
Intro and Exploration

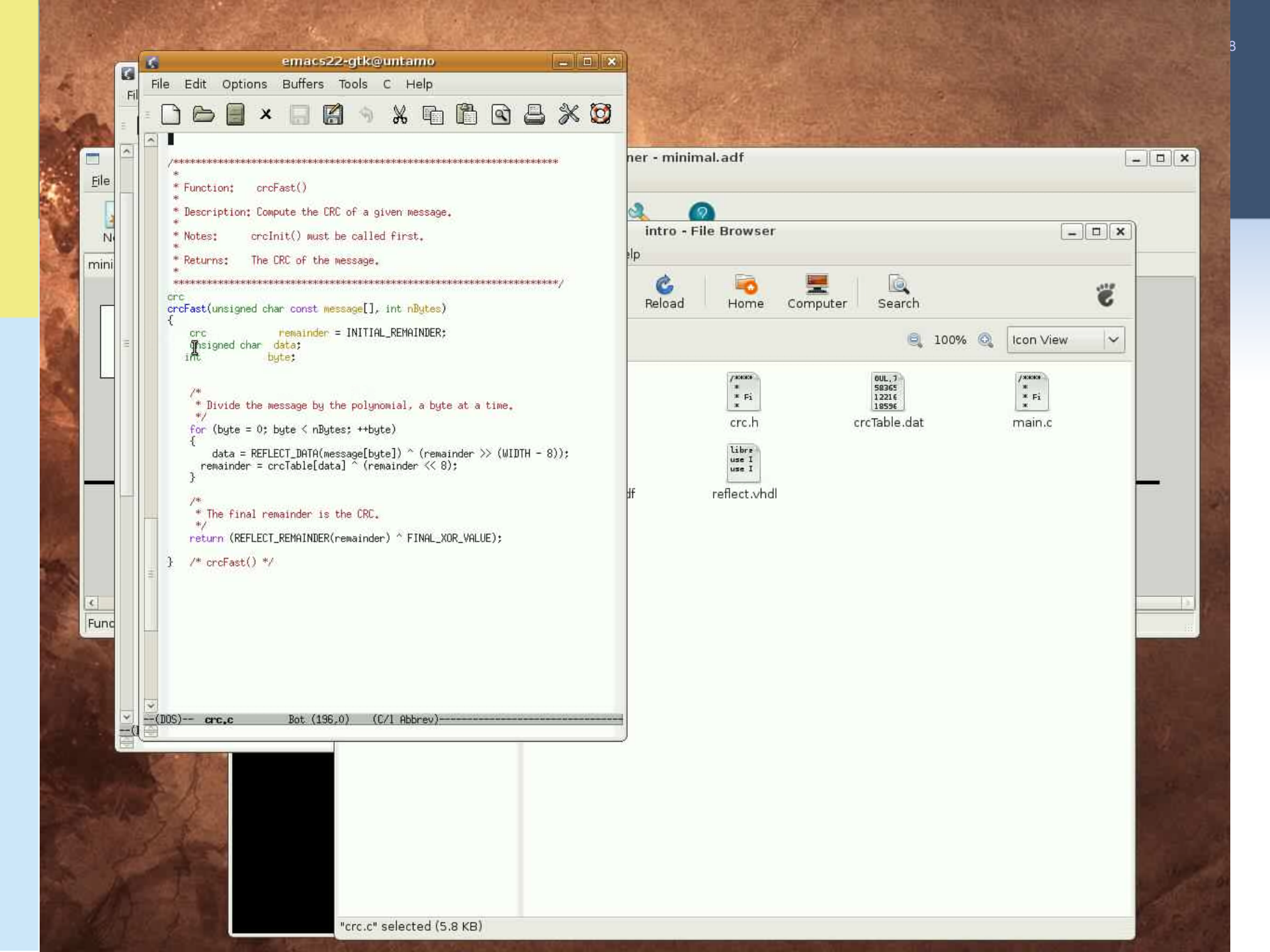
- Open the minimal.adf to the Processor Design GUI
  - “prode minimal.adf &”


TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
minimal.adf
FU:
lsu
{ ldw, ldq, ldh, stw, stq, sth, ldqu,...
FU:
alu_comp
{ add, sub, eq, gt, gtu }
FU:
logic
{ and, ior, xor }
FU:
ext1
{ sxhw }
FU:
ext2
{ sxqw }
FU:
shifter
{ shl, shr, shru }
FU
rotat
{ rotl, r
0

Intro and Exploration

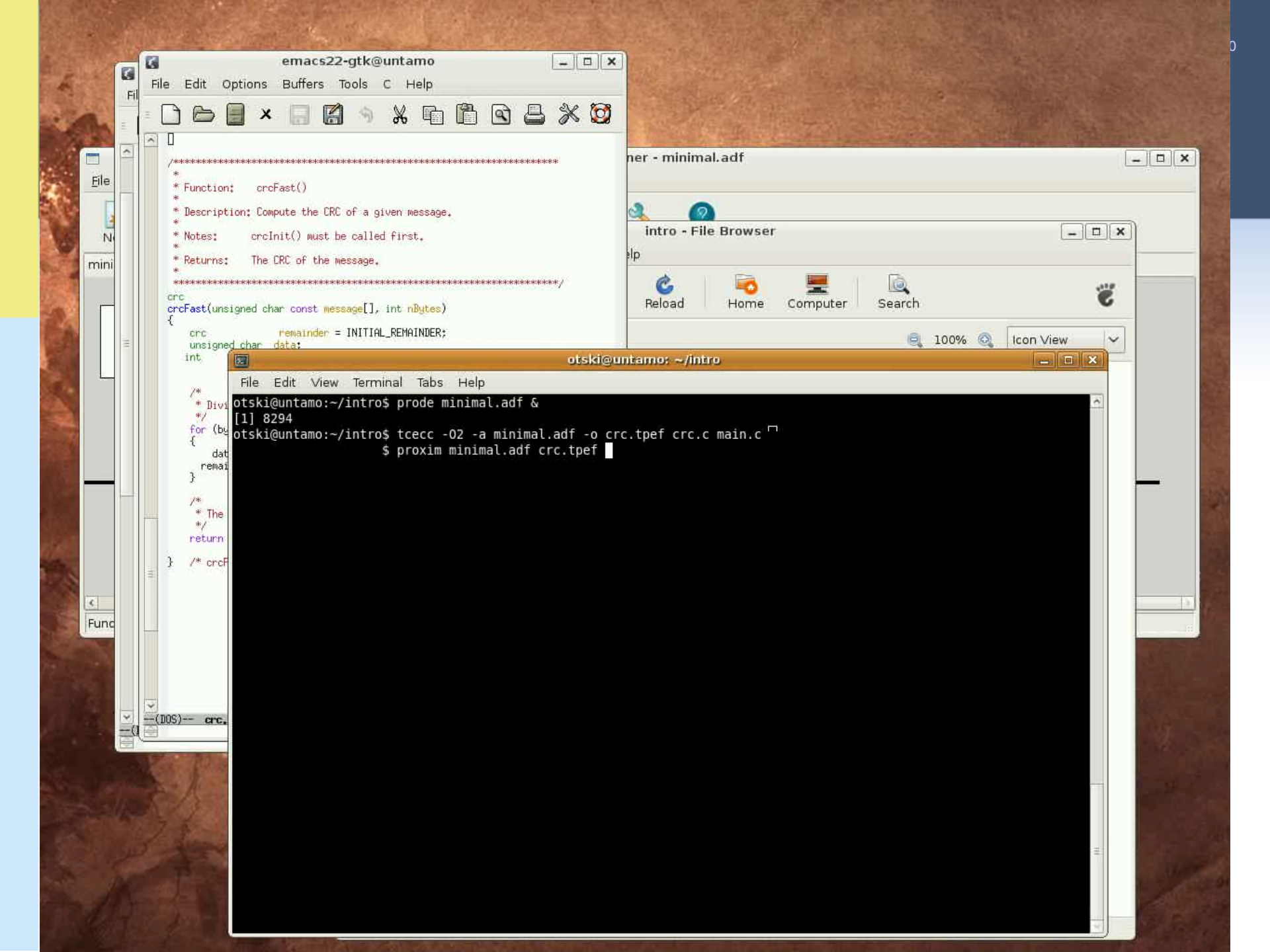
- The main.c has a simple string “TCE rocks!” for which we are going to compute the CRC


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
* Notes: To test a different CRC standard, modify crc.h.
* Copyright (c) 2000 by Michael Barr. This software is placed into
* the public domain and may be used for any purpose. However, this
* notice must not be changed or removed and no warranty is either
* expressed or implied by its publication or distribution.
* TCE version modified by Otto Esko
* Notes: Only crc-32 is used
#ifdef _DEBUG
#include <stdio.h>
#endif /* _DEBUG */
#include "crc.h"
volatile crc result = 0;
unsigned char test[] = "TCE rocks!";
#define LENGTH 10
int
main(void) {
/*
* Compute the CRC of the test message, more efficiently.
*/
result = crcFast(test, LENGTH);
#ifdef _DEBUG
printf("The crcFast() of \"TCE rocks!\" is 0xXX\n", result);
#endif /* _DEBUG */
return 0;
}
--(DOS)-- main.c Bot (29,25) (C/l Abbrev)
signer - minimal.adf
intro - File Browser
Help
Reload
Home
Computer
Search
100%
Icon View
crc.h
crcTable.dat
main.c
reflect.vhdl
"main.c" selected (1.2 KB)


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
/***
 *
 * Function: crcFast()
 *
 * Description: Compute the CRC of a given message.
 *
 * Notes: crcInit() must be called first.
 *
 * Returns: The CRC of the message.
 *
 ***/
crc
crcFast(unsigned char const message[], int nBytes)
{
    crc   remainder = INITIAL_REMAINDER;
    unsigned char data;
    int   byte;
    /*
     * Divide the message by the polynomial, a byte at a time.
     */
    for (byte = 0; byte < nBytes; ++byte)
    {
        data = REFLECT_DATA(message[byte]) ^ (remainder >> (WIDTH - 8));
      remainder = crcTable[data] ^ (remainder << 8);
    }
    /*
     * The final remainder is the CRC.
     */
    return (REFLECT_REMAINDER(remainder) ^ FINAL_XOR_VALUE);
}   /* crcFast() */
--(DOS)-- crc.c Bot (196,0) (C/l Abbrev)
intro - File Browser
Reload
Home
Computer
Search
100%
Icon View
crc.h
crcTable.dat
main.c
reflect.vhdl
"crc.c" selected (5.8 KB)

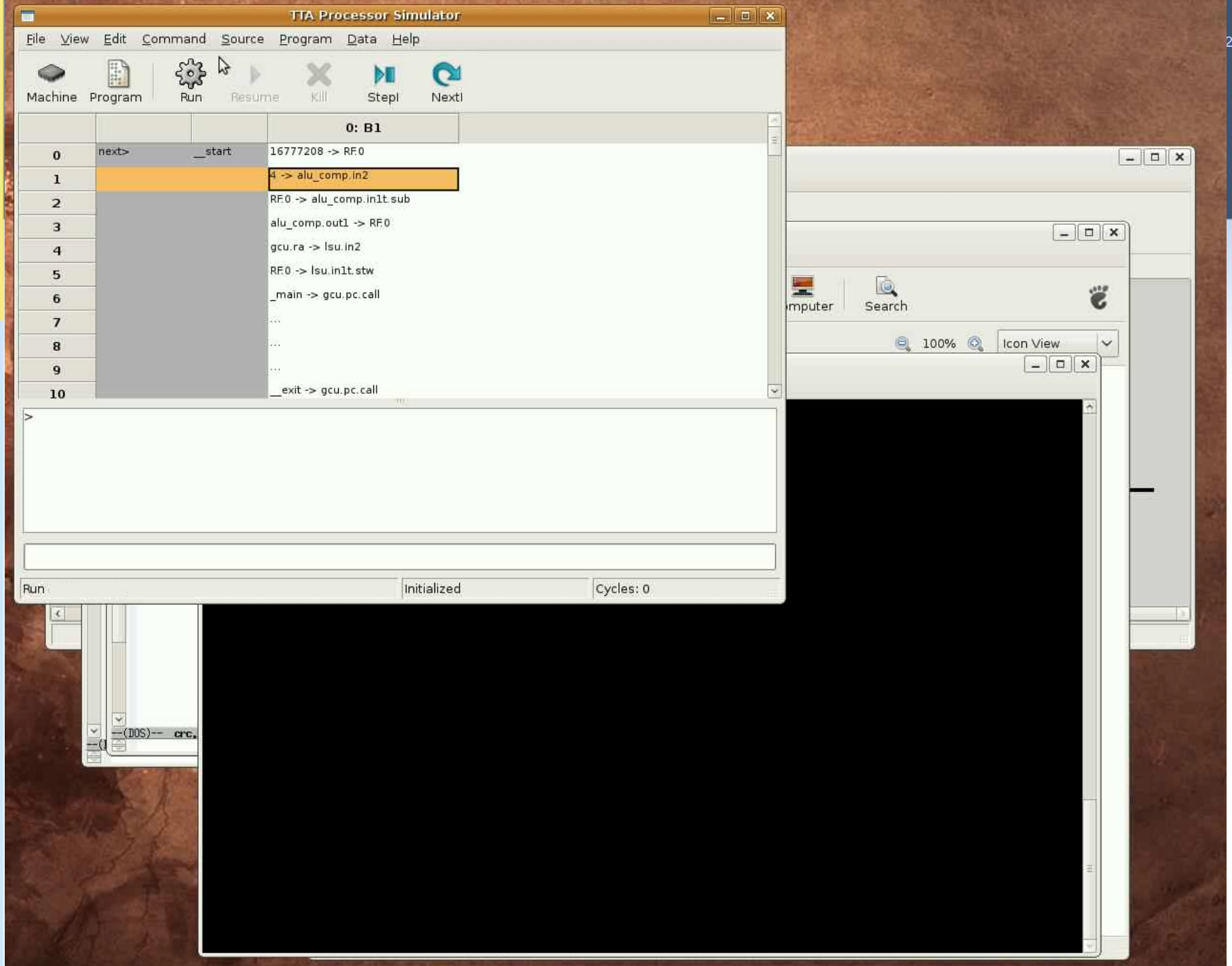
Intro and Exploration

- Let's see how well the CRC code runs with the smallest supported TTA
  - Compile the code to the TTA with the retargetable tcecc compiler
  - Load the processor architecture description and the compiled program to the Processor Simulator GUI (Proxim)


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
/***
*
* Function: crcFast()
*
* Description: Compute the CRC of a given message.
*
* Notes: crcInit() must be called first.
*
* Returns: The CRC of the message.
*
***/
crc
crcFast(unsigned char const message[], int nBytes)
{
crc remainder = INITIAL_REMAINDER;
unsigned char data;
int
/*
* Divi
*/
for (by
{
dat
remai
}
/*
* The
*/
return
} /* crcF
--(DOS)-- crc.
ner - minimal.adf
intro - File Browser
Reload
Home
Computer
Search
100%
Icon View
otski@untamo: ~/intro
File Edit View Terminal Tabs Help
otski@untamo:~/intro$ prode minimal.adf &
[1] 8294
otski@untamo:~/intro$ tcecc -O2 -a minimal.adf -o crc.tpef crc.c main.c
$ proxim minimal.adf crc.tpef

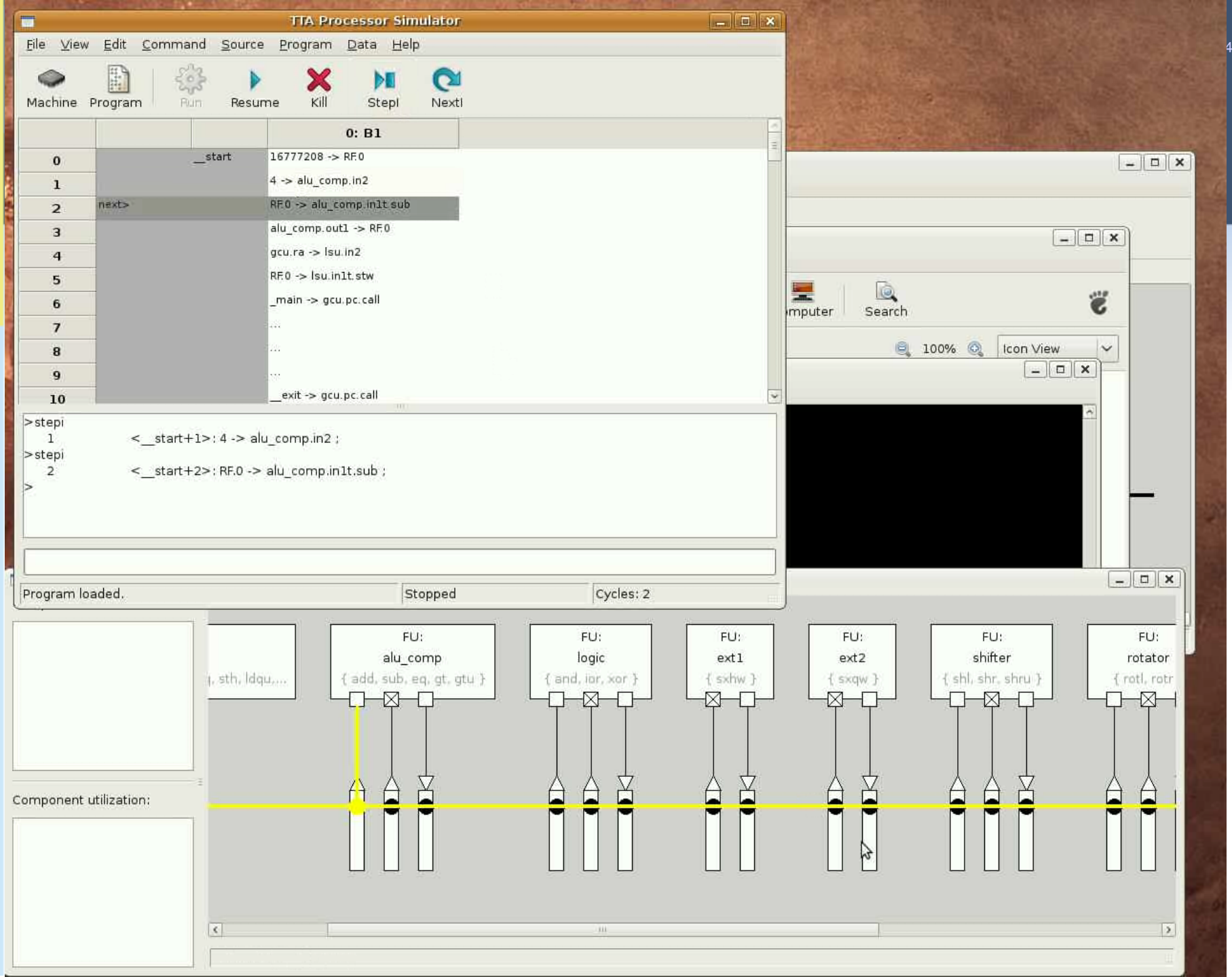
Intro and Exploration

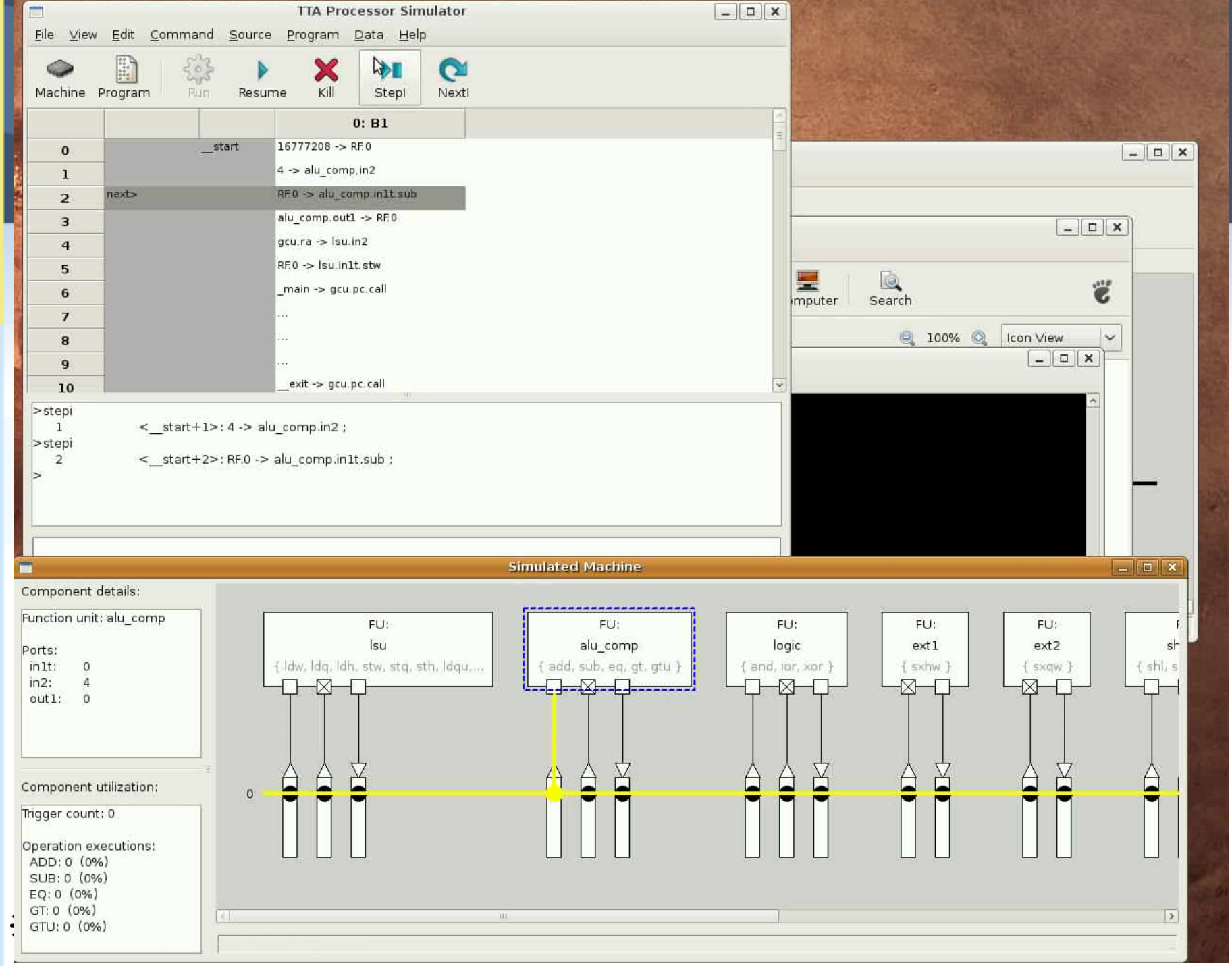
- Proxim's main window displays the disassembly of the TTA program
- The minimal.adf has only one bus, thus the moves cannot be parallelized

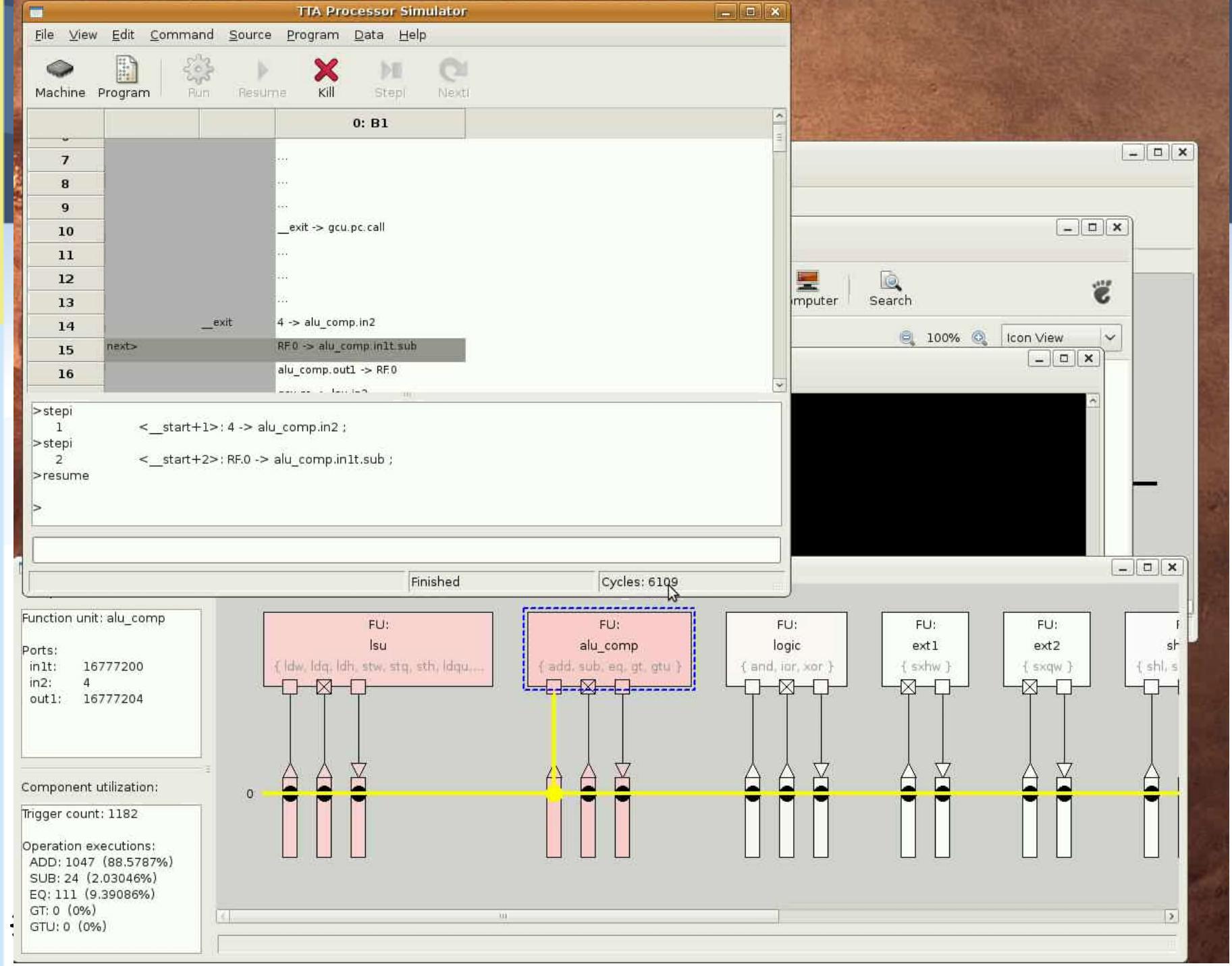

TTA Processor Simulator
File View Edit Command Source Program Data Help
Machine Program Run Resume Kill StepI NextI
0: B1
0 next> __start 16777208 -> RF.0
1 4 -> alu_comp.in2
2 RF.0 -> alu_comp.in1t.sub
3 alu_comp.out1 -> RF.0
4 gcu.ra -> lsu.in2
5 RF.0 -> lsu.in1t.stw
6 _main -> gcu.pc.call
7 ...
8 ...
9 ...
10 __exit -> gcu.pc.call
>
Run
Initialized
Cycles: 0
Search
100%
Icon View

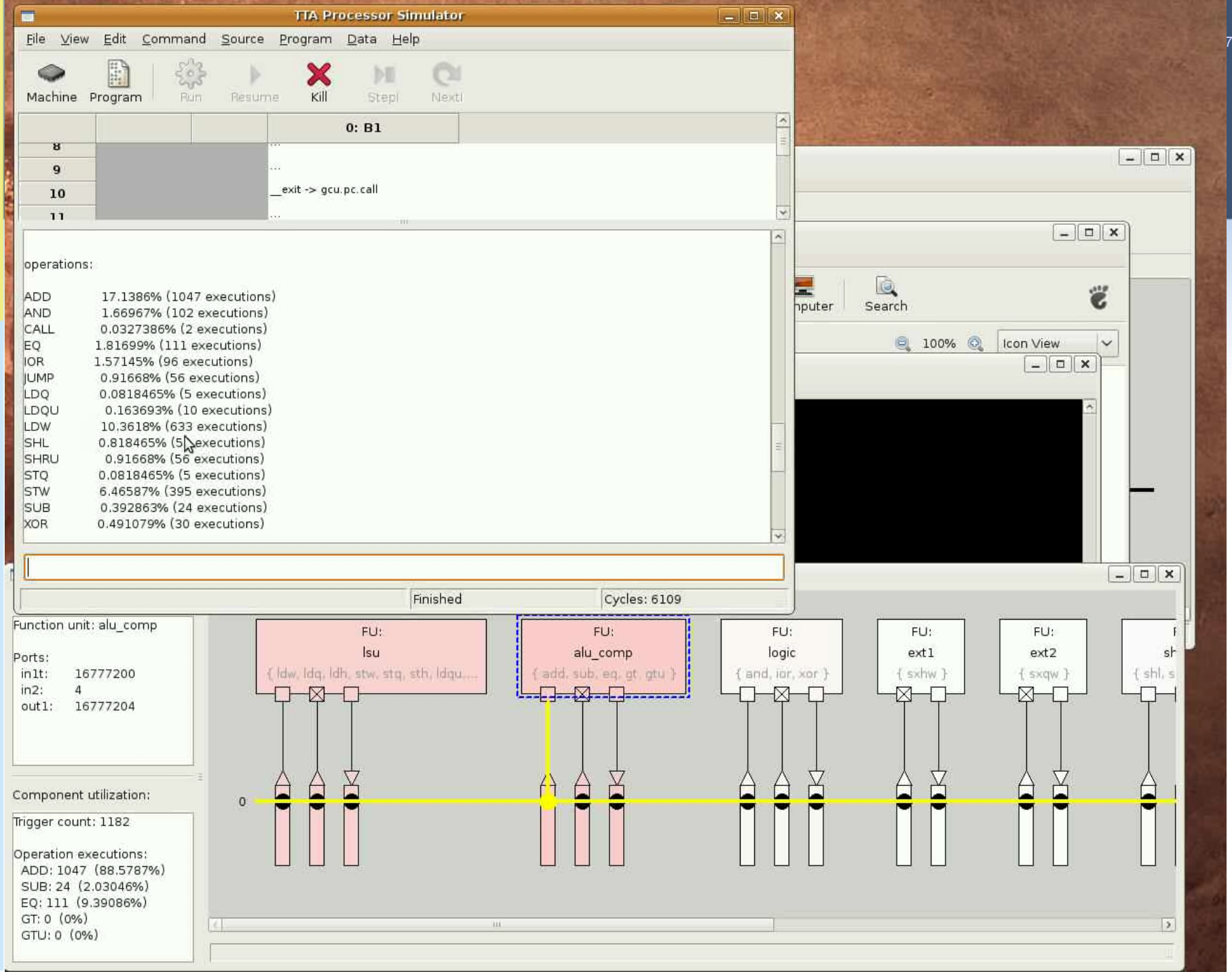
Intro and Exploration

- Proxim's machine window:
  - Visualizes the TTA processor when running the given program
  - Single stepping the assembly code highlights the transport paths in the processor accessed by the moves in the current instruction
  - Allows inspecting the values in programmer-visible registers of the TTA such as FU ports, the utilization of the components (color coding), etc.
  - Most importantly for this case, the simulator displays the total cycle count **6109** (number of TTA instructions executed)
  - Also statistics for the different operations executed, registers used etc. can be produced to guide manual exploration of the architecture
    - "info proc stats"


TTA Processor Simulator
File View Edit Command Source Program Data Help
Machine Program Run Resume Kill StepI NextI
0: B1
0 __start 16777208 -> RF.0
1 4 -> alu_comp.in2
2 next> RF.0 -> alu_comp.in1t.sub
3 alu_comp.out1 -> RF.0
4 gcu.ra -> lsu.in2
5 RF.0 -> lsu.in1t.stw
6 _main -> gcu.pc.call
7 ...
8 ...
9 ...
10 __exit -> gcu.pc.call
>stepi
1 <__start+1>: 4 -> alu_comp.in2 ;
>stepi
2 <__start+2>: RF.0 -> alu_comp.in1t.sub ;
>
Program loaded.
Stopped
Cycles: 2
Search
100%
Icon View
FU: alu_comp { add, sub, eq, gt, gtu }
FU: logic { and, ior, xor }
FU: ext1 { sxhw }
FU: ext2 { sxqw }
FU: shifter { shl, shr, shru }
FU: rotator { rotl, rotr
Component utilization:

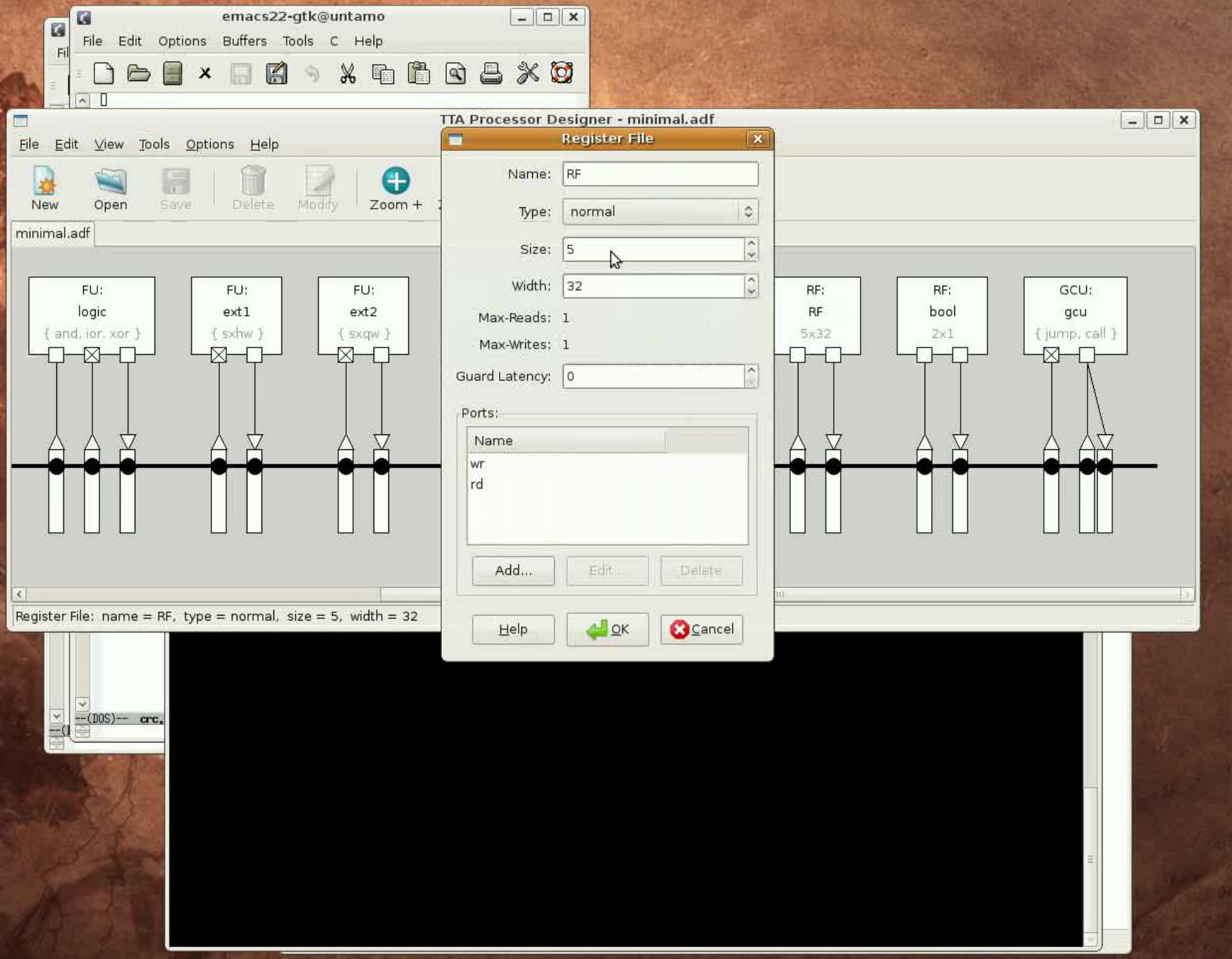

TTA Processor Simulator
File View Edit Command Source Program Data Help
Machine Program Run Resume Kill Stepi Nexti
0: B1
0 __start 16777208 -> RF.0
1 4 -> alu_comp.in2
2 next> RF.0 -> alu_comp.in1t.sub
3 alu_comp.out1 -> RF.0
4 gcu.ra -> lsu.in2
5 RF.0 -> lsu.in1t.stw
6 _main -> gcu.pc.call
7 ...
8 ...
9 ...
10 __exit -> gcu.pc.call
>stepi
1 <__start+1>: 4 -> alu_comp.in2 ;
>stepi
2 <__start+2>: RF.0 -> alu_comp.in1t.sub ;
>
Search
100%
Icon View
Simulated Machine
Component details:
Function unit: alu_comp
Ports:
in1t: 0
in2: 4
out1: 0
Component utilization:
Trigger count: 0
Operation executions:
ADD: 0 (0%)
SUB: 0 (0%)
EQ: 0 (0%)
GT: 0 (0%)
GTU: 0 (0%)
FU: lsu { ldw, ldq, ldh, stw, stq, sth, ldqu,... }
FU: alu_comp { add, sub, eq, gt, gtu }
FU: logic { and, ior, xor }
FU: ext1 { sxhw }
FU: ext2 { sxqw }
0

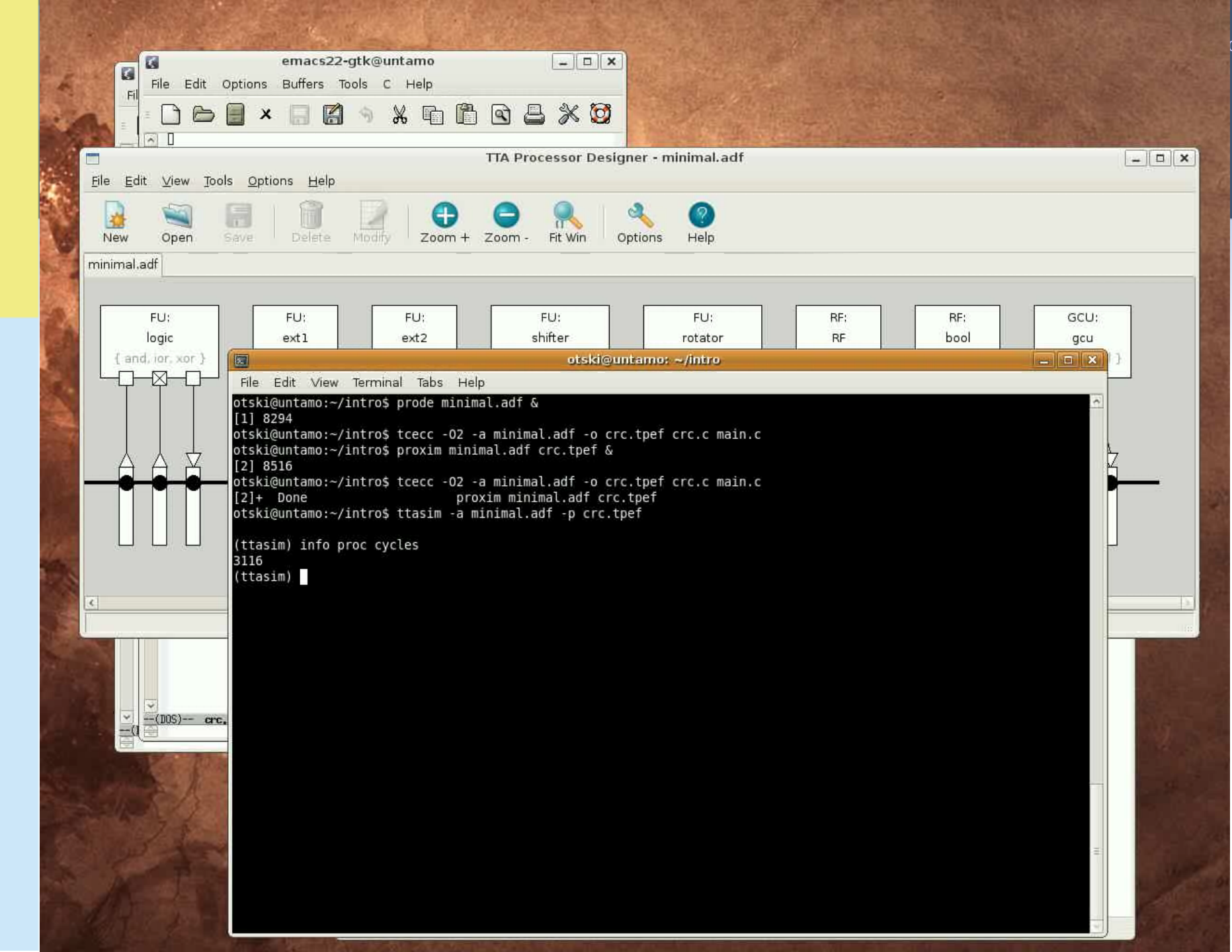

TTA Processor Simulator
File View Edit Command Source Program Data Help
Machine Program Run Resume Kill Stepi Nexti
0: B1
7 ...
8 ...
9 ...
10 __exit -> gcu.pc.call
11 ...
12 ...
13 ...
14 __exit 4 -> alu_comp.in2
15 next> RF.0 -> alu_comp.in1t.sub
16 alu_comp.out1 -> RF.0
>stepi
1 <__start+1>: 4 -> alu_comp.in2 ;
>stepi
2 <__start+2>: RF.0 -> alu_comp.in1t.sub ;
>resume
>
Finished
Cycles: 6109
Function unit: alu_comp
Ports:
in1t: 16777200
in2: 4
out1: 16777204
Component utilization:
Trigger count: 1182
Operation executions:
ADD: 1047 (88.5787%)
SUB: 24 (2.03046%)
EQ: 111 (9.39086%)
GT: 0 (0%)
GTU: 0 (0%)
Search
100%
Icon View
FU: lsu { ldw, ldq, ldh, stw, stq, sth, ldqu,... }
FU: alu_comp { add, sub, eq, gt, gtu }
FU: logic { and, ior, xor }
FU: ext1 { sxhw }
FU: ext2 { sxqw }
0


TTA Processor Simulator
File View Edit Command Source Program Data Help
Machine Program Run Resume Kill Step! Next!
0: B1
8
9
10
11
__exit -> gcu.pc.call
operations:
ADD 17.1386% (1047 executions)
AND 1.66967% (102 executions)
CALL 0.0327386% (2 executions)
EQ 1.81699% (111 executions)
IOR 1.57145% (96 executions)
JUMP 0.91668% (56 executions)
LDQ 0.0818465% (5 executions)
LDQU 0.163693% (10 executions)
LDW 10.3618% (633 executions)
SHL 0.818465% (5 executions)
SHRU 0.91668% (56 executions)
STQ 0.0818465% (5 executions)
STW 6.46587% (395 executions)
SUB 0.392863% (24 executions)
XOR 0.491079% (30 executions)
Finished
Cycles: 6109
Search
100%
Icon View
Function unit: alu_comp
Ports:
in1t: 16777200
in2: 4
out1: 16777204
Component utilization:
Trigger count: 1182
Operation executions:
ADD: 1047 (88.5787%)
SUB: 24 (2.03046%)
EQ: 111 (9.39086%)
GT: 0 (0%)
GTU: 0 (0%)
FU: lsu { ldw, ldq, ldh, stw, stq, sth, ldqu,... }
FU: alu_comp { add, sub, eq, gt, gtu }
FU: logic { and, ior, xor }
FU: ext1 { sxhw }
FU: ext2 { sxqw }
0

Intro and Exploration

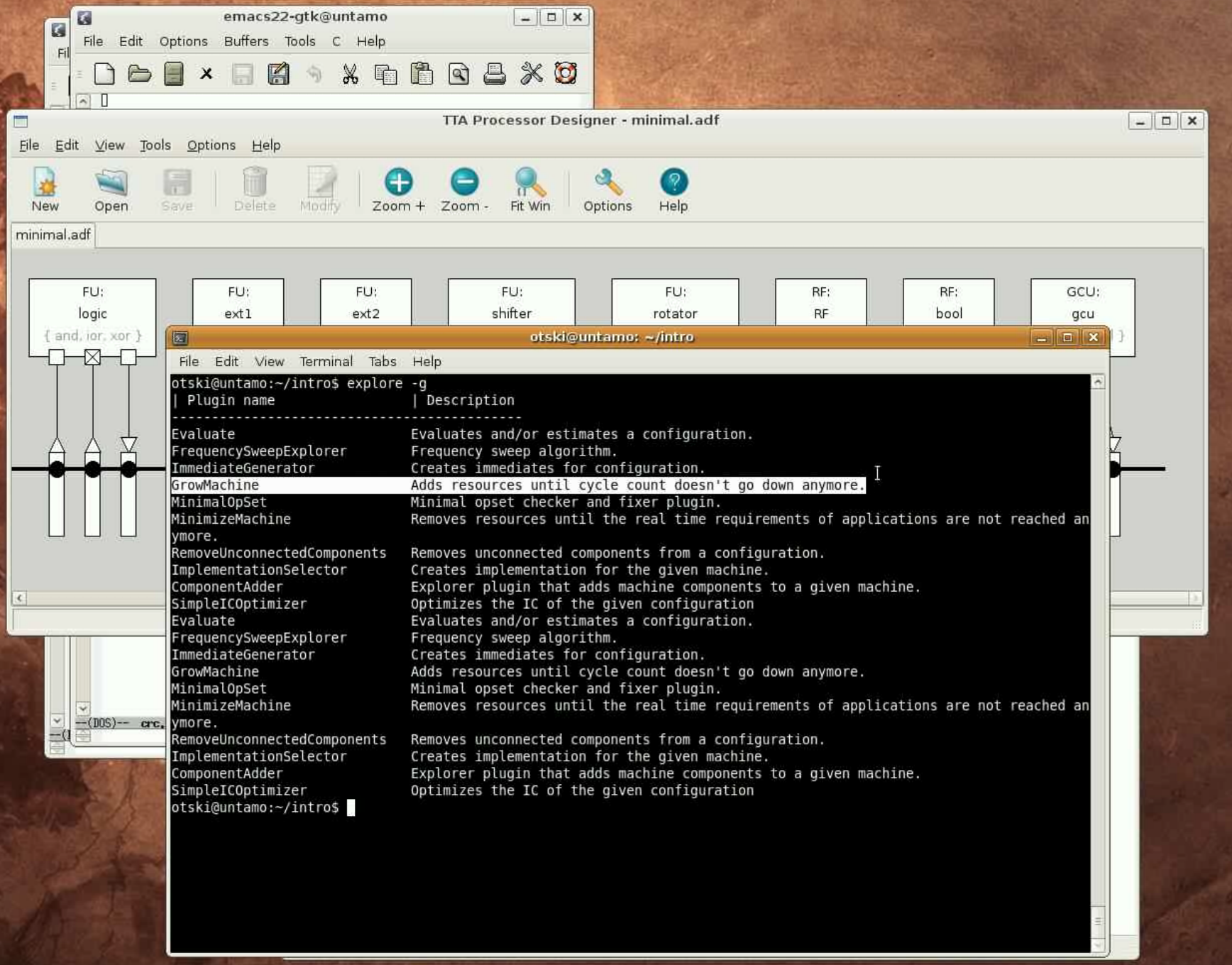
- The minimal.adf has only 5 registers, the CRC algorithm can use more as we saw from the stats
- Let's add some more registers using the Processor Designer (ProDe)
  - In this case we double the number to 10 registers
- Recompile the program for the new architecture with 10 registers using tcecc
- This time we'll load the processor+program to the command line interface of the simulator (ttasim)
  - “info proc cycles” works here also and produces the cycle count **3116** which is almost halved from the one we got using a machine with only 5 registers


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom +
minimal.adf
FU: logic { and, ior, xor }
FU: ext1 { sxhw }
FU: ext2 { sxqw }
RF: RF 5x32
RF: bool 2x1
GCU: gcu { jump, call }
Register File: name = RF, type = normal, size = 5, width = 32
Register File
Name: RF
Type: normal
Size: 5
Width: 32
Max-Reads: 1
Max-Writes: 1
Guard Latency: 0
Ports:
Name
wr
rd
Add... Edit... Delete
Help OK Cancel


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
minimal.adf
FU: logic { and, ior, xor }
FU: ext1
FU: ext2
FU: shifter
FU: rotator
RF: RF
RF: bool
GCU: gcu
otski@untamo: ~/intro
File Edit View Terminal Tabs Help
otski@untamo:~/intro$ prode minimal.adf &
[1] 8294
otski@untamo:~/intro$ tcecc -O2 -a minimal.adf -o crc.tpef crc.c main.c
otski@untamo:~/intro$ proxim minimal.adf crc.tpef &
[2] 8516
otski@untamo:~/intro$ tcecc -O2 -a minimal.adf -o crc.tpef crc.c main.c
[2]+ Done proxim minimal.adf crc.tpef
otski@untamo:~/intro$ ttasim -a minimal.adf -p crc.tpef
(ttasim) info proc cycles
3116
(ttasim)

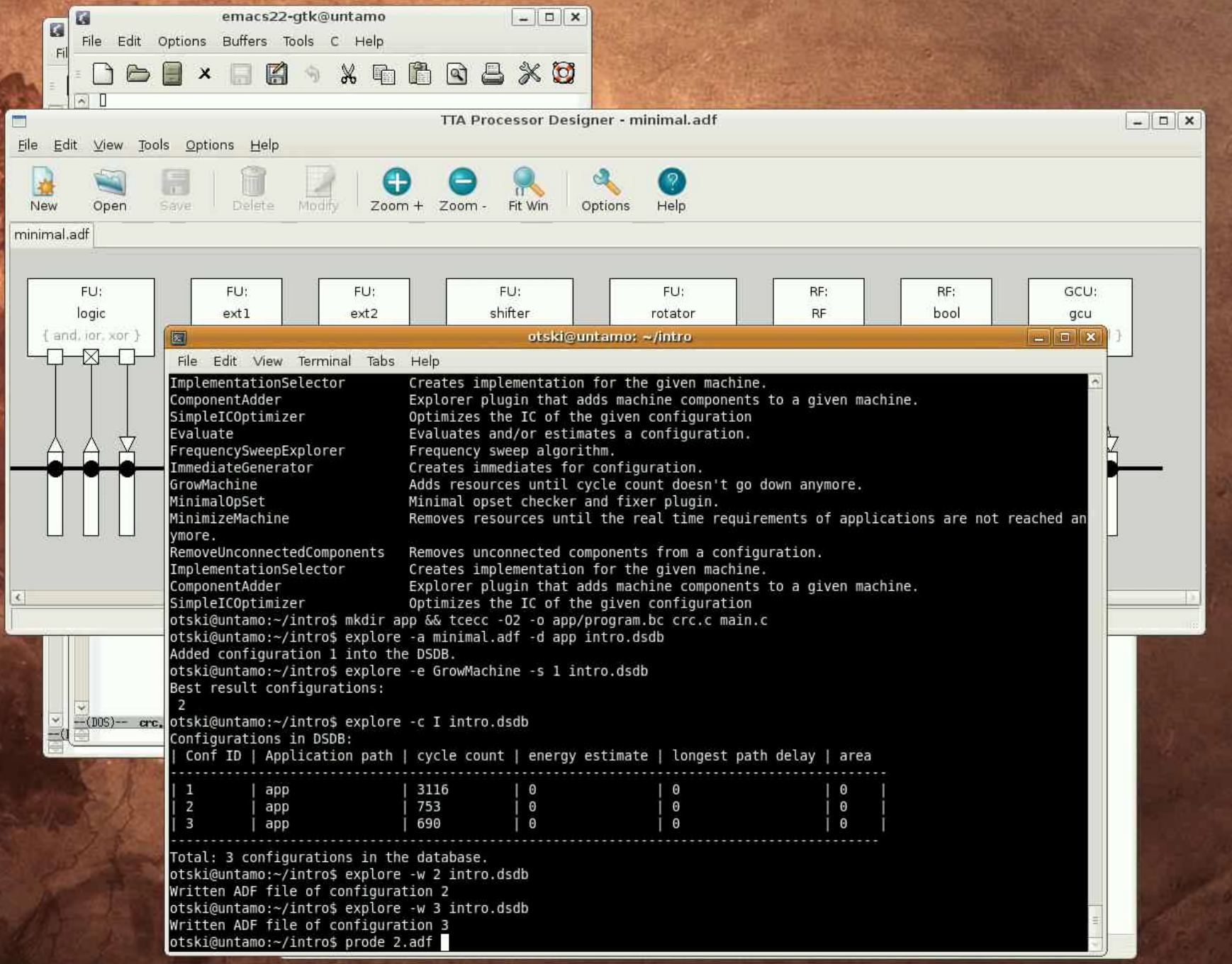
Intro and Exploration

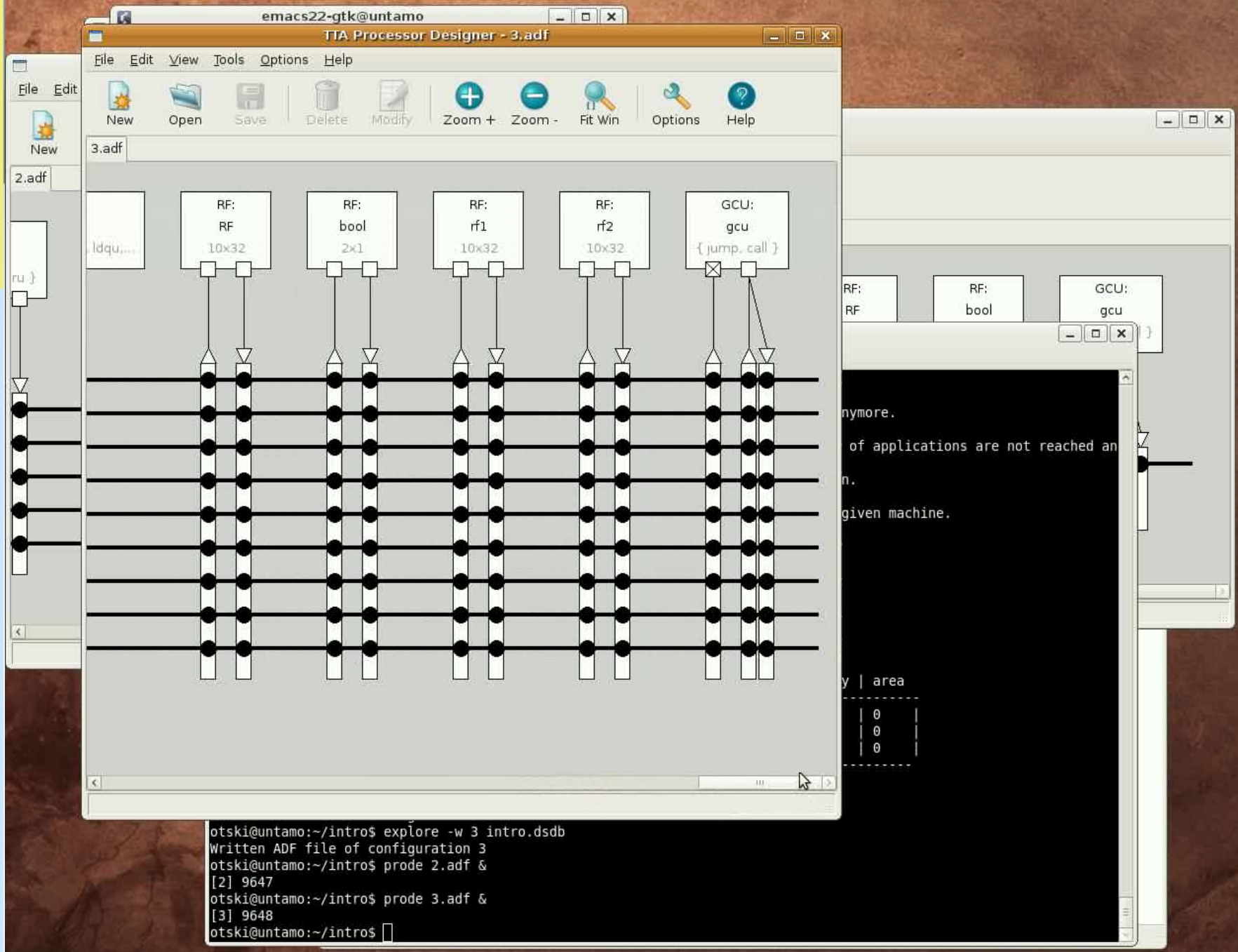
- Next we'll try the Design Space Explorer tool
- The tool is used to launch “explorer plugins” which perform modifications to the target and measure their effect to
  - cycle count
  - area estimate
  - energy estimate
  - longest path delay estimate
- The plugins can be fully automated or semi-automated
  - Can implement a loop that explores multiple points in the design space or just generate one new design space point (processor configuration)
- “explore -g” prints a list of available exploration plugins
  - In this example we use the GrowMachine plugin which adds basic resources to the machine until the cycle count does not drop anymore significantly


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
minimal.adf
FU: logic { and, ior, xor }
FU: ext1
FU: ext2
FU: shifter
FU: rotator
RF: RF
RF: bool
GCU: gcu
otski@untamo: ~/intro
File Edit View Terminal Tabs Help
otski@untamo:~/intro$ explore -g
| Plugin name | Description
Evaluate Evaluates and/or estimates a configuration.
FrequencySweepExplorer Frequency sweep algorithm.
ImmediateGenerator Creates immediates for configuration.
GrowMachine Adds resources until cycle count doesn't go down anymore.
MinimalOpSet Minimal opset checker and fixer plugin.
MinimizeMachine Removes resources until the real time requirements of applications are not reached an
ymore.
RemoveUnconnectedComponents Removes unconnected components from a configuration.
ImplementationSelector Creates implementation for the given machine.
ComponentAdder Explorer plugin that adds machine components to a given machine.
SimpleICOptimizer Optimizes the IC of the given configuration
Evaluate Evaluates and/or estimates a configuration.
FrequencySweepExplorer Frequency sweep algorithm.
ImmediateGenerator Creates immediates for configuration.
GrowMachine Adds resources until cycle count doesn't go down anymore.
MinimalOpSet Minimal opset checker and fixer plugin.
MinimizeMachine Removes resources until the real time requirements of applications are not reached an
ymore.
RemoveUnconnectedComponents Removes unconnected components from a configuration.
ImplementationSelector Creates implementation for the given machine.
ComponentAdder Explorer plugin that adds machine components to a given machine.
SimpleICOptimizer Optimizes the IC of the given configuration
otski@untamo:~/intro$

Intro and Exploration

- The explorer is used by first adding the software of the application to a Design Space Database (dsdb)
- Then we launch the explorer plugin which produces one or more new “configurations” to the DSDB along with their characteristics data (at least cycle counts)
  - Plugins usually have parameters which can also be configured through explorer
- In this case 2 new configurations were produced after starting the GrowMachine from our minimal.adf starting point
- The best cycle count we got using this explorer plugin is **690**
- Let's see with ProDe how the generated best architecture looks like


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
minimal.adf
FU: logic { and, ior, xor }
FU: ext1
FU: ext2
FU: shifter
FU: rotator
RF: RF
RF: bool
GCU: gcu
otski@untamo: ~/intro
File Edit View Terminal Tabs Help
ImplementationSelector Creates implementation for the given machine.
ComponentAdder Explorer plugin that adds machine components to a given machine.
SimpleICOptimizer Optimizes the IC of the given configuration
Evaluate Evaluates and/or estimates a configuration.
FrequencySweepExplorer Frequency sweep algorithm.
ImmediateGenerator Creates immediates for configuration.
GrowMachine Adds resources until cycle count doesn't go down anymore.
MinimalOpSet Minimal opset checker and fixer plugin.
MinimizeMachine Removes resources until the real time requirements of applications are not reached an
ymore.
RemoveUnconnectedComponents Removes unconnected components from a configuration.
ImplementationSelector Creates implementation for the given machine.
ComponentAdder Explorer plugin that adds machine components to a given machine.
SimpleICOptimizer Optimizes the IC of the given configuration
otski@untamo:~/intro$ mkdir app && tcecc -O2 -o app/program.bc crc.c main.c
otski@untamo:~/intro$ explore -a minimal.adf -d app intro.dsdb
Added configuration 1 into the DSDB.
otski@untamo:~/intro$ explore -e GrowMachine -s 1 intro.dsdb
Best result configurations:
2
otski@untamo:~/intro$ explore -c I intro.dsdb
Configurations in DSDB:
| Conf ID | Application path | cycle count | energy estimate | longest path delay | area
1	app	3116	0	0	0
2	app	753	0	0	0
3	app	690	0	0	0
Total: 3 configurations in the database.
otski@untamo:~/intro$ explore -w 2 intro.dsdb
Written ADF file of configuration 2
otski@untamo:~/intro$ explore -w 3 intro.dsdb
Written ADF file of configuration 3
otski@untamo:~/intro$ prode 2.adf


emacs22-gtk@untamo
TTA Processor Designer - 3.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
3.adf
RF: RF 10x32
RF: bool 2x1
RF: rf1 10x32
RF: rf2 10x32
GCU: gcu { jump, call }
nymore.
of applications are not reached an
n.
given machine.
y | area
otski@untamo:~/intro$ explore -w 3 intro.dsdb
Written ADF file of configuration 3
otski@untamo:~/intro$ prode 2.adf &
[2] 9647
otski@untamo:~/intro$ prode 3.adf &
[3] 9648
otski@untamo:~/intro$

Intro and Exploration

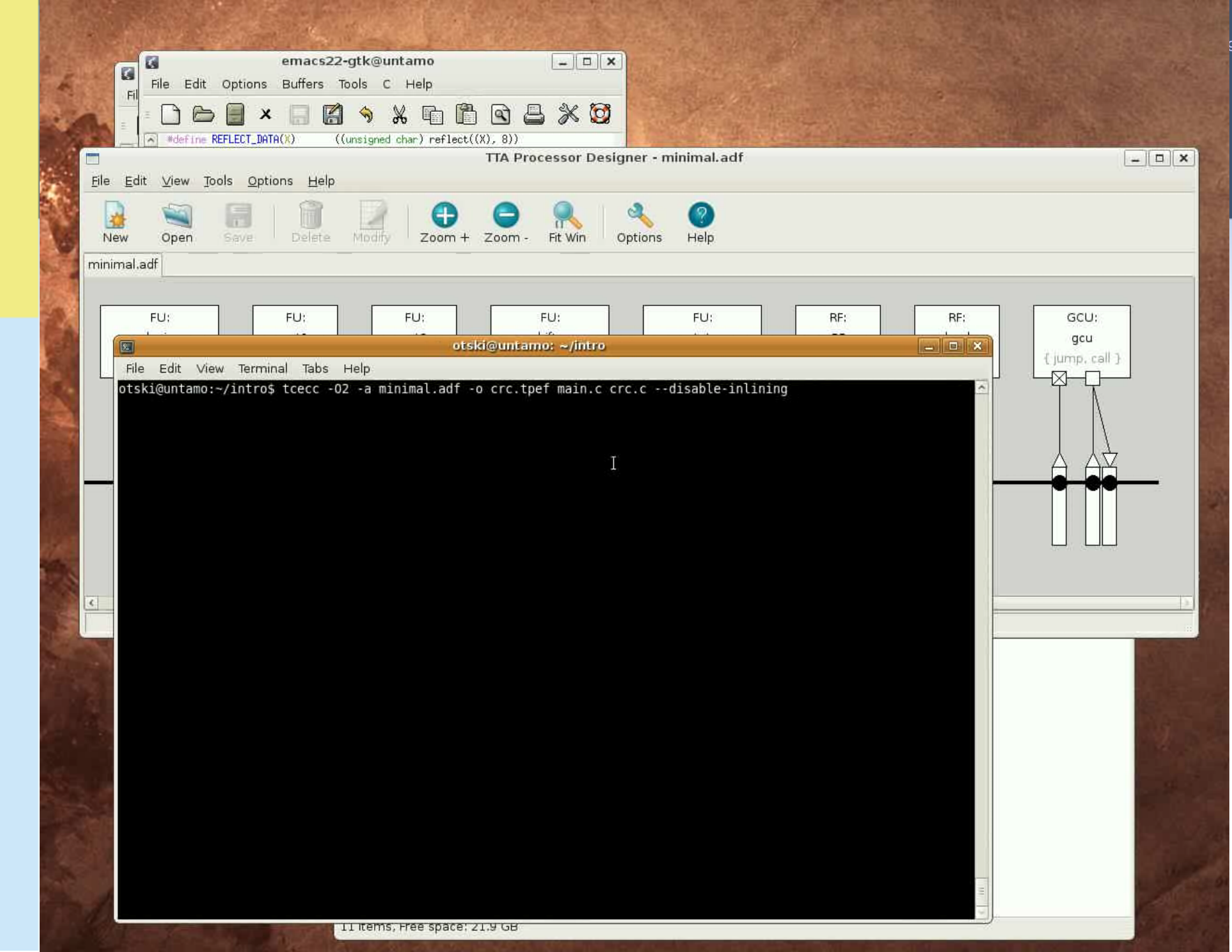
- As we can see, the GrowMachine plugin has added more buses and FUs to the machine
  - Currently a brute-force approach of incrementing the current resource set with a constant factor is used
- For example, the machine has 9 buses (instead of 1), many more function units and additional two register files

Profiling and Using a Custom Operation (about 4 minutes)

- The GrowMachine plugin managed to squeeze the cycle count down to **690** by just duplicating resources
- We are not happy with this number yet as we know it can get much lower when some custom hardware is used
- This video shows how to profile the application and use a custom operation (special function unit) to accelerate a “hot spot” in the CRC program

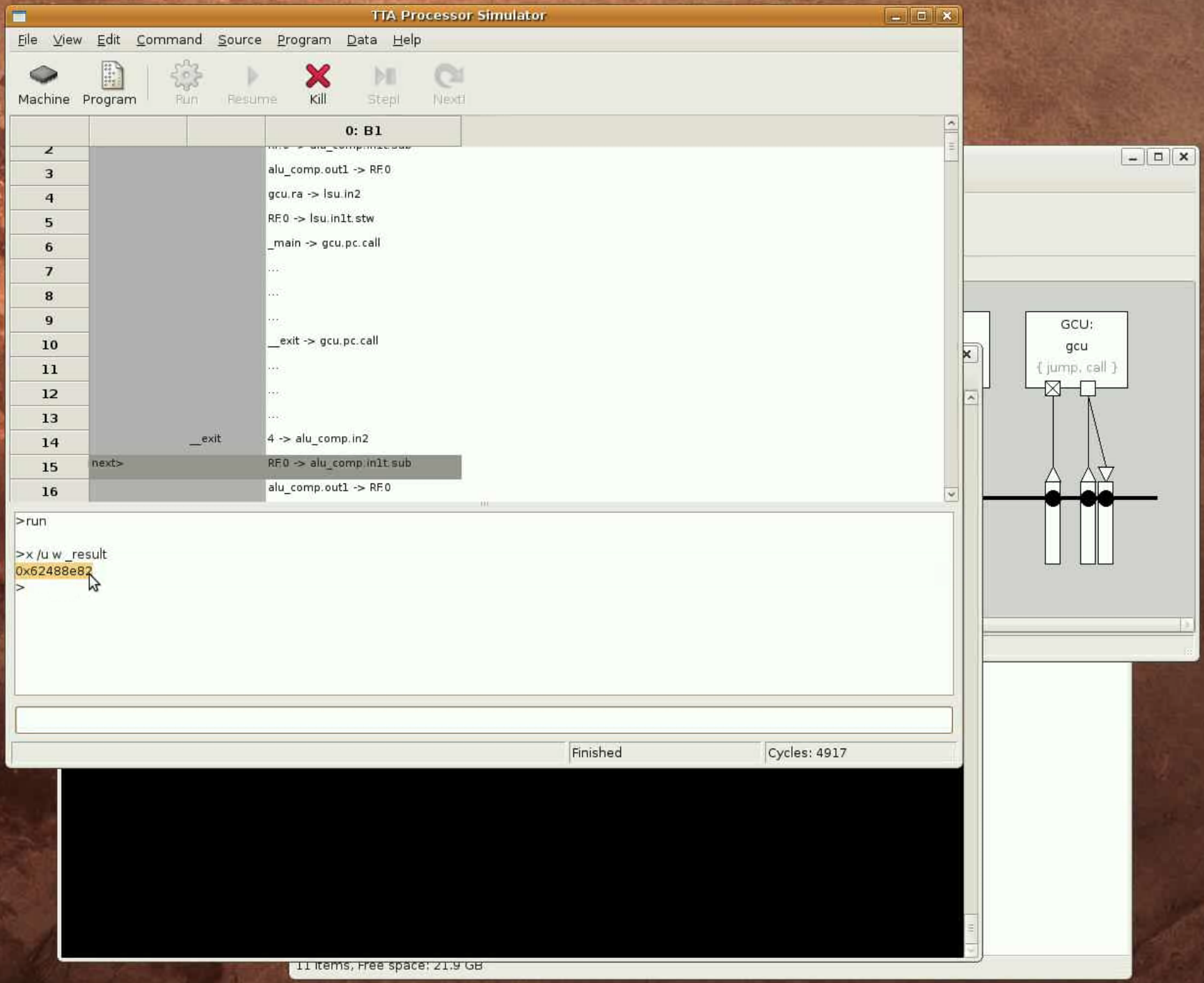
Profiling and Using a Custom Operation

- First we'll compile the program with procedure inlining disabled so we get a proper function profile of the program


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
#define REFLECT_DATA(X) ((unsigned char) reflect((X), 8))
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
minimal.adf
FU:
FU:
FU:
FU:
FU:
RF:
RF:
GCU:
gcu
{ jump, call }
otski@untamo: ~/intro
File Edit View Terminal Tabs Help
otski@untamo:~/intro$ tcecc -O2 -a minimal.adf -o crc.tpef main.c crc.c --disable-inlining
11 items, Free space: 21.9 GB

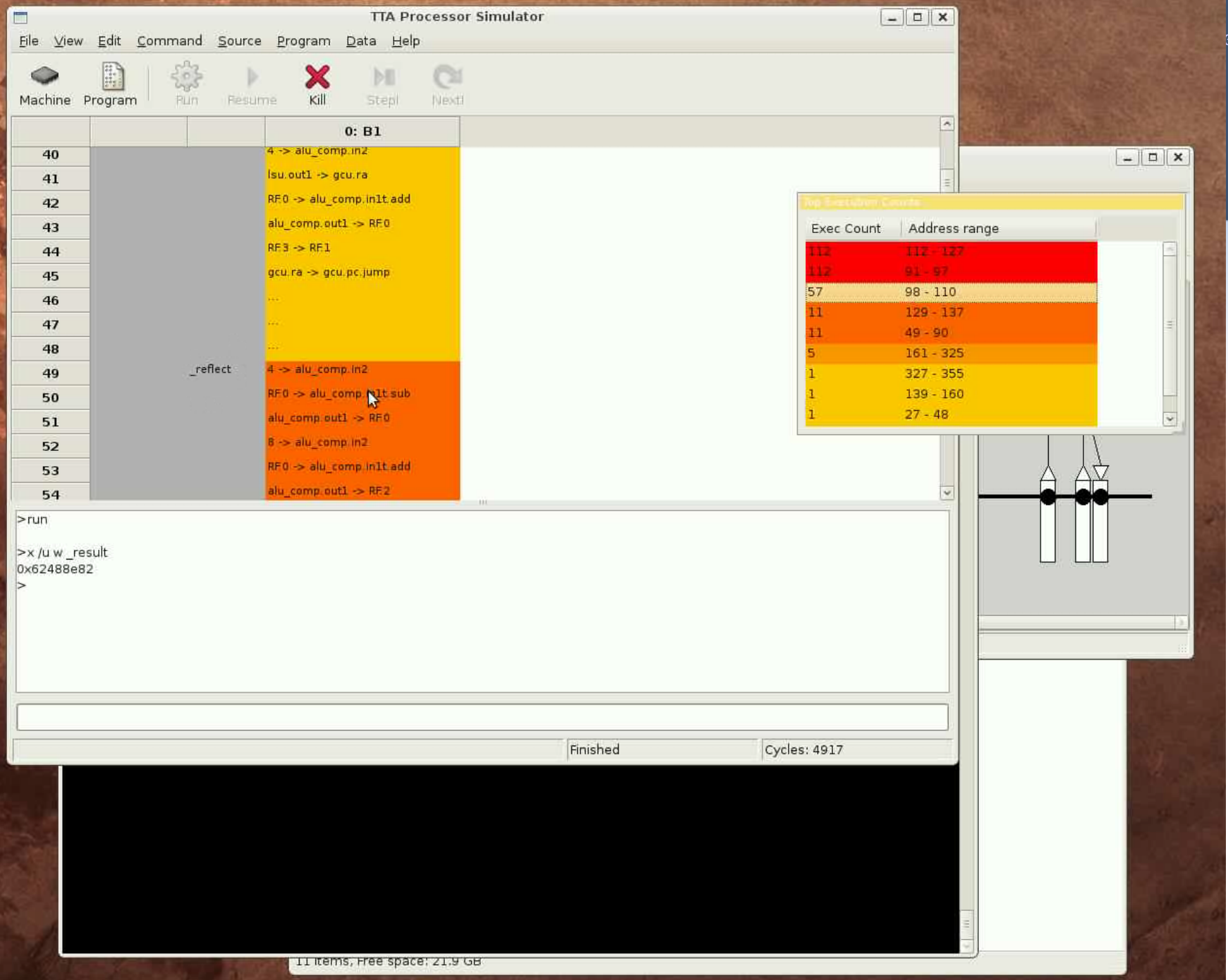
Profiling and Using a Custom Operation

- Simulate the program:
  - Note that the cycle count has increased due to the disabled inlining to **4917**
  - Verify the program by dumping the computed CRC number from memory


TTA Processor Simulator
File View Edit Command Source Program Data Help
Machine Program Run Resume Kill Step! Next!
0: B1
2
3 alu_comp.out1 -> RF.0
4 gcu.ra -> lsu.in2
5 RF.0 -> lsu.in1t.stw
6 _main -> gcu.pc.call
7 ...
8 ...
9 ...
10 __exit -> gcu.pc.call
11 ...
12 ...
13 ...
14 __exit 4 -> alu_comp.in2
15 next> RF.0 -> alu_comp.in1t.sub
16 alu_comp.out1 -> RF.0
>run
>x /u w _result
0x62488e82
>
Finished
Cycles: 4917
GCU:
gcu
{ jump, call }
11 items, Free space: 21.9 GB

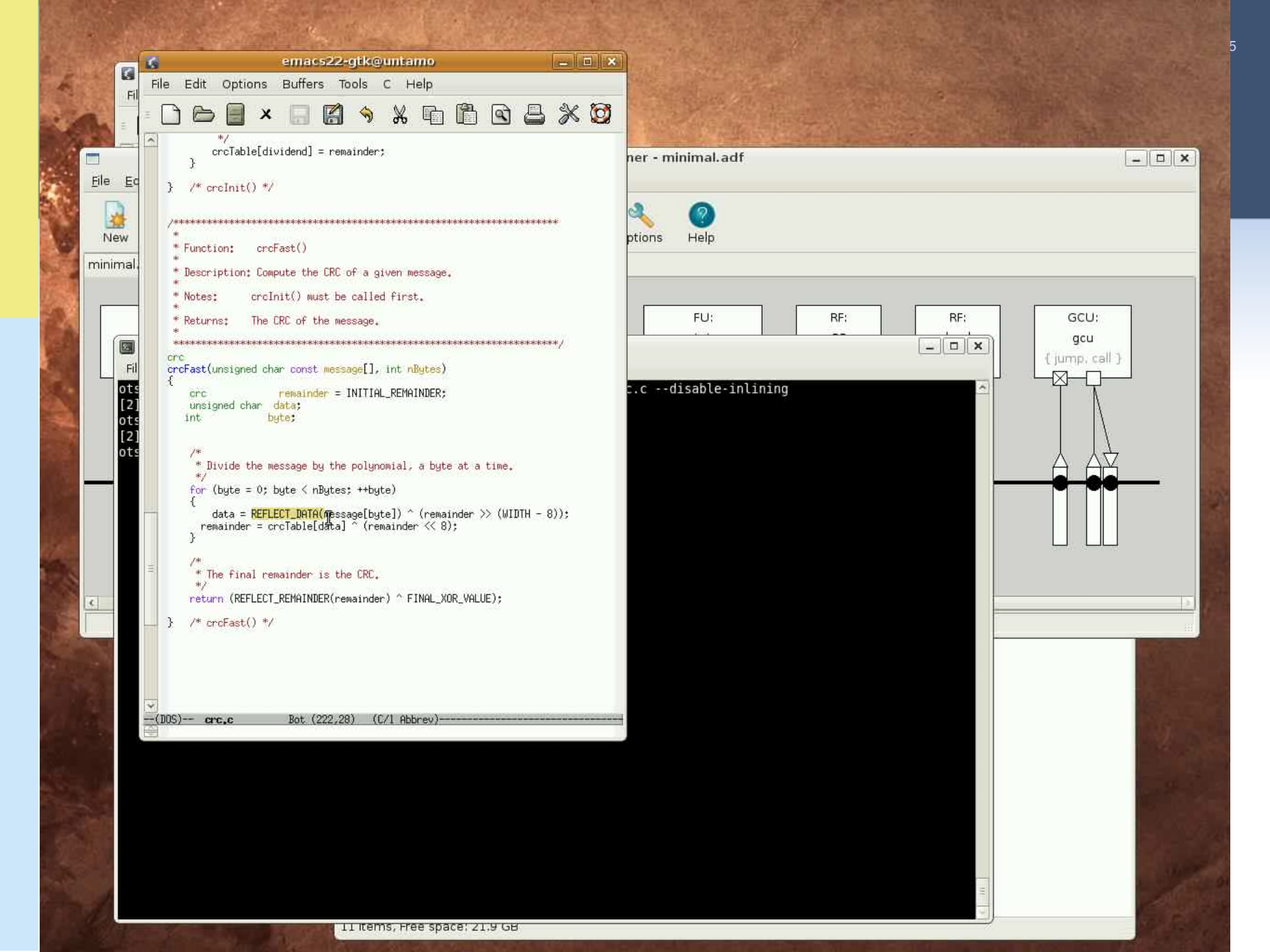
Profiling and Using a Custom Operation

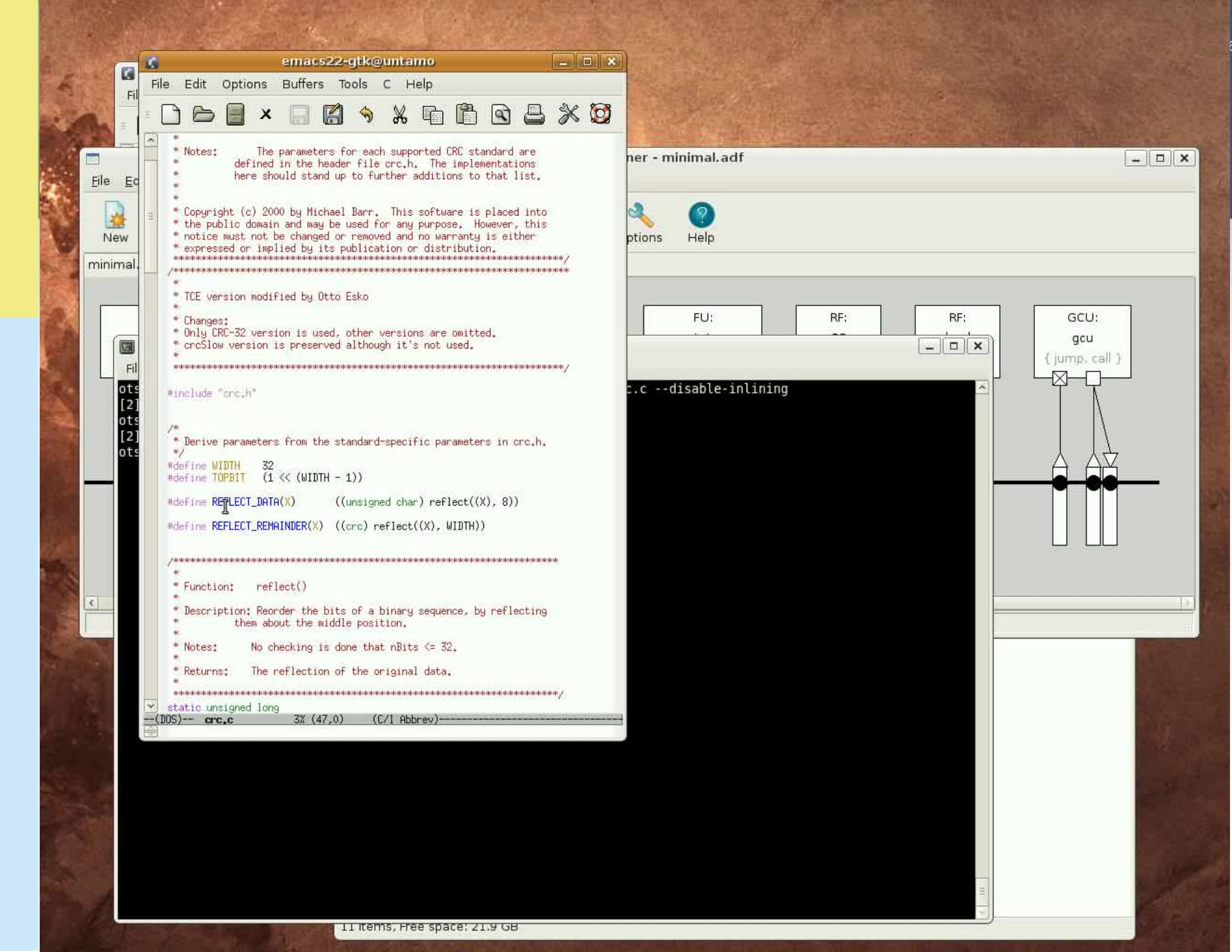
- Program profile:
  - To find out the “hot spot” in the program, we highlight the top executed instructions
  - We find out the instructions in the _reflect() function are executed very frequently, thus it's a potential candidate for acceleration with a custom operation (special function unit)


TTA Processor Simulator
File View Edit Command Source Program Data Help
Machine Program Run Resume Kill Step! Next!
0: B1
40 4 -> alu_comp.in2
41 lsu.out1 -> gcu.ra
42 RF:0 -> alu_comp.in1t.add
43 alu_comp.out1 -> RF:0
44 RF:3 -> RF:1
45 gcu.ra -> gcu.pc.jump
46 ...
47 ...
48 ...
49 _reflect 4 -> alu_comp.in2
50 RF:0 -> alu_comp.in1t.sub
51 alu_comp.out1 -> RF:0
52 8 -> alu_comp.in2
53 RF:0 -> alu_comp.in1t.add
54 alu_comp.out1 -> RF:2
Exec Count Address range
112 112 - 127
112 91 - 97
57 98 - 110
11 129 - 137
11 49 - 90
5 161 - 325
1 327 - 355
1 139 - 160
1 27 - 48
>run
>x /u w _result
0x62488e82
>
Finished
Cycles: 4917
11 items, Free space: 21.9 GB

Profiling and Using a Custom Operation

- We find out that the reflect function is called through macros REFLECT_DATA and REFLECT_REMAINDER in the core loop of the C code
- The reflect() computes a “bit reflection”
  - Reverses bits like a mirror was placed in the middle of the word)
  - We see from the macros that it's done only for word sizes 8 and 32 bits


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
*/
crcTable[dividend] = remainder;
}
} /* crcInit() */
/**
*
* Function: crcFast()
*
* Description: Compute the CRC of a given message.
*
* Notes: crcInit() must be called first.
*
* Returns: The CRC of the message.
*
**/
crc
crcFast(unsigned char const message[], int nBytes)
{
crc remainder = INITIAL_REMAINDER;
unsigned char data;
int byte;
/*
* Divide the message by the polynomial, a byte at a time.
*/
for (byte = 0; byte < nBytes; ++byte)
{
data = REFLECT_DATA(message[byte]) ^ (remainder >> (WIDTH - 8));
remainder = crcTable[data] ^ (remainder << 8);
}
/*
* The final remainder is the CRC.
*/
return (REFLECT_REMAINDER(remainder) ^ FINAL_XOR_VALUE);
} /* crcFast() */
--(DOS)-- crc.c Bot (222,28) (C/l Abbrev)
ner - minimal.adf
ptions
Help
FU:
RF:
RF:
GCU:
gcu
{ jump, call }
c.c --disable-inlining
11 items, Free space: 21.9 GB


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
* Notes: The parameters for each supported CRC standard are
* defined in the header file crc.h. The implementations
* here should stand up to further additions to that list.
* Copyright (c) 2000 by Michael Barr. This software is placed into
* the public domain and may be used for any purpose. However, this
* notice must not be changed or removed and no warranty is either
* expressed or implied by its publication or distribution.
* TCE version modified by Otto Esko
* Changes:
* Only CRC-32 version is used, other versions are omitted.
* crcSlow version is preserved although it's not used.
#include "crc.h"
* Derive parameters from the standard-specific parameters in crc.h.
#define WIDTH 32
#define TOPBIT (1 << (WIDTH - 1))
#define REFLECT_DATA(X) ((unsigned char) reflect((X), 8))
#define REFLECT_REMAINDER(X) ((crc) reflect((X), WIDTH))
* Function: reflect()
* Description: Reorder the bits of a binary sequence, by reflecting
* them about the middle position.
* Notes: No checking is done that nBits <= 32.
* Returns: The reflection of the original data.
static unsigned long
--(DOS)-- crc.c 3% (47,0) (C/l Abbrev)
ner - minimal.adf
ptions
Help
FU:
RF:
RF:
GCU:
gcu
{ jump, call }
.c --disable-inlining
11 items, Free space: 21.9 GB

emacs22-gtk@untamo

File Edit Options Buffers Tools C Help

```
#define REFLECT_DATA(X)         ((unsigned char) reflect((X), 8))

#define REFLECT_REMAINDER(X)    ((crc) reflect((X), WIDTH))


/***************************************************************************
 *
 * Function:    reflect()
 *
 * Description: Reorder the bits of a binary sequence, by reflecting
 *              them about the middle position.
 *
 * Notes:       No checking is done that nBits <= 32.
 *
 * Returns:     The reflection of the original data.
 *
 ***************************************************************************/
static unsigned long
reflect(unsigned long data, unsigned char nBits)
{
   unsigned long  reflection = 0x00000000;
   unsigned char  bit;

   /*
    * Reflect the data about the center bit.
    */
   for (bit = 0; bit < nBits; ++bit)
   {
      /*
       * If the LSB bit is set, set the reflection of it.
       */
      if (data & 0x01)
      {
         reflection |= (1 << ((nBits - 1) - bit));
      }

      data = (data >> 1);
   }

   return (reflection);

}  /* reflect() */


/***************************************************************************
 *
 * Function:    crcSlow()
 *
```

--(DOS)-- crc.c 21% (38,55) (C/l Abbrev)----------

ner - minimal.adf

ptions Help

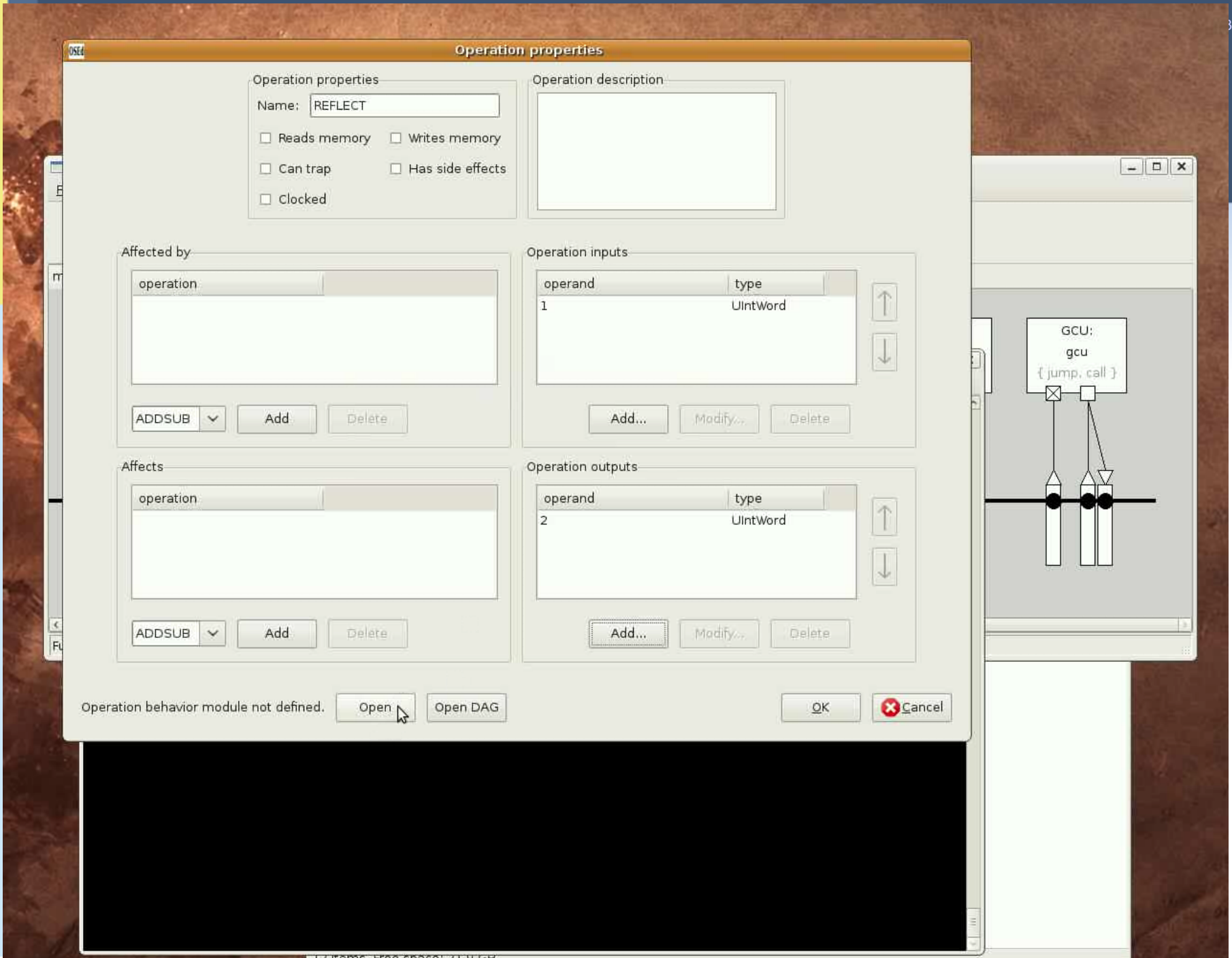
FU: RF: RF: GCU: gcu { jump, call }

c.c --disable-inlining

11 items, Free space: 21.9 GB

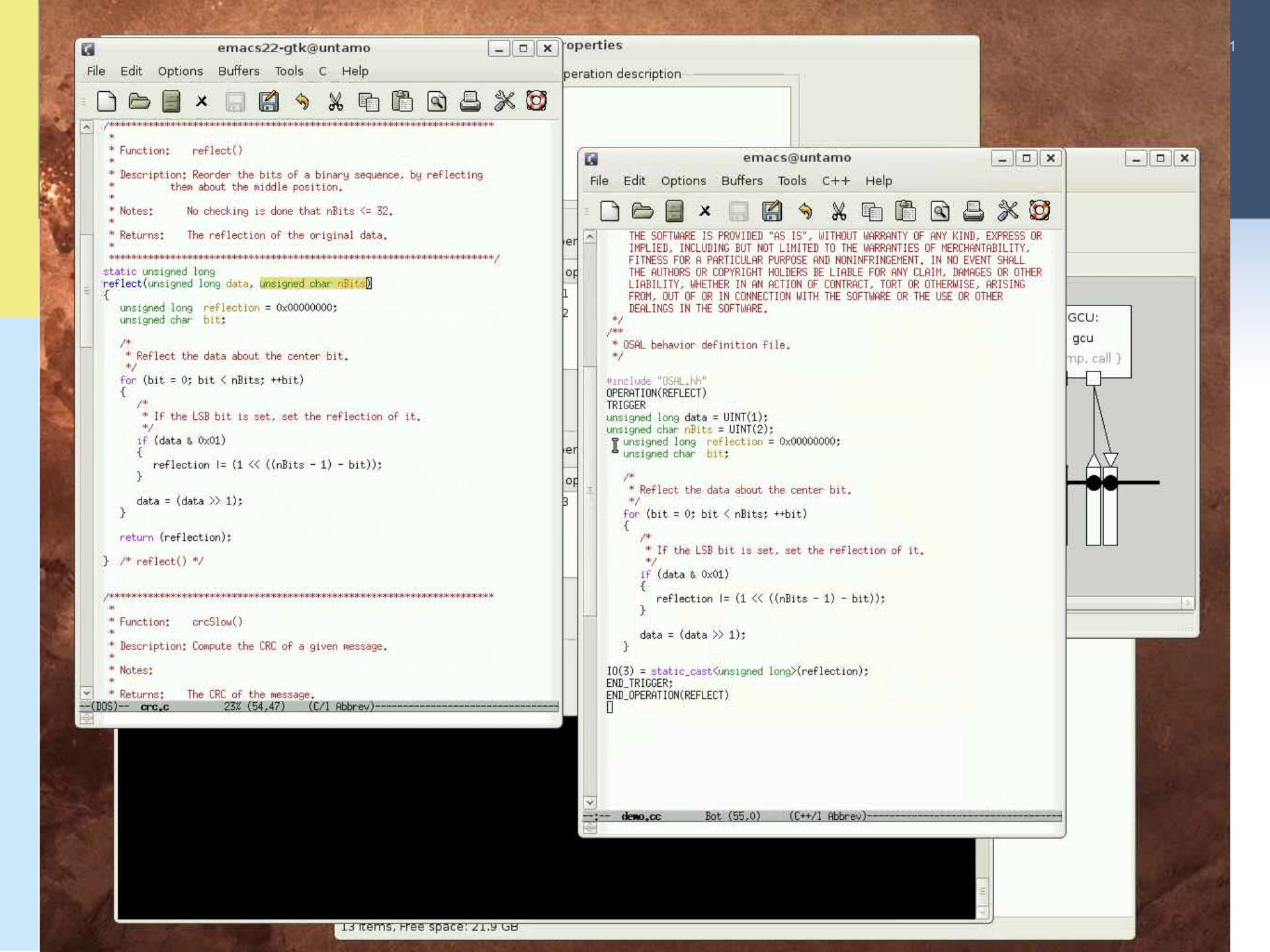
Profiling and Using a Custom Operation

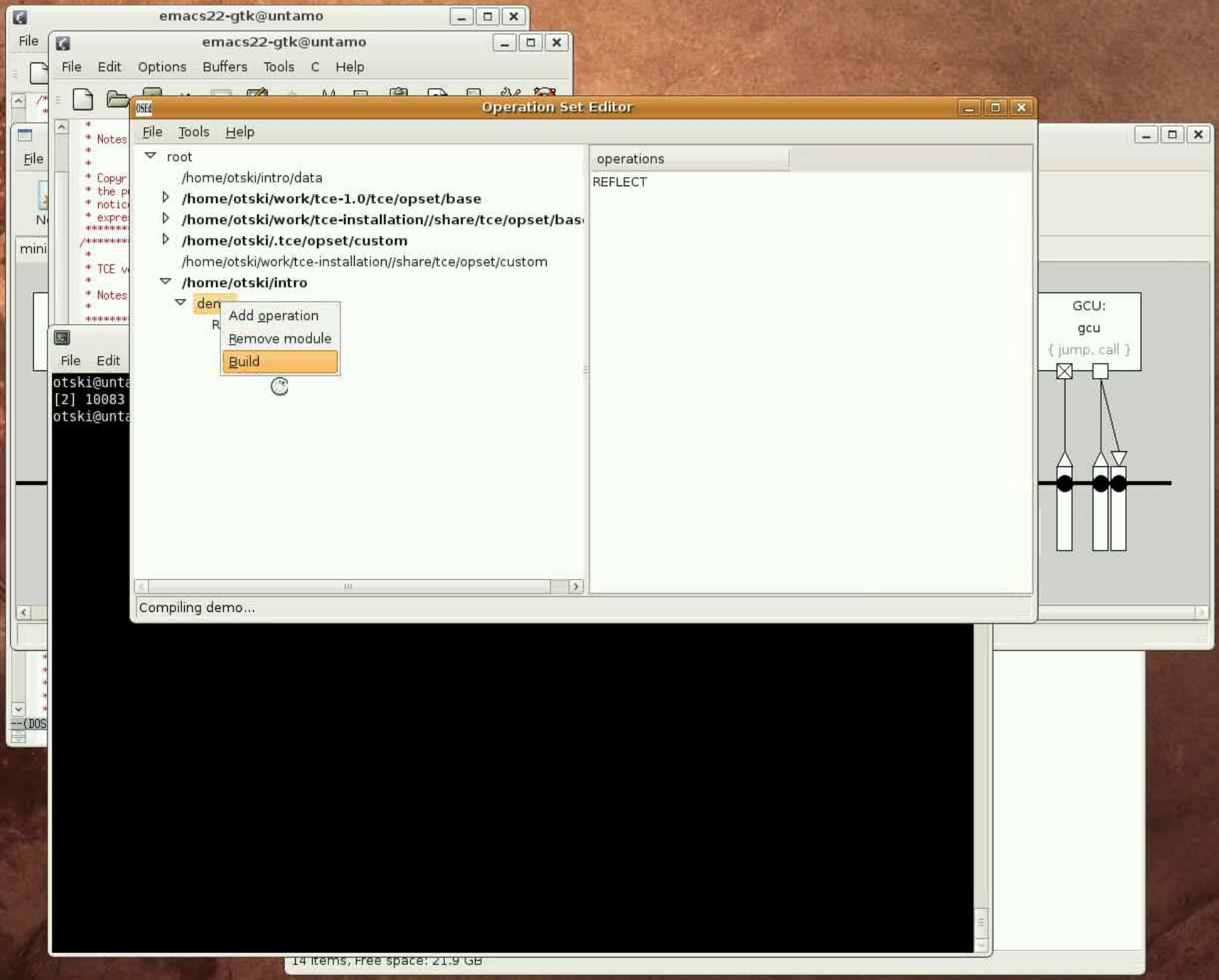
- The reflect() function is extremely simple and efficient to implement in hardware (just wiring and shifting if necessary), but looks like a heavy loop when implemented in C code
- Let's create a custom operation for the REFLECT
  - Custom operations added to TCE using a tool called Operation Set Editor (OSEd)
- First we add general “static” information about the operation like its name and the number and type of inputs and outputs

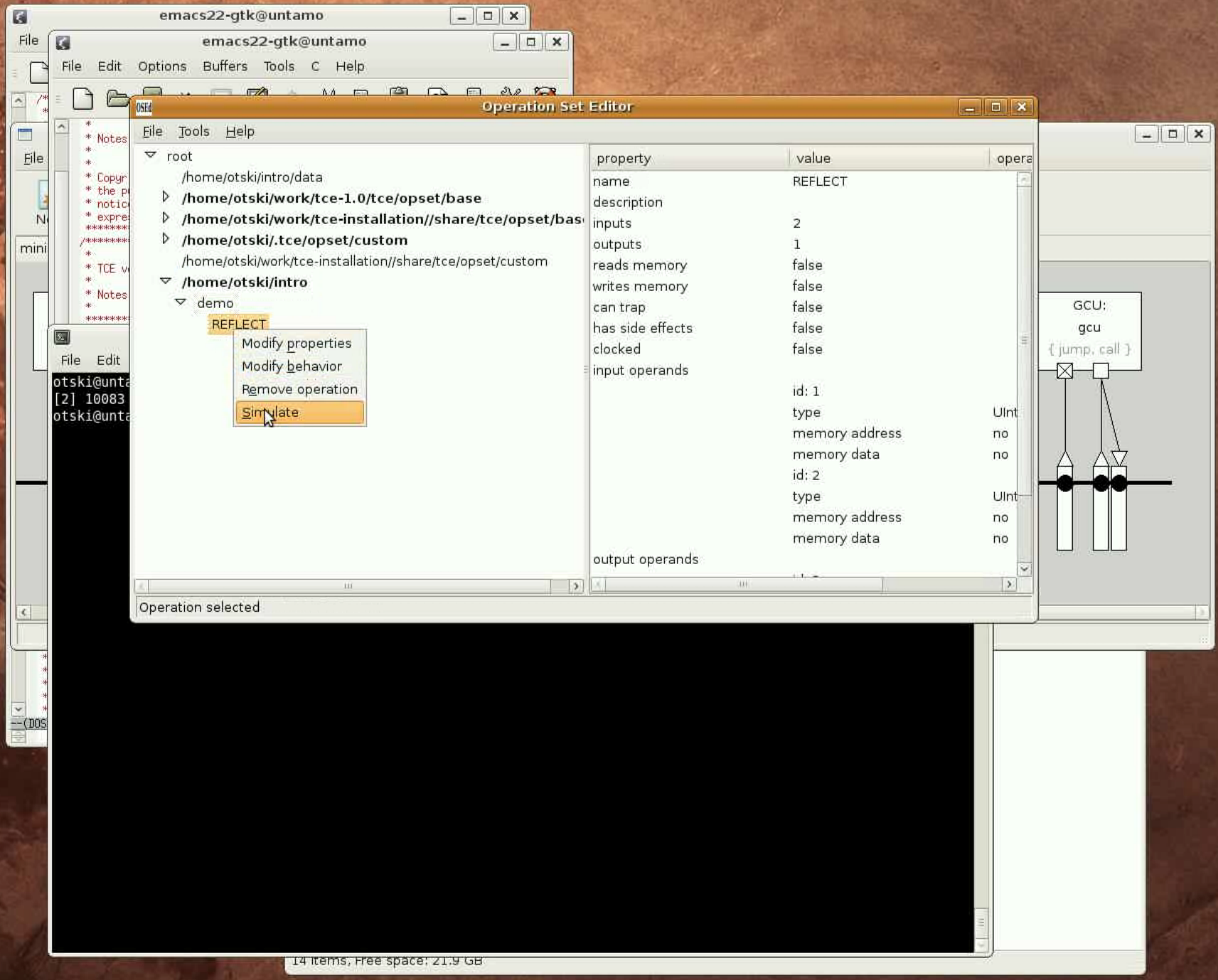

Operation properties
Operation properties
Name: REFLECT
Reads memory
Writes memory
Can trap
Has side effects
Clocked
Operation description
Affected by
operation
ADDSUB
Add
Delete
Operation inputs
operand
type
1
UIntWord
Add...
Modify...
Delete
Affects
operation
ADDSUB
Add
Delete
Operation outputs
operand
type
2
UIntWord
Add...
Modify...
Delete
Operation behavior module not defined.
Open
Open DAG
OK
Cancel
GCU:
gcu
{ jump, call }

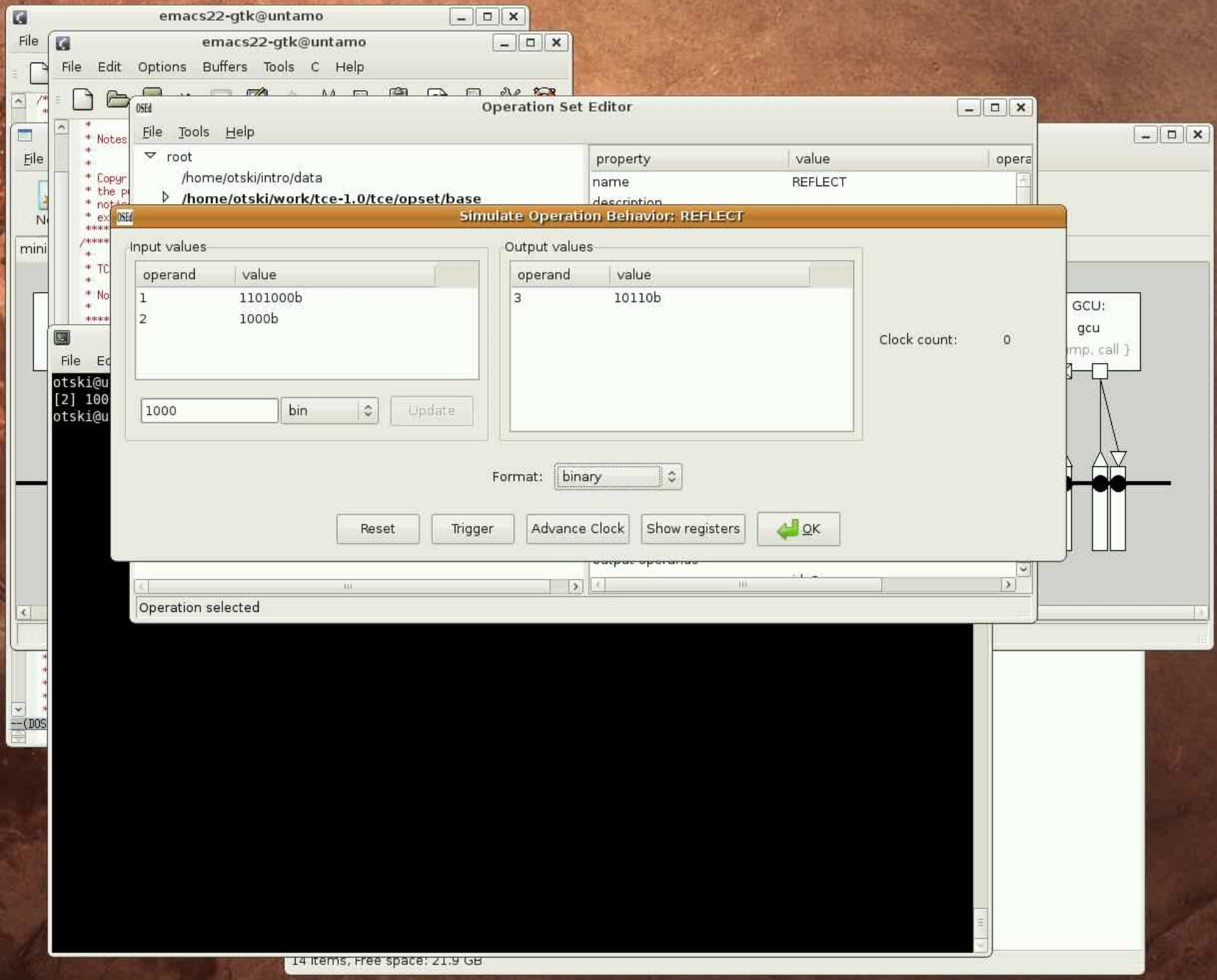
Profiling and Using a Custom Operation

- The let's add a simulation behavior description for the operation
- We can copy the original C code to the simulation behavior definition, just define:
  - Reads from operation inputs (UINT(1), UINT(2)) to variables in the C code
  - Write result to the operation output (IO(3))
- The simulation behavior is loaded runtime to the processor simulator
  - It's a “plugin” module which needs to be compiled
  - Build it with OSEd
  - Test that the simulation behavior definition works using the operation behavior simulator


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
/**********
* Function: reflect()
* Description: Reorder the bits of a binary sequence, by reflecting
* them about the middle position.
* Notes: No checking is done that nBits <= 32.
* Returns: The reflection of the original data.
**********/
static unsigned long
reflect(unsigned long data, unsigned char nBits)
{
unsigned long reflection = 0x00000000;
unsigned char bit;
/*
* Reflect the data about the center bit.
*/
for (bit = 0; bit < nBits; ++bit)
{
/*
* If the LSB bit is set, set the reflection of it.
*/
if (data & 0x01)
{
reflection |= (1 << ((nBits - 1) - bit));
}
data = (data >> 1);
}
return (reflection);
} /* reflect() */
/**********
* Function: crcSlow()
* Description: Compute the CRC of a given message.
* Notes:
* Returns: The CRC of the message.
--(DOS)-- crc.c 23% (54,47) (C/l Abbrev)
operties
peration description
emacs@untamo
File Edit Options Buffers Tools C++ Help
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
*/
/**
* OSAL behavior definition file.
*/
#include "OSAL.hh"
OPERATION(REFLECT)
TRIGGER
unsigned long data = UINT(1);
unsigned char nBits = UINT(2);
unsigned long reflection = 0x00000000;
unsigned char bit;
/*
* Reflect the data about the center bit.
*/
for (bit = 0; bit < nBits; ++bit)
{
/*
* If the LSB bit is set, set the reflection of it.
*/
if (data & 0x01)
{
reflection |= (1 << ((nBits - 1) - bit));
}
data = (data >> 1);
}
IO(3) = static_cast<unsigned long>(reflection);
END_TRIGGER;
END_OPERATION(REFLECT)
--:-- demo.cc Bot (55,0) (C++/l Abbrev)
GCU:
gcu
mp, call }
13 items, Free space: 21.9 GB

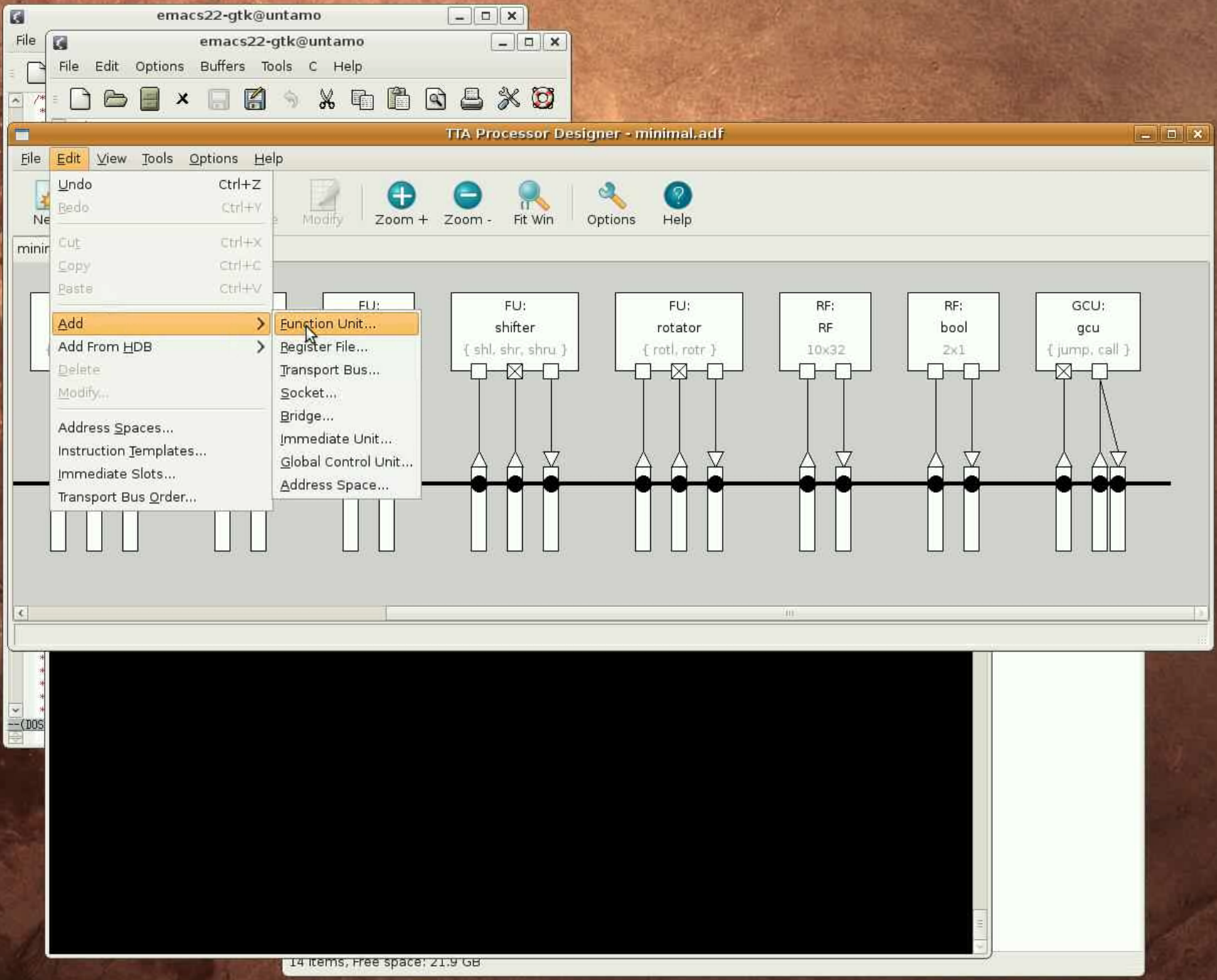

emacs22-gtk@untamo
File
emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
Operation Set Editor
File Tools Help
root
/home/otski/intro/data
/home/otski/work/tce-1.0/tce/opset/base
/home/otski/work/tce-installation//share/tce/opset/bas
/home/otski/.tce/opset/custom
/home/otski/work/tce-installation//share/tce/opset/custom
/home/otski/intro
Add operation
Remove module
Build
operations
REFLECT
Compiling demo...
GCU:
gcu
{ jump, call }
File Edit
otski@unta
[2] 10083
otski@unta
14 items, Free space: 21.9 GB

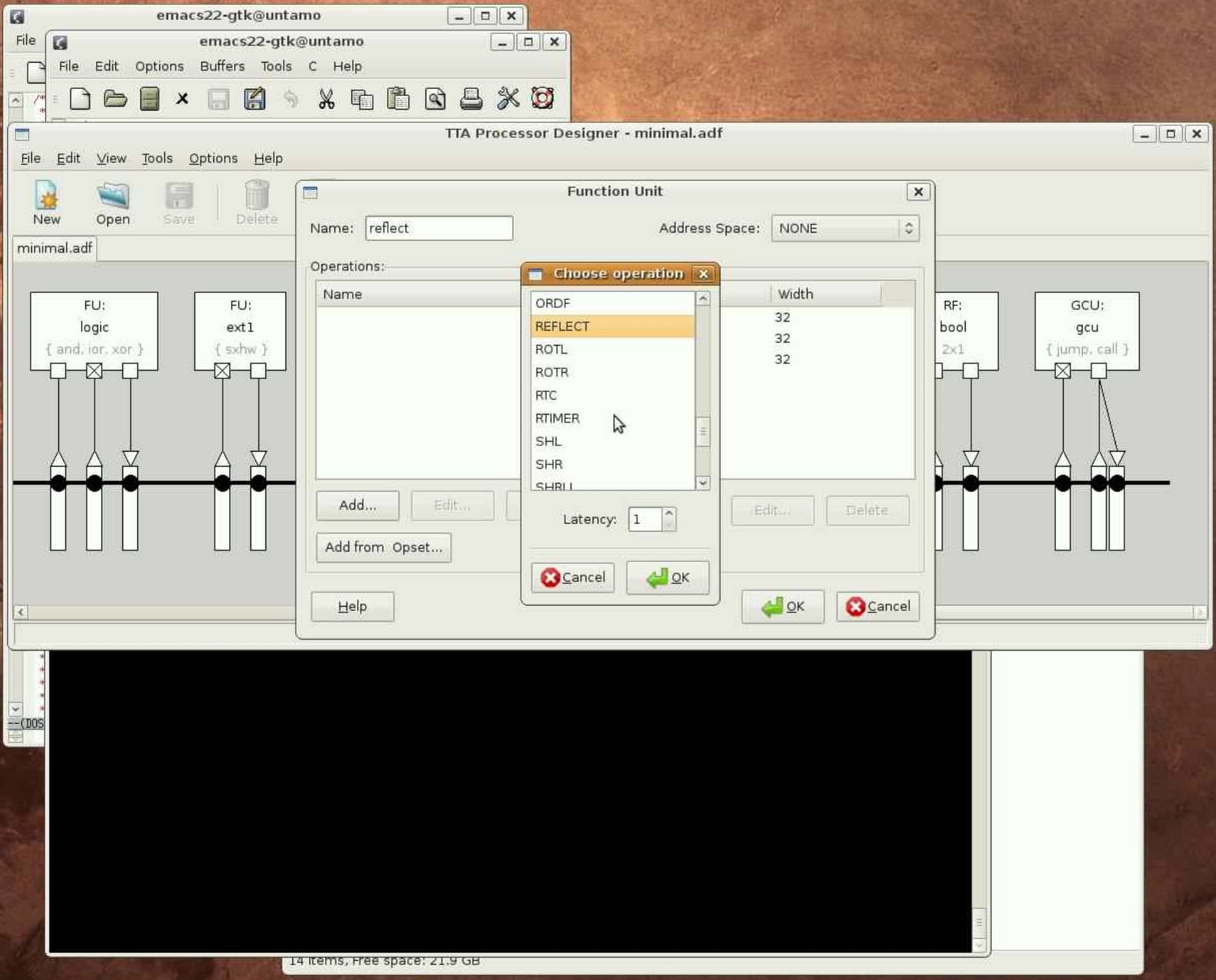

emacs22-gtk@untamo
File
emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
Operation Set Editor
File Tools Help
root
/home/otski/intro/data
/home/otski/work/tce-1.0/tce/opset/base
/home/otski/work/tce-installation//share/tce/opset/bas
/home/otski/.tce/opset/custom
/home/otski/work/tce-installation//share/tce/opset/custom
/home/otski/intro
demo
REFLECT
Modify properties
Modify behavior
Remove operation
Simulate
property
value
name
REFLECT
description
inputs
2
outputs
1
reads memory
false
writes memory
false
can trap
false
has side effects
false
clocked
false
input operands
id: 1
type
memory address
memory data
id: 2
type
memory address
memory data
output operands
Operation selected
GCU:
gcu
{ jump, call }
File Edit
otski@unta
[2] 10083
otski@unta
14 items, Free space: 21.9 GB

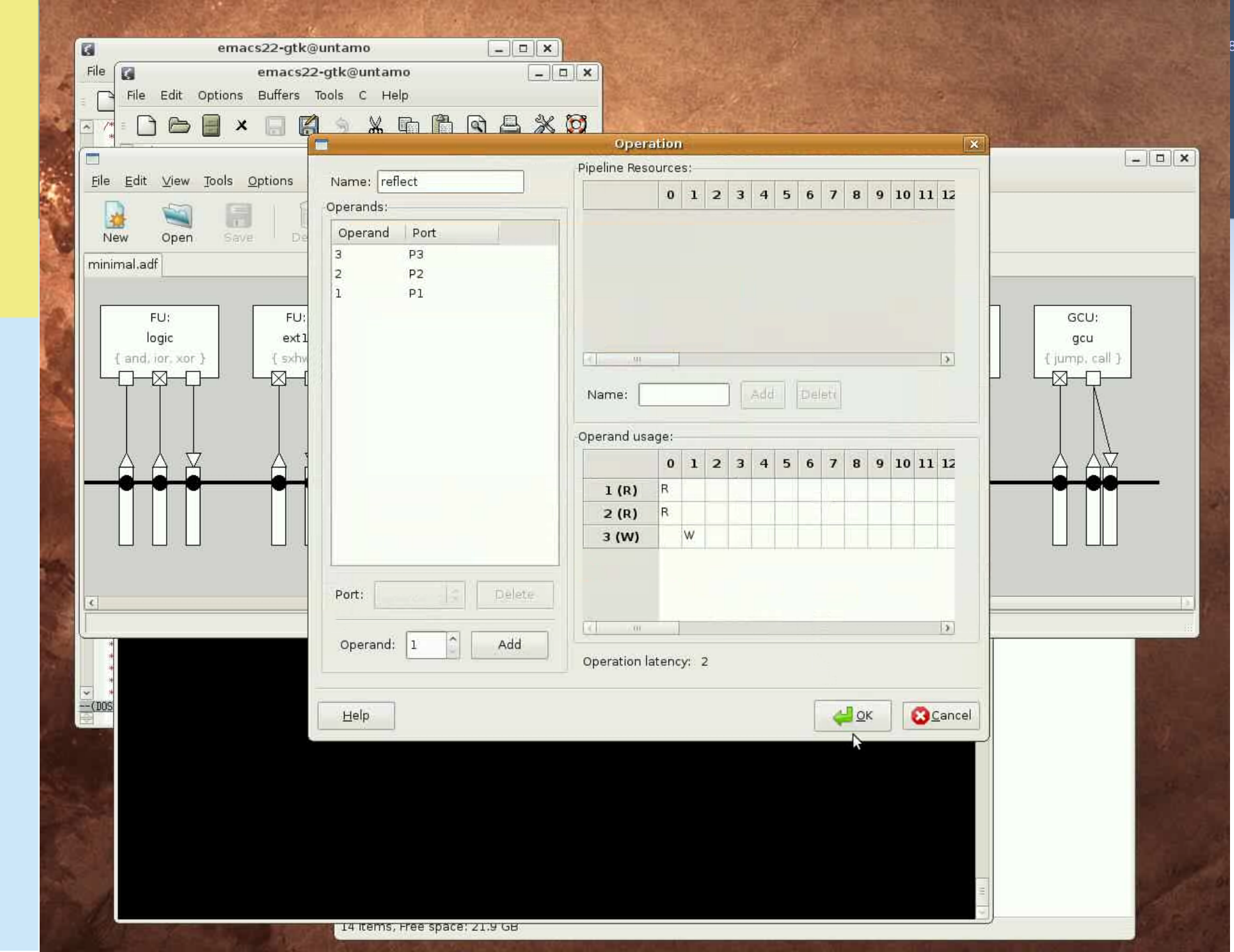

emacs22-gtk@untamo
File
emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
Operation Set Editor
File Tools Help
root
/home/otski/intro/data
/home/otski/work/tce-1.0/tce/opset/base
property
value
name
REFLECT
Simulate Operation Behavior: REFLECT
Input values
operand
value
1
1101000b
2
1000b
1000
bin
Update
Output values
operand
value
3
10110b
Clock count:
0
Format:
binary
Reset
Trigger
Advance Clock
Show registers
OK
Operation selected
GCU:
gcu
14 items, Free space: 21.9 GB

Adding SFU to the Machine and Using it in C Code (1.5 minutes)

- Now that we have defined a new custom operation to the TCE, we can use it in our TTA in a special function unit and execute it from our C code
- Add the custom operation to a new function unit in the TTA with the Processor Designer tool
  - Add a function unit
  - Add ports to the function unit
  - Add the operation to the function unit
  - Edit the operations port bindings, pipeline resource usage, and latency
- In this case we are certain that the REFLECT operation can be done in 2 cycles in hardware
  - Probably 1 cycle would be enough due to the operation's simplicity, but we “play it safe”

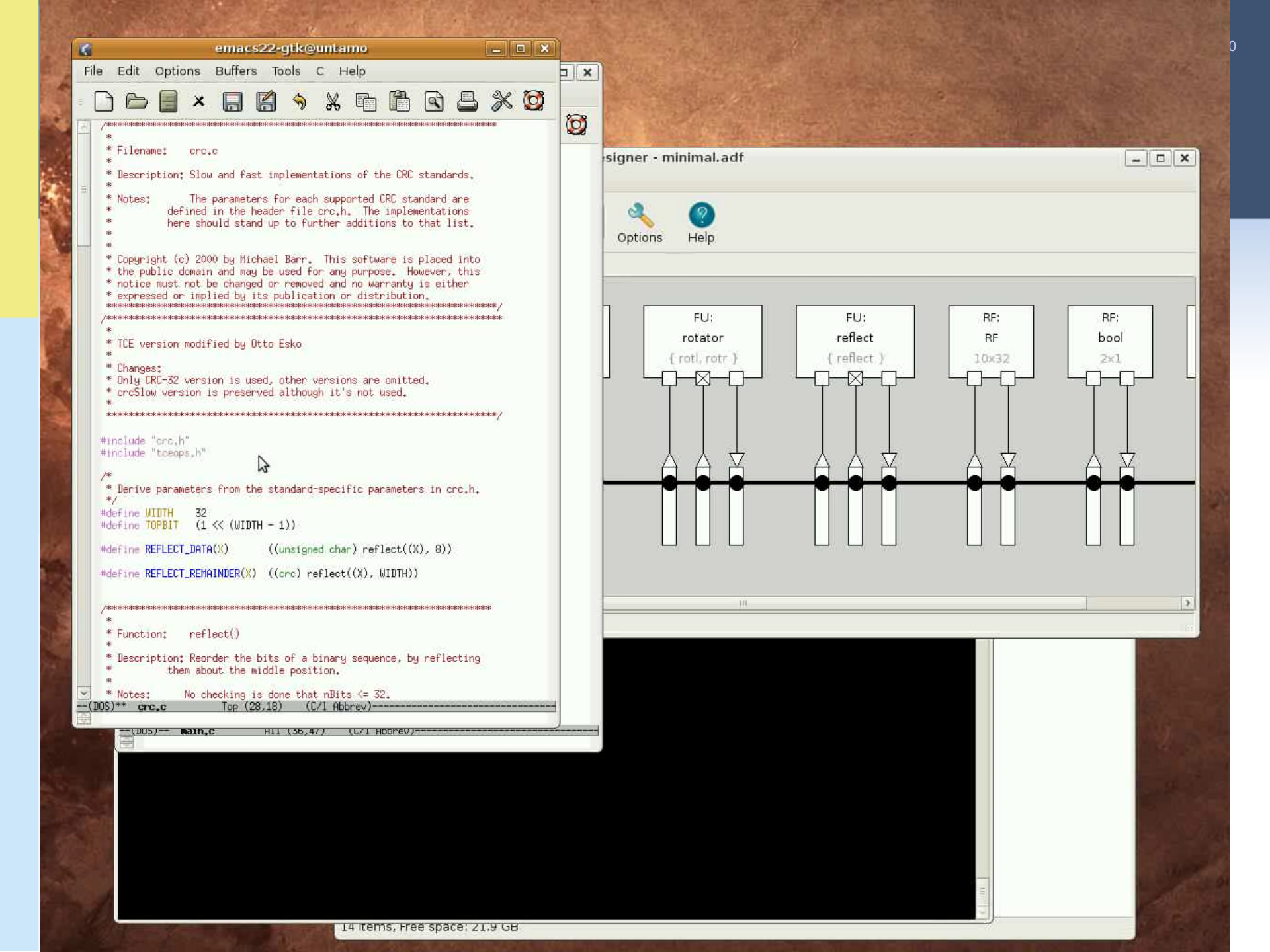

emacs22-gtk@untamo
File
emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Add
Add From HDB
Delete
Modify...
Address Spaces...
Instruction Templates...
Immediate Slots...
Transport Bus Order...
Function Unit...
Register File...
Transport Bus...
Socket...
Bridge...
Immediate Unit...
Global Control Unit...
Address Space...
Modify
Zoom +
Zoom -
Fit Win
Options
Help
FU:
shifter
{ shl, shr, shru }
FU:
rotator
{ rotl, rotr }
RF:
RF
10x32
RF:
bool
2x1
GCU:
gcu
{ jump, call }
14 items, Free space: 21.9 GB

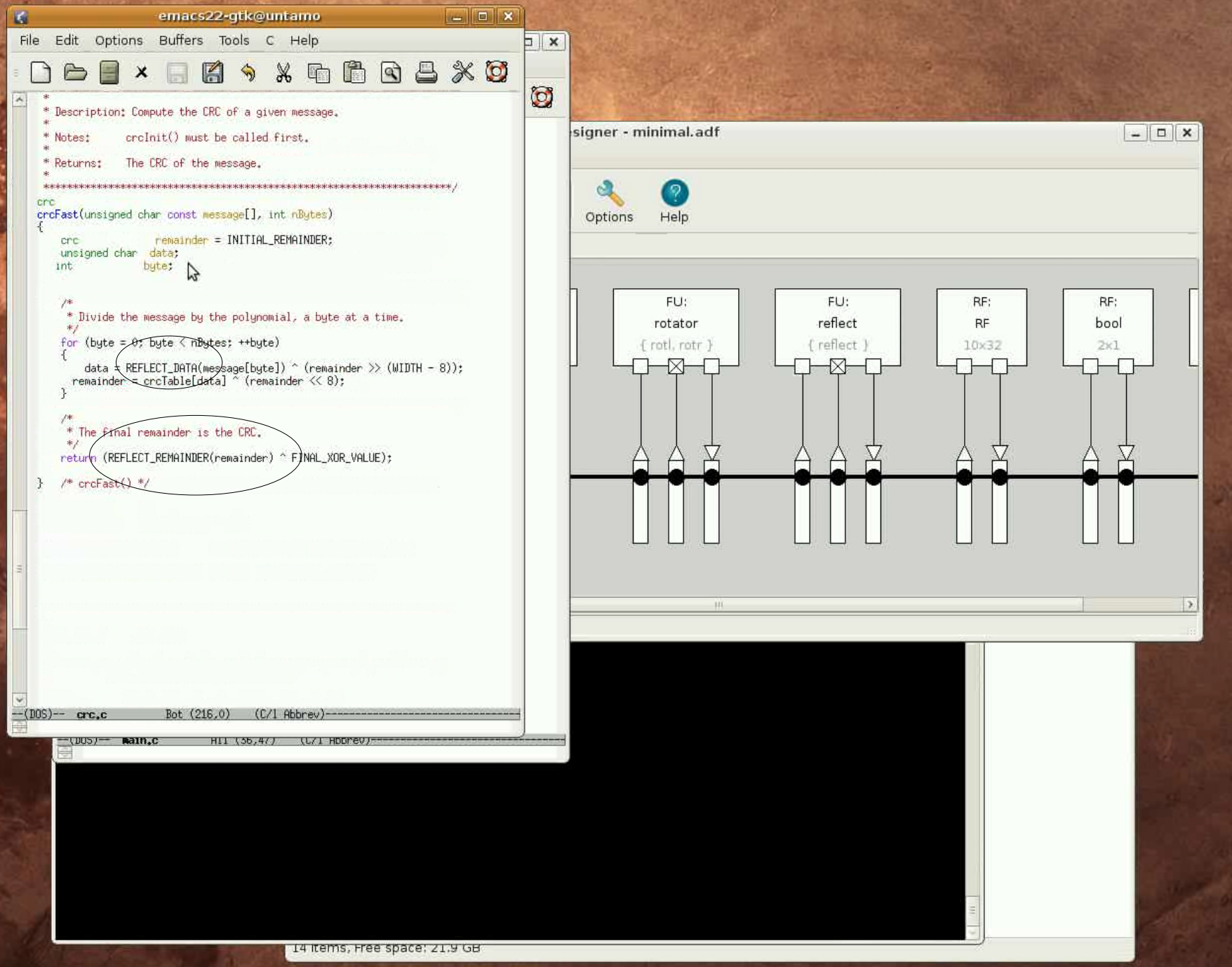

emacs22-gtk@untamo
File
emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New
Open
Save
Delete
minimal.adf
FU:
logic
{ and, ior, xor }
FU:
ext1
{ sxhw }
Function Unit
Name: reflect
Address Space: NONE
Operations:
Name
Width
32
32
32
Add...
Edit...
Add from Opset...
Help
Choose operation
ORDF
REFLECT
ROTL
ROTR
RTC
RTIMER
SHL
SHR
SHRU
Latency: 1
Cancel
OK
Edit...
Delete
OK
Cancel
RF:
bool
2x1
GCU:
gcu
{ jump, call }
14 items, Free space: 21.9 GB

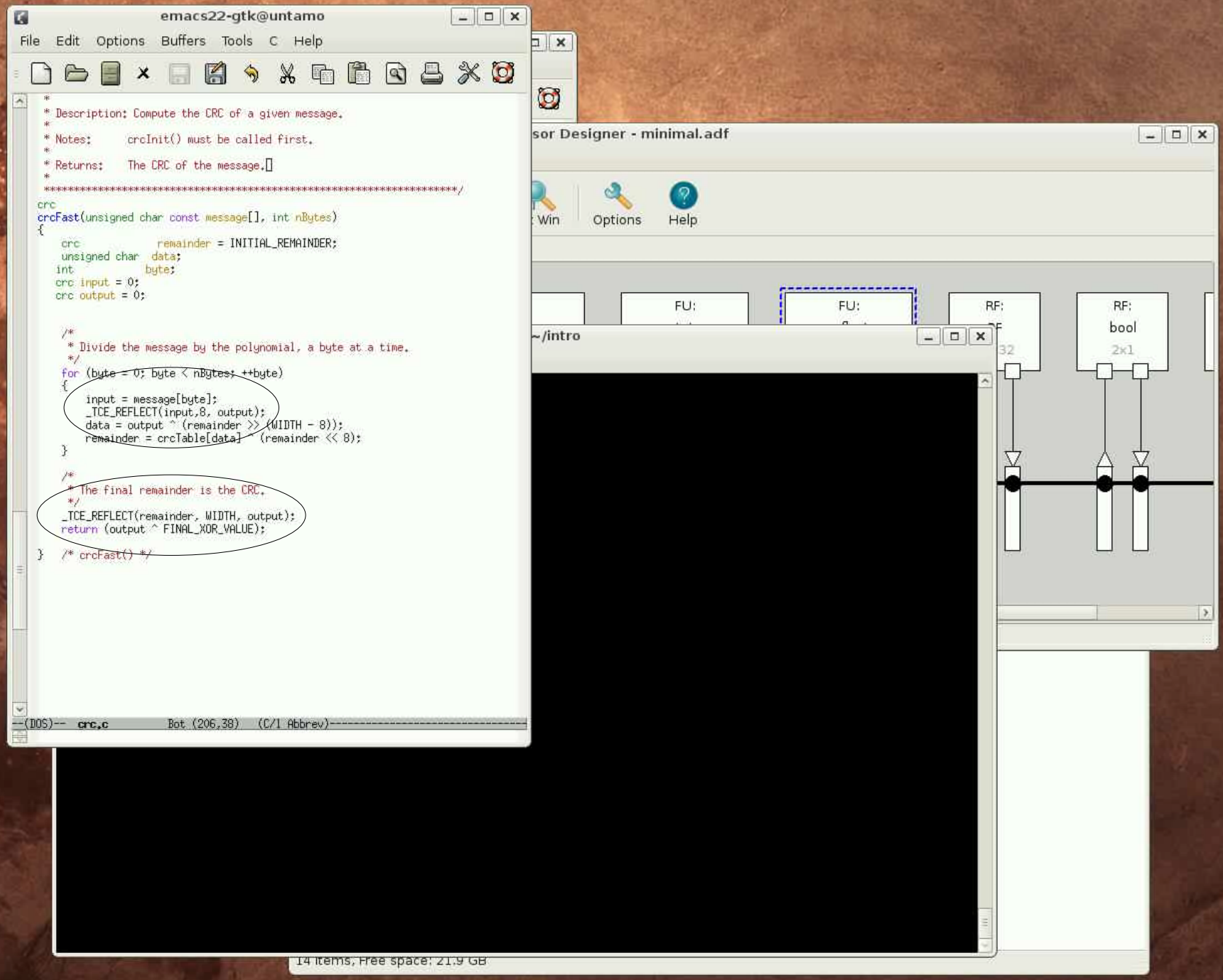

emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
File Edit View Tools Options
New
Open
Save
minimal.adf
FU:
logic
{ and, ior, xor }
GCU:
gcu
{ jump, call }
Operation
Name: reflect
Operands:
Operand Port
3 P3
2 P2
1 P1
Port:
Delete
Operand: 1
Add
Pipeline Resources:
0 1 2 3 4 5 6 7 8 9 10 11 12
Name:
Add
Delete
Operand usage:
1 (R) R
2 (R) R
3 (W) W
Operation latency: 2
Help
OK
Cancel
14 items, Free space: 21.9 GB

Adding SFU to the Machine and Using it in C Code (1.5 minutes)

- Now the architecture supports the REFLECT custom operation with the added function unit
- Let's now use the REFLECT operation from our C code to accelerate the algorithm
- First add:
  - #include “tceops.h”
  - This brings in the macros that are used to invoke TTA operations manually
- Then call the REFLECT operation through a TCE operation macro:
  - _TCE_REFLECT(...);

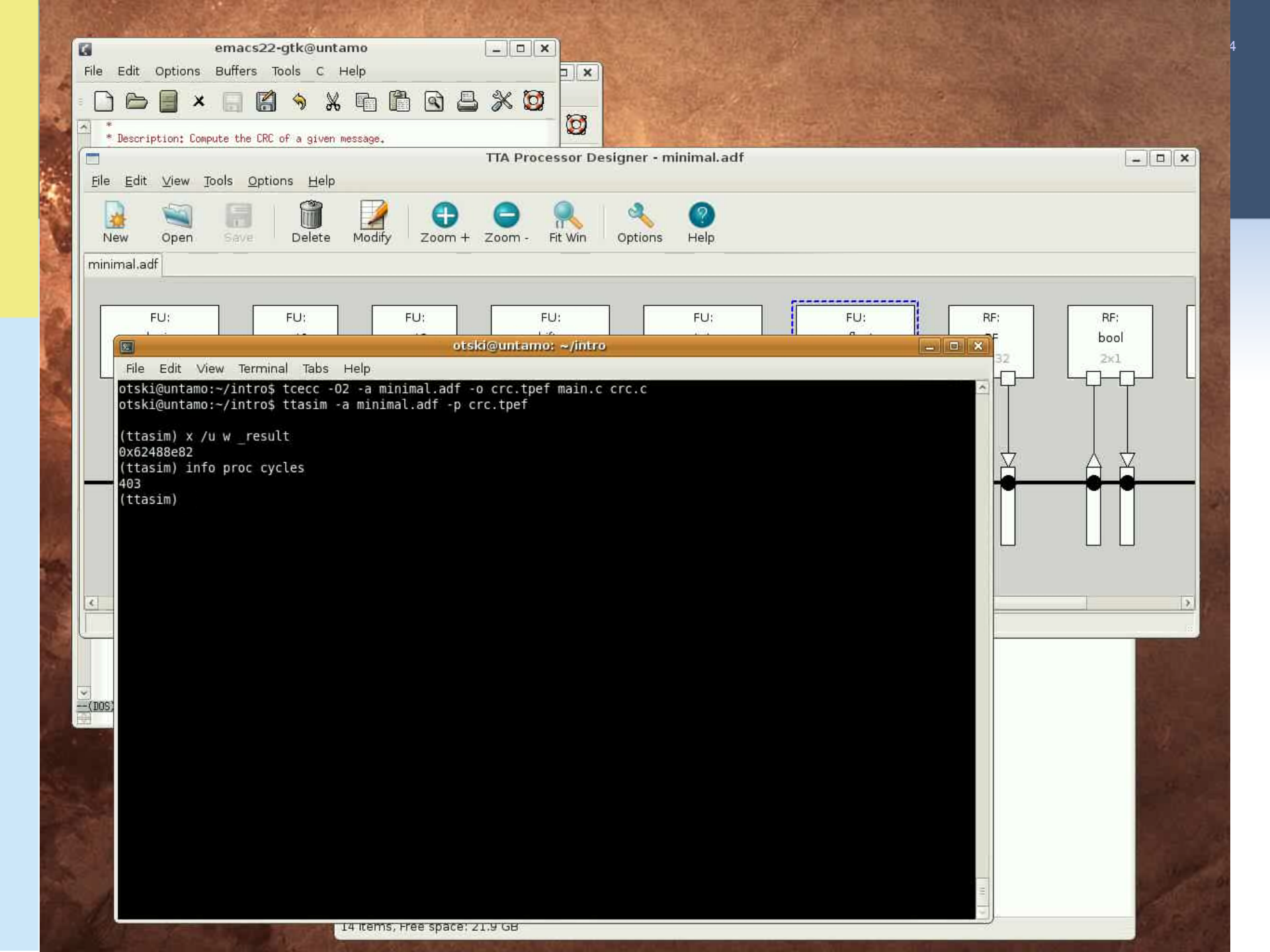

emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
/*
* Filename: crc.c
* Description: Slow and fast implementations of the CRC standards.
* Notes: The parameters for each supported CRC standard are
* defined in the header file crc.h. The implementations
* here should stand up to further additions to that list.
* Copyright (c) 2000 by Michael Barr. This software is placed into
* the public domain and may be used for any purpose. However, this
* notice must not be changed or removed and no warranty is either
* expressed or implied by its publication or distribution.
* TCE version modified by Otto Esko
* Changes:
* Only CRC-32 version is used, other versions are omitted.
* crcSlow version is preserved although it's not used.
#include "crc.h"
#include "tceops.h"
* Derive parameters from the standard-specific parameters in crc.h.
#define WIDTH 32
#define TOPBIT (1 << (WIDTH - 1))
#define REFLECT_DATA(X) ((unsigned char) reflect((X), 8))
#define REFLECT_REMAINDER(X) ((crc) reflect((X), WIDTH))
* Function: reflect()
* Description: Reorder the bits of a binary sequence, by reflecting
* them about the middle position.
* Notes: No checking is done that nBits <= 32.
--(DOS)** crc.c Top (28,18) (C/l Abbrev)
signer - minimal.adf
Options
Help
FU:
rotator
{ rotl, rotr }
FU:
reflect
{ reflect }
RF:
RF
10x32
RF:
bool
2x1
14 items, Free space: 21.9 GB


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
* Description: Compute the CRC of a given message.
* Notes: crcInit() must be called first.
* Returns: The CRC of the message.
crc crcFast(unsigned char const message[], int nBytes)
{
crc remainder = INITIAL_REMAINDER;
unsigned char data;
int byte;
/* Divide the message by the polynomial, a byte at a time. */
for (byte = 0; byte < nBytes; ++byte)
data = REFLECT_DATA(message[byte]) ^ (remainder >> (WIDTH - 8));
remainder = crcTable[data] ^ (remainder << 8);
/* The final remainder is the CRC. */
return (REFLECT_REMAINDER(remainder) ^ FINAL_XOR_VALUE);
} /* crcFast() */
--(DOS)-- crc.c Bot (216,0) (C/l Abbrev)
signer - minimal.adf
Options
Help
FU: rotator { rotl, rotr }
FU: reflect { reflect }
RF: RF 10x32
RF: bool 2x1
14 items, Free space: 21.9 GB


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
* Description: Compute the CRC of a given message.
* Notes: crcInit() must be called first.
* Returns: The CRC of the message.
crc
crcFast(unsigned char const message[], int nBytes)
{
crc remainder = INITIAL_REMAINDER;
unsigned char data;
int byte;
crc input = 0;
crc output = 0;
/*
* Divide the message by the polynomial, a byte at a time.
*/
for (byte = 0; byte < nBytes; ++byte)
{
input = message[byte];
_TCE_REFLECT(input,8, output);
data = output ^ (remainder >> (WIDTH - 8));
remainder = crcTable[data] ^ (remainder << 8);
}
/*
* The final remainder is the CRC.
*/
_TCE_REFLECT(remainder, WIDTH, output);
return (output ^ FINAL_XOR_VALUE);
} /* crcFast() */
--(DOS)-- crc.c Bot (206,38) (C/l Abbrev)
sor Designer - minimal.adf
Win
Options
Help
FU:
FU:
RF:
RF:
bool
2x1
~/intro
14 items, Free space: 21.9 GB

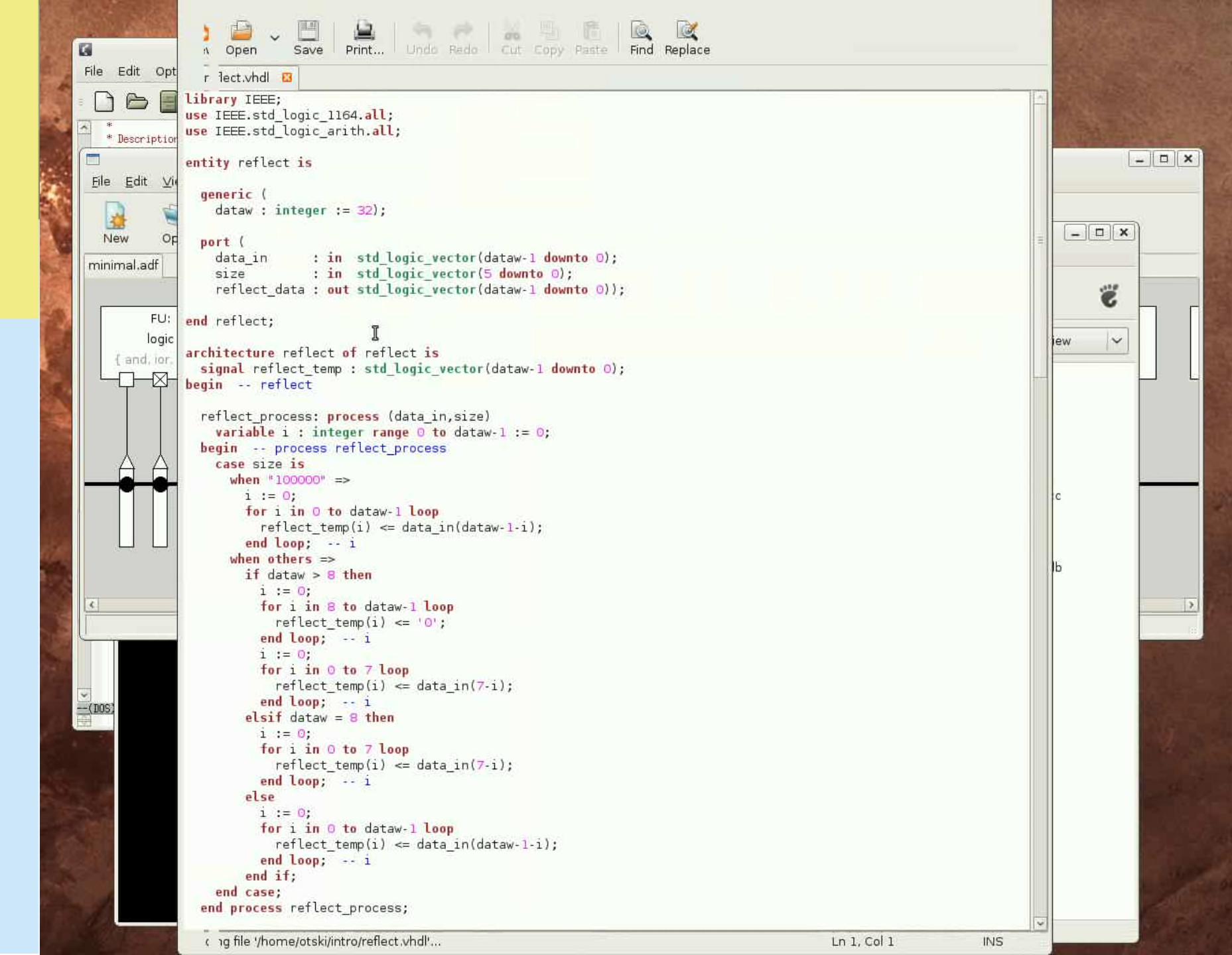
Adding SFU to the Machine and Using it in C Code (1.5 minutes)

- Finally, recompile the code which now uses the custom operation, verify that the program still works correctly, and see its effect to the cycle count using the simulator
  - Cycle count now dropped to **403**
  - By using custom operation we reached a lower cycle count with much less hardware
- Now we could use explorer to increase the performance
  - Current architecture has only one bus
  - By increasing concurrency we would reach lower cycle count


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
* Description: Compute the CRC of a given message.
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
minimal.adf
FU:
RF:
bool
2x1
otski@untamo: ~/intro
File Edit View Terminal Tabs Help
otski@untamo:~/intro$ tcecc -O2 -a minimal.adf -o crc.tpef main.c crc.c
otski@untamo:~/intro$ ttasim -a minimal.adf -p crc.tpef
(ttasim) x /u w _result
0x62488e82
(ttasim) info proc cycles
403
(ttasim)
14 items, Free space: 21.9 GB

Adding Implementation of the SFU to the Hardware Database (50 sec)

- Now we have found a good custom operation to accelerate our algorithm and used it in our architecture and C code
- In order to generate VHDL for the processor, we still need to add an implementation of the SFU to a Hardware Database (HDB)
- Of course, implementing the SFU might take a bit longer than the 50 sec, thus we use a previously implemented VHDL block for demonstration purposes :)
- HDBEditor is a GUI for editing HDBs, we use it to add the implementation to an HDB along with the data needed to generate a processor
  - The names of the input/output ports and the entity name in the VHDL, etc..

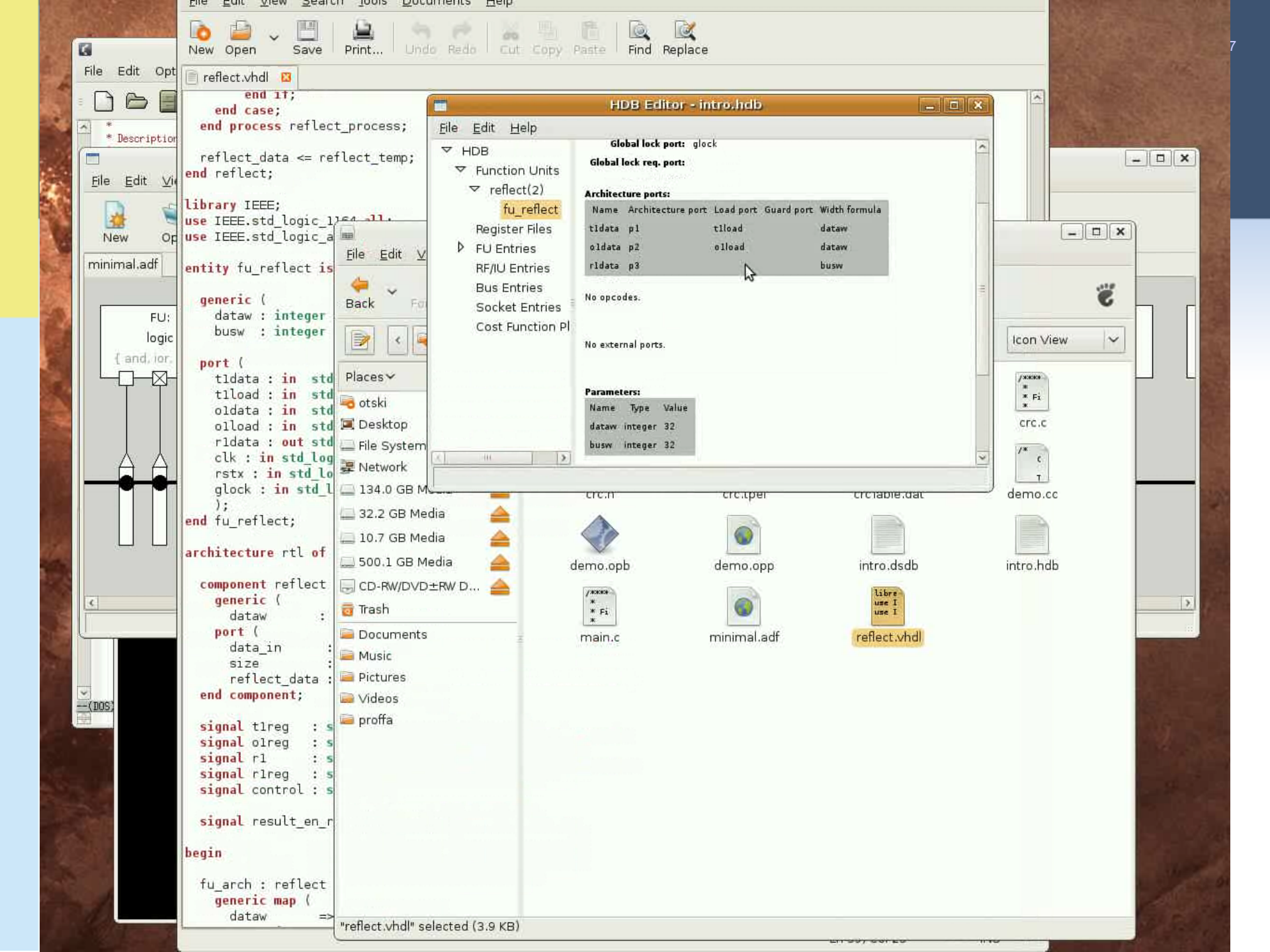


Open Save Print... Undo Redo Cut Copy Paste Find Replace

lect.vhdl

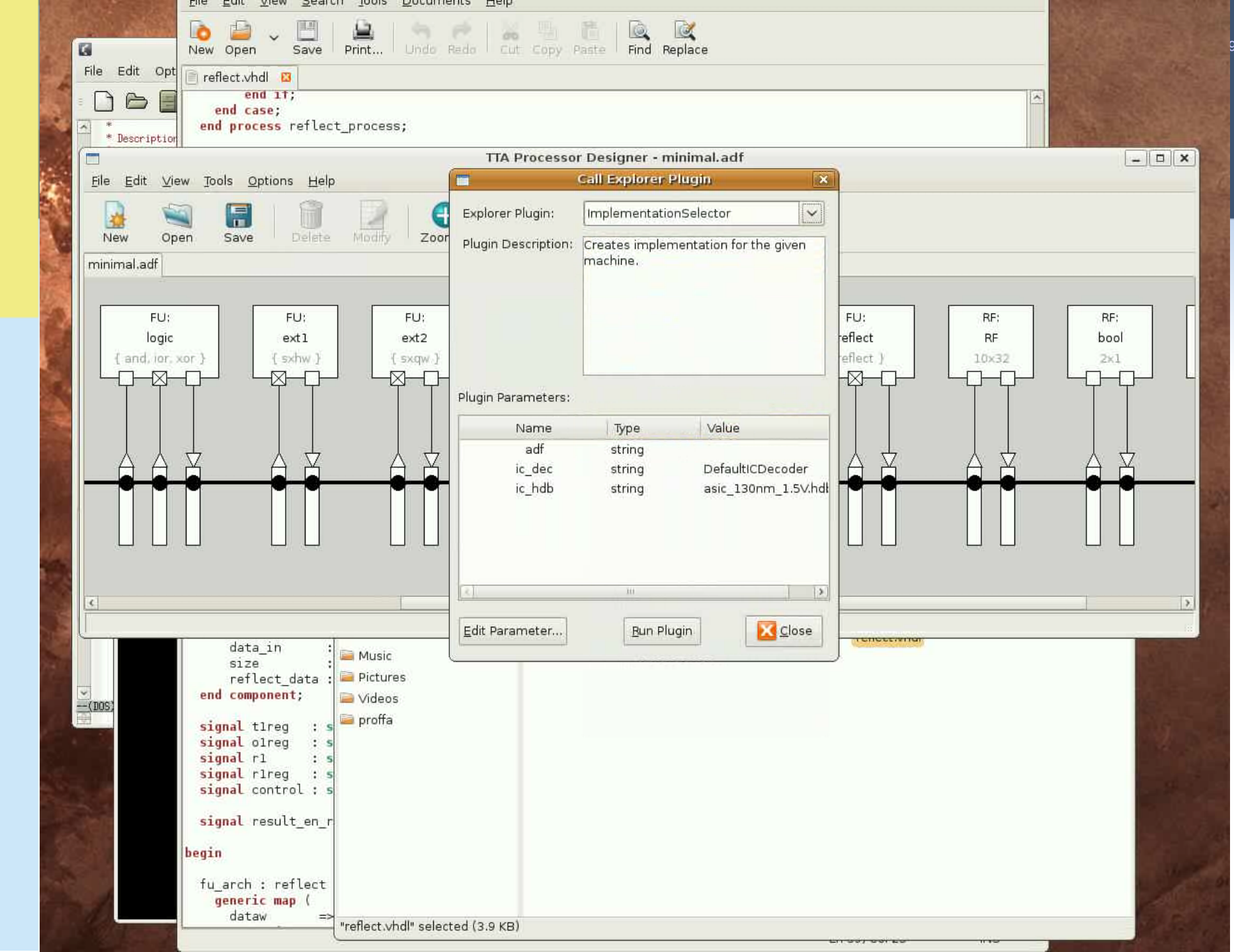
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity reflect is

  generic (
    dataw : integer := 32);

  port (
    data_in      : in  std_logic_vector(dataw-1 downto 0);
    size         : in  std_logic_vector(5 downto 0);
    reflect_data : out std_logic_vector(dataw-1 downto 0));

end reflect;

architecture reflect of reflect is
  signal reflect_temp : std_logic_vector(dataw-1 downto 0);
begin  -- reflect

  reflect_process: process (data_in,size)
    variable i : integer range 0 to dataw-1 := 0;
  begin  -- process reflect_process
    case size is
      when "100000" =>
        i := 0;
        for i in 0 to dataw-1 loop
          reflect_temp(i) <= data_in(dataw-1-i);
        end loop;  -- i
      when others =>
        if dataw > 8 then
          i := 0;
          for i in 8 to dataw-1 loop
            reflect_temp(i) <= '0';
          end loop;  -- i
          i := 0;
          for i in 0 to 7 loop
            reflect_temp(i) <= data_in(7-i);
          end loop;  -- i
        elsif dataw = 8 then
          i := 0;
          for i in 0 to 7 loop
            reflect_temp(i) <= data_in(7-i);
          end loop;  -- i
        else
          i := 0;
          for i in 0 to dataw-1 loop
            reflect_temp(i) <= data_in(dataw-1-i);
          end loop;  -- i
        end if;
    end case;
  end process reflect_process;
```

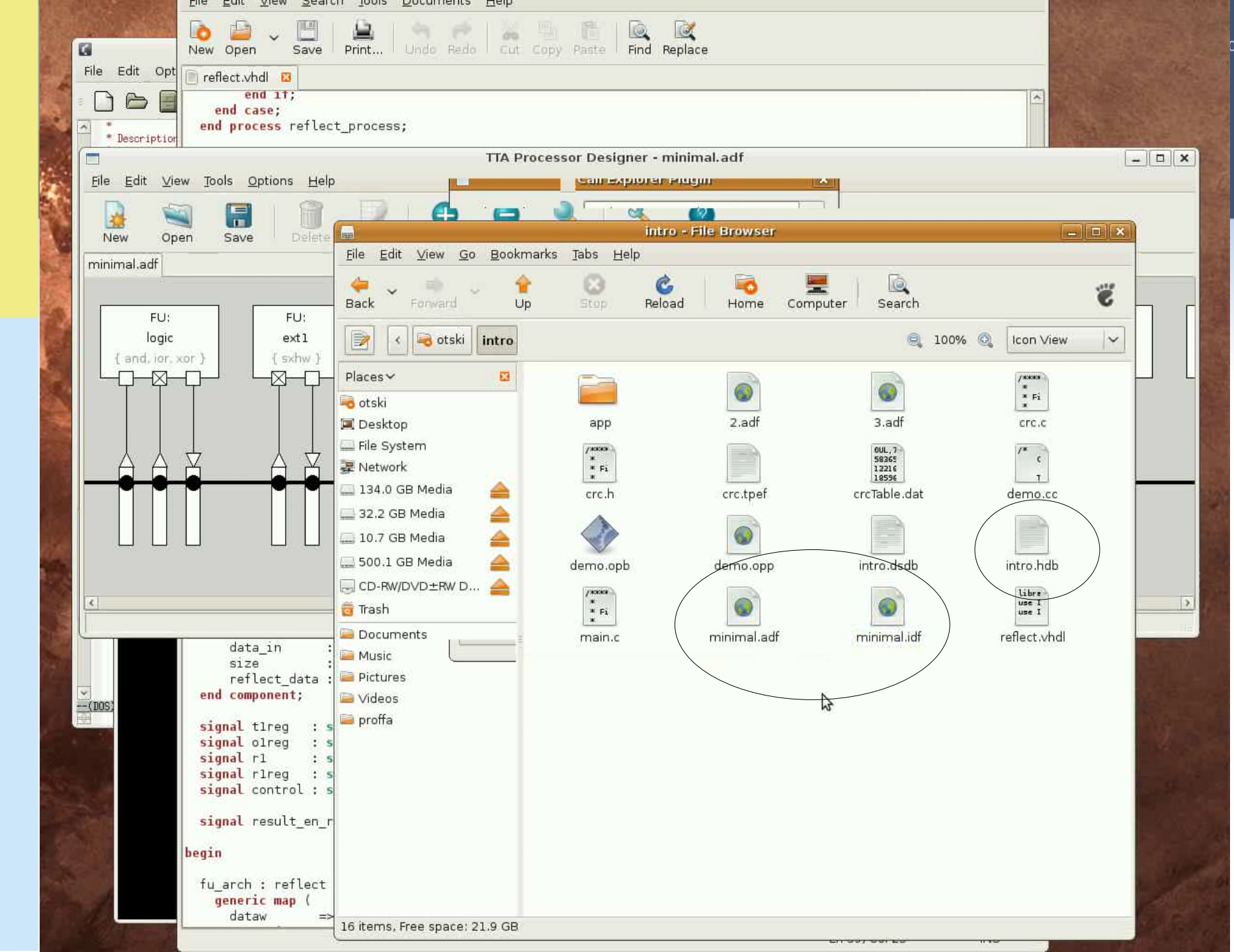
g file '/home/otski/intro/reflect.vhdl'... Ln 1, Col 1 INS


File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
HDB Editor - intro.hdb
File Edit Help
HDB
Function Units
reflect(2)
fu_reflect
Register Files
FU Entries
RF/IU Entries
Bus Entries
Socket Entries
Cost Function Pl
Global lock port: glock
Global lock req. port:
Architecture ports:
Name Architecture port Load port Guard port Width formula
t1data p1 t1load dataw
o1data p2 o1load dataw
r1data p3 busw
No opcodes.
No external ports.
Parameters:
Name Type Value
dataw integer 32
busw integer 32
Places
otski
Desktop
File System
Network
32.2 GB Media
10.7 GB Media
500.1 GB Media
CD-RW/DVD±RW D...
Trash
Documents
Music
Pictures
Videos
proffa
Icon View
crc.c
demo.cc
demo.opb
demo.opp
intro.dsdb
intro.hdb
main.c
minimal.adf
reflect.vhdl
"reflect.vhdl" selected (3.9 KB)

Adding Implementation of the SFU to the Hardware Database (50 sec)

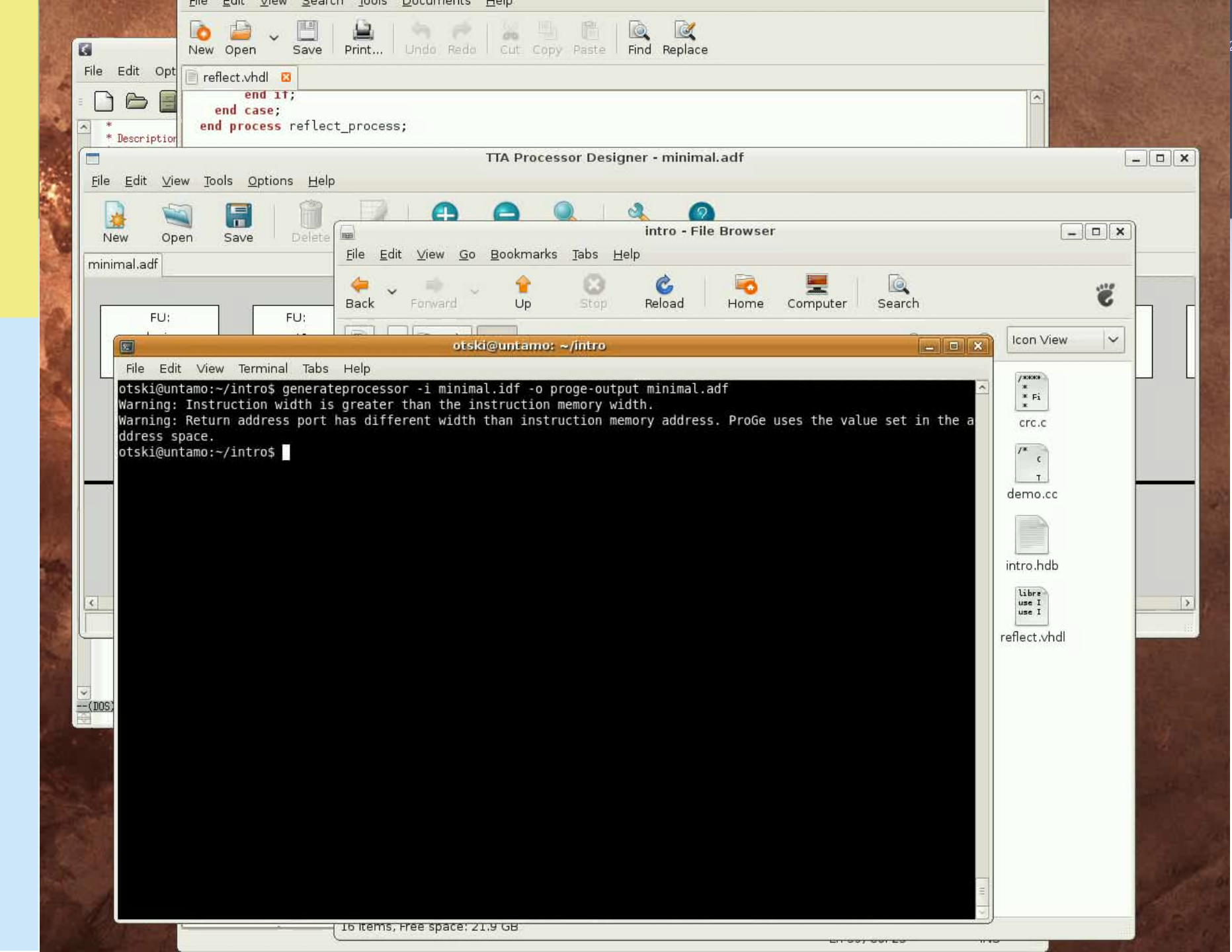
- Now we have added an implementation of the REFLECT SFU to a HDB
- Finally we need to connect the architecture of the FU in our TTA architecture file to this implementation
  - Use automated exploration plugin for this
- In TCE, architecture of the processor components and the actual implementation are separated
  - Architecture components (in ADF files edited with ProDe) are connected to HDB implementations through an Implementation Definition File (IDF)
  - Architecture definition file (ADF), implementation definition file (IDF) and one or more Hardware Databases (HDB) form a “processor configuration” that can be outputted as a VHDL implementation

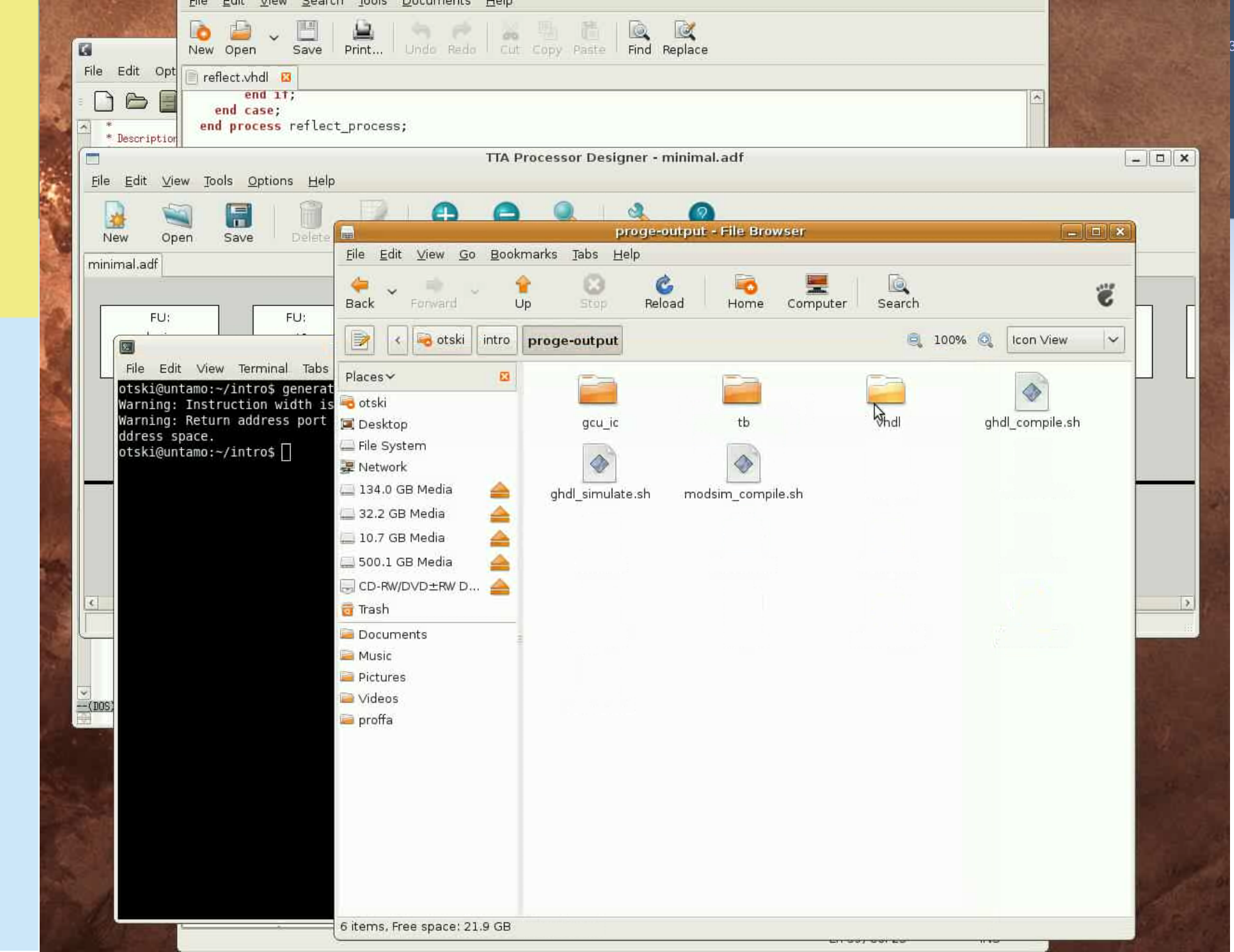

File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
end if;
end case;
end process reflect_process;
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify
minimal.adf
FU: logic { and, ior, xor }
FU: ext1 { sxhw }
FU: ext2 { sxqw }
FU: reflect
RF: RF 10x32
RF: bool 2x1
Call Explorer Plugin
Explorer Plugin: ImplementationSelector
Plugin Description: Creates implementation for the given machine.
Plugin Parameters:
Name Type Value
adf string
ic_dec string DefaultICDecoder
ic_hdb string asic_130nm_1.5V.hdb
Edit Parameter...
Run Plugin
Close
data_in :
size :
reflect_data :
end component;
signal t1reg : s
signal o1reg : s
signal r1 : s
signal r1reg : s
signal control : s
signal result_en_r
begin
fu_arch : reflect
generic map (
dataw =>
Music
Pictures
Videos
proffa
"reflect.vhdl" selected (3.9 KB)

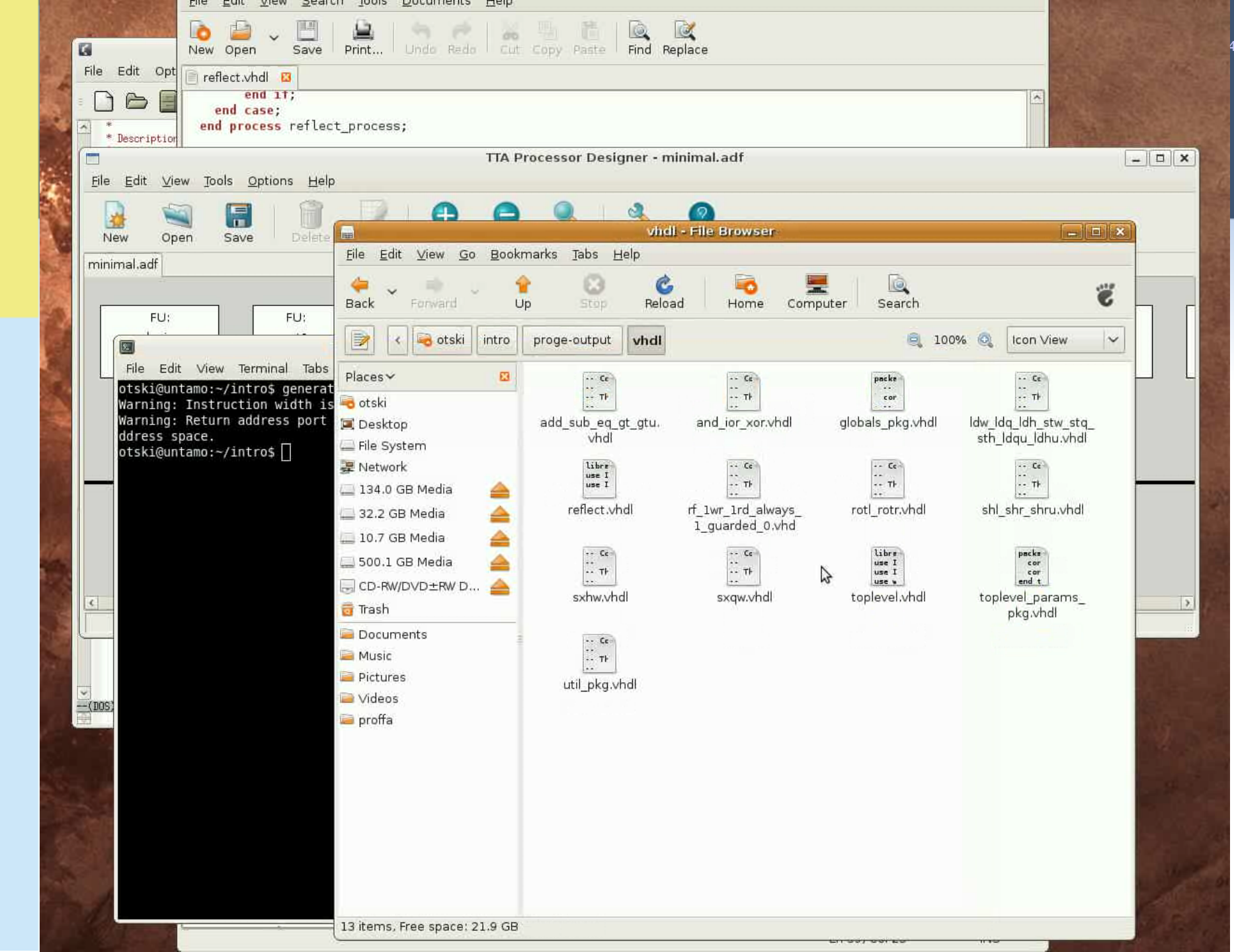

File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
end if;
end case;
end process reflect_process;
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete
minimal.adf
FU: logic { and, ior, xor }
FU: ext1 { sxhw }
intro - File Browser
File Edit View Go Bookmarks Tabs Help
Back Forward Up Stop Reload Home Computer Search
otski intro
100% Icon View
Places
otski
Desktop
File System
Network
134.0 GB Media
32.2 GB Media
10.7 GB Media
500.1 GB Media
CD-RW/DVD±RW D...
Trash
Documents
Music
Pictures
Videos
proffa
app
2.adf
3.adf
crc.c
crc.h
crc.tpef
crcTable.dat
demo.cc
demo.opb
demo.opp
intro.dsdb
intro.hdb
main.c
minimal.adf
minimal.idf
reflect.vhdl
16 items, Free space: 21.9 GB
data_in :
size :
reflect_data :
end component;
signal t1reg : s
signal o1reg : s
signal r1 : s
signal r1reg : s
signal control : s
signal result_en_r
begin
fu_arch : reflect
generic map (
dataw =>

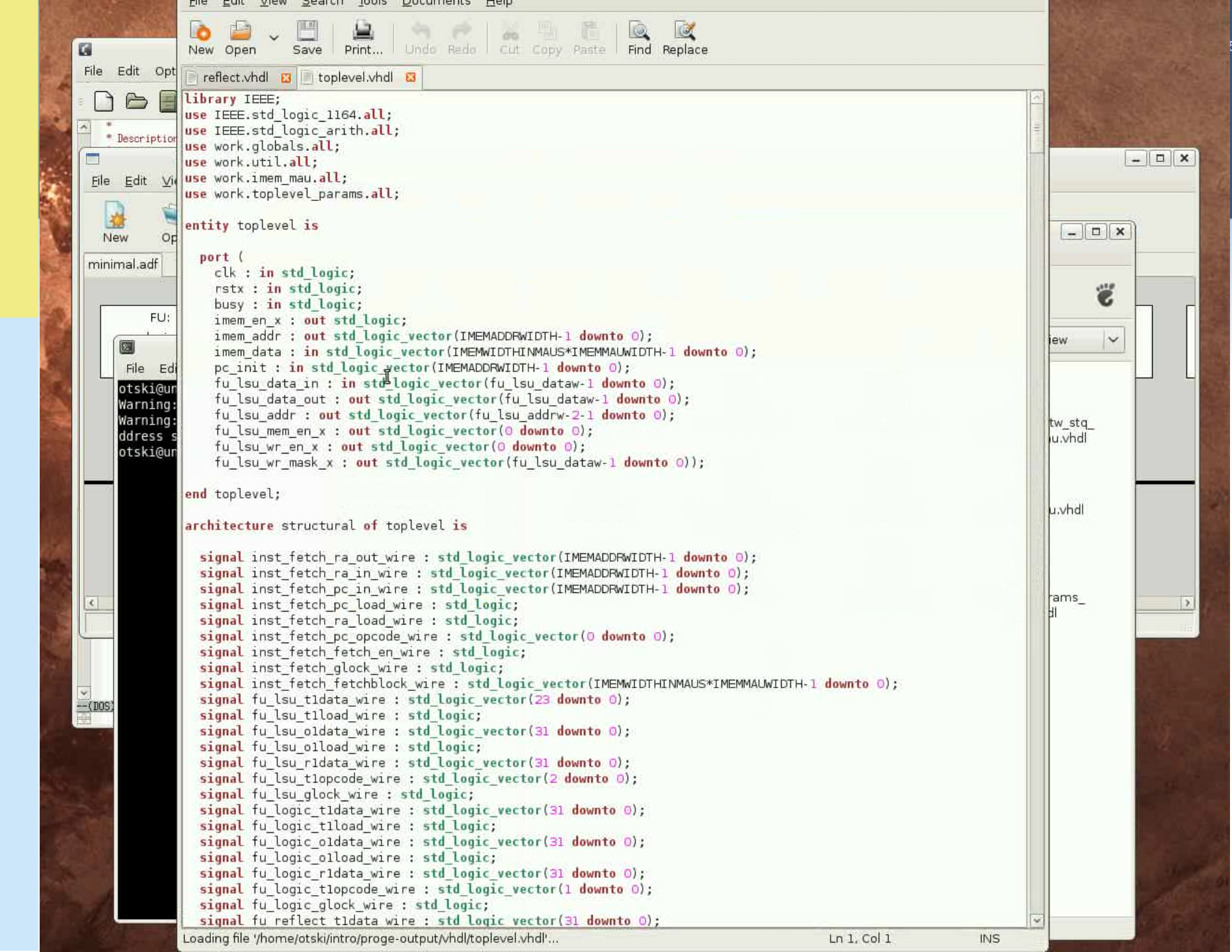
Generating the Processor (32 sec)

- Now we have all we need to generate the processor implementation in VHDL
- For this we use the Processor Generator (ProGe) tool which can be invoked from the command line or from the ProDe GUI


New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
end if;
end case;
end process reflect_process;
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete
minimal.adf
FU:
FU:
intro - File Browser
File Edit View Go Bookmarks Tabs Help
Back Forward Up Stop Reload Home Computer Search
Icon View
crc.c
demo.cc
intro.hdb
reflect.vhdl
16 items, Free space: 21.9 GB
otski@untamo: ~/intro
File Edit View Terminal Tabs Help
otski@untamo:~/intro$ generateprocessor -i minimal.idf -o proge-output minimal.adf
Warning: Instruction width is greater than the instruction memory width.
Warning: Return address port has different width than instruction memory address. ProGe uses the value set in the a
ddress space.
otski@untamo:~/intro$

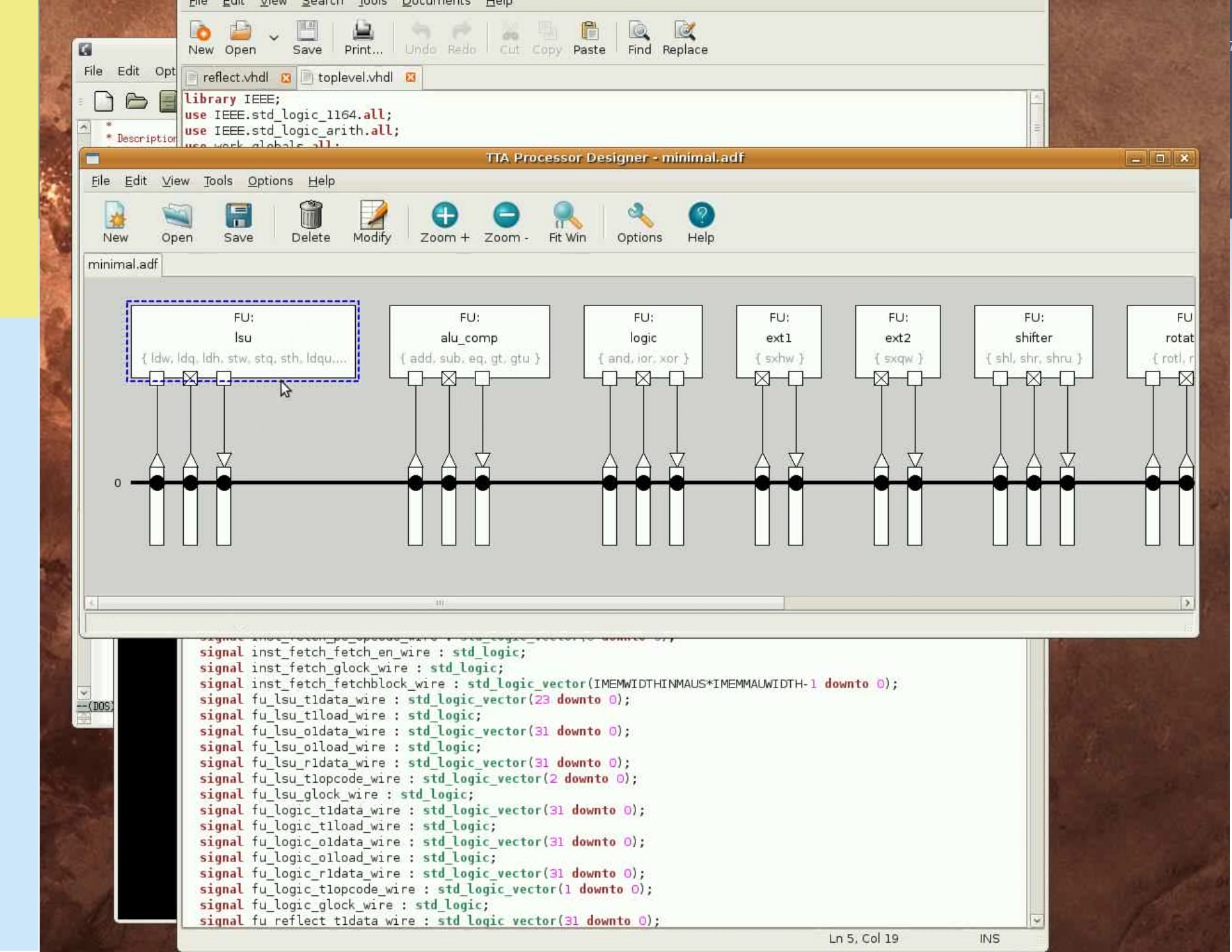

File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
end if;
end case;
end process reflect_process;
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete
minimal.adf
FU:
FU:
proge-output - File Browser
File Edit View Go Bookmarks Tabs Help
Back Forward Up Stop Reload Home Computer Search
otski intro proge-output
100% Icon View
Places
otski
Desktop
File System
Network
134.0 GB Media
32.2 GB Media
10.7 GB Media
500.1 GB Media
CD-RW/DVD±RW D...
Trash
Documents
Music
Pictures
Videos
proffa
gcu_ic
tb
vhdl
ghdl_compile.sh
ghdl_simulate.sh
modsim_compile.sh
6 items, Free space: 21.9 GB
File Edit View Terminal Tabs
otski@untamo:~/intro$ generat
Warning: Instruction width is
Warning: Return address port
ddress space.
otski@untamo:~/intro$

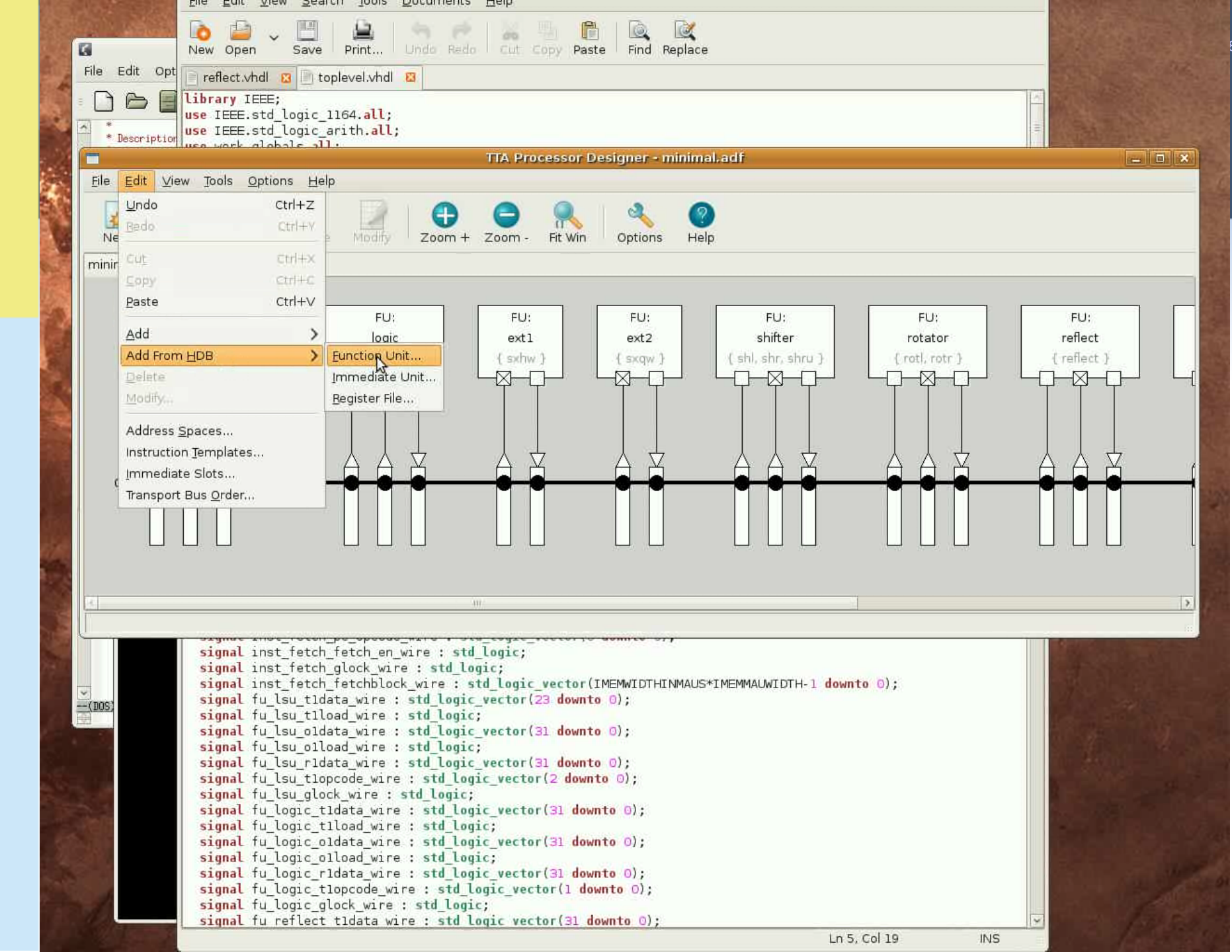

File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
end if;
end case;
end process reflect_process;
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete
minimal.adf
FU:
FU:
vhdl - File Browser
File Edit View Go Bookmarks Tabs Help
Back Forward Up Stop Reload Home Computer Search
otski intro proge-output vhdl
100% Icon View
Places
otski
Desktop
File System
Network
134.0 GB Media
32.2 GB Media
10.7 GB Media
500.1 GB Media
CD-RW/DVD±RW D...
Trash
Documents
Music
Pictures
Videos
proffa
add_sub_eq_gt_gtu.vhdl
and_ior_xor.vhdl
globals_pkg.vhdl
ldw_ldq_ldh_stw_stq_sth_ldqu_ldhu.vhdl
reflect.vhdl
rf_1wr_1rd_always_1_guarded_0.vhd
rotl_rotr.vhdl
shl_shr_shru.vhdl
sxhw.vhdl
sxqw.vhdl
toplevel.vhdl
toplevel_params_pkg.vhdl
util_pkg.vhdl
13 items, Free space: 21.9 GB
File Edit View Terminal Tabs
otski@untamo:~/intro$ generat
Warning: Instruction width is
Warning: Return address port
ddress space.
otski@untamo:~/intro$

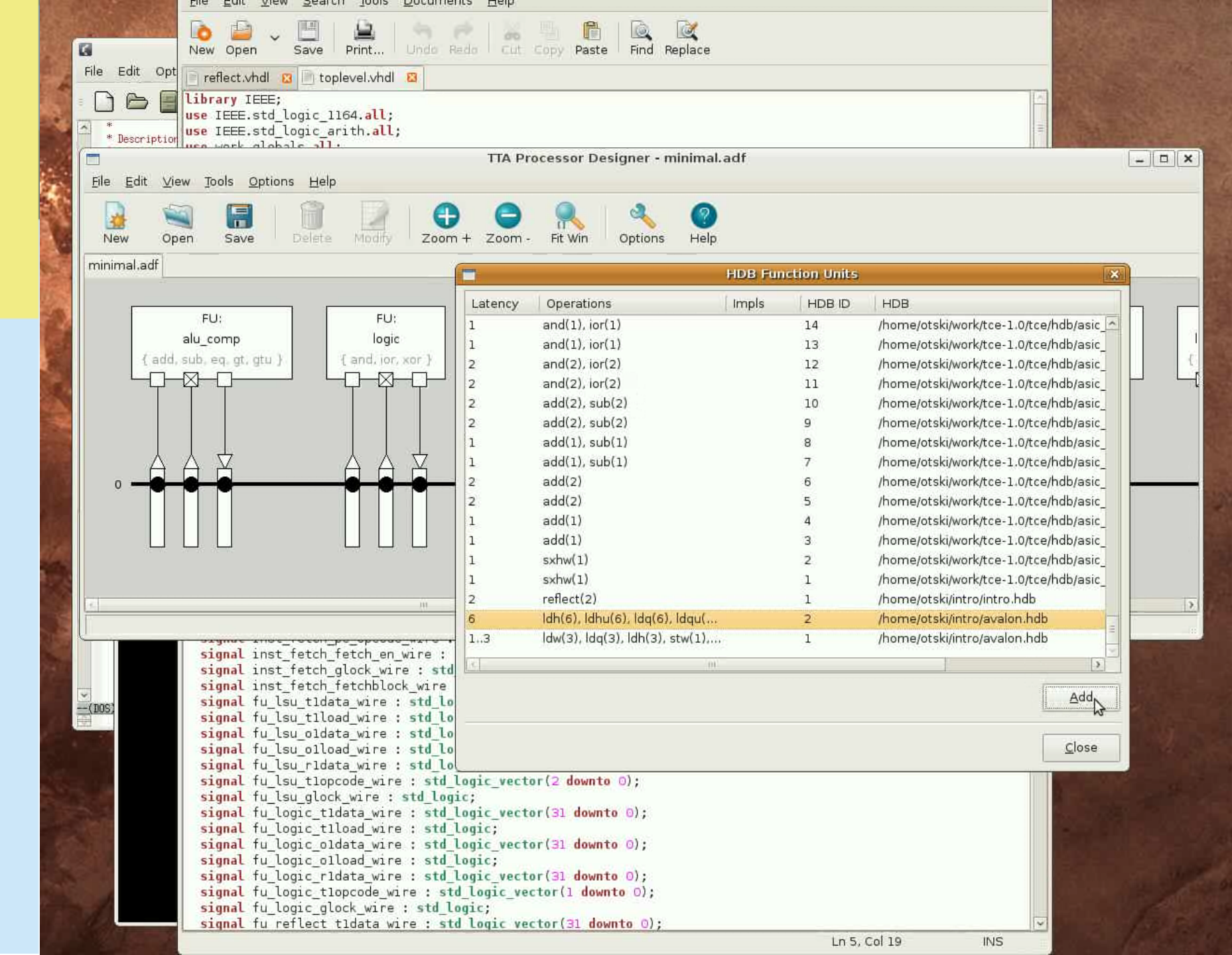

File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
toplevel.vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use work.globals.all;
use work.util.all;
use work.imem_mau.all;
use work.toplevel_params.all;
entity toplevel is
port (
clk : in std_logic;
rstx : in std_logic;
busy : in std_logic;
imem_en_x : out std_logic;
imem_addr : out std_logic_vector(IMEMADDRWIDTH-1 downto 0);
imem_data : in std_logic_vector(IMEMWIDTHINMAUS*IMEMMAUWIDTH-1 downto 0);
pc_init : in std_logic_vector(IMEMADDRWIDTH-1 downto 0);
fu_lsu_data_in : in std_logic_vector(fu_lsu_dataw-1 downto 0);
fu_lsu_data_out : out std_logic_vector(fu_lsu_dataw-1 downto 0);
fu_lsu_addr : out std_logic_vector(fu_lsu_addrw-2-1 downto 0);
fu_lsu_mem_en_x : out std_logic_vector(0 downto 0);
fu_lsu_wr_en_x : out std_logic_vector(0 downto 0);
fu_lsu_wr_mask_x : out std_logic_vector(fu_lsu_dataw-1 downto 0));
end toplevel;
architecture structural of toplevel is
signal inst_fetch_ra_out_wire : std_logic_vector(IMEMADDRWIDTH-1 downto 0);
signal inst_fetch_ra_in_wire : std_logic_vector(IMEMADDRWIDTH-1 downto 0);
signal inst_fetch_pc_in_wire : std_logic_vector(IMEMADDRWIDTH-1 downto 0);
signal inst_fetch_pc_load_wire : std_logic;
signal inst_fetch_ra_load_wire : std_logic;
signal inst_fetch_pc_opcode_wire : std_logic_vector(0 downto 0);
signal inst_fetch_fetch_en_wire : std_logic;
signal inst_fetch_glock_wire : std_logic;
signal inst_fetch_fetchblock_wire : std_logic_vector(IMEMWIDTHINMAUS*IMEMMAUWIDTH-1 downto 0);
signal fu_lsu_t1data_wire : std_logic_vector(23 downto 0);
signal fu_lsu_t1load_wire : std_logic;
signal fu_lsu_o1data_wire : std_logic_vector(31 downto 0);
signal fu_lsu_o1load_wire : std_logic;
signal fu_lsu_r1data_wire : std_logic_vector(31 downto 0);
signal fu_lsu_t1opcode_wire : std_logic_vector(2 downto 0);
signal fu_lsu_glock_wire : std_logic;
signal fu_logic_t1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1load_wire : std_logic;
signal fu_logic_o1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_o1load_wire : std_logic;
signal fu_logic_r1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1opcode_wire : std_logic_vector(1 downto 0);
signal fu_logic_glock_wire : std_logic;
signal fu_reflect_t1data_wire : std_logic_vector(31 downto 0);
Loading file '/home/otski/intro/proge-output/vhdl/toplevel.vhdl'...
Ln 1, Col 1
INS

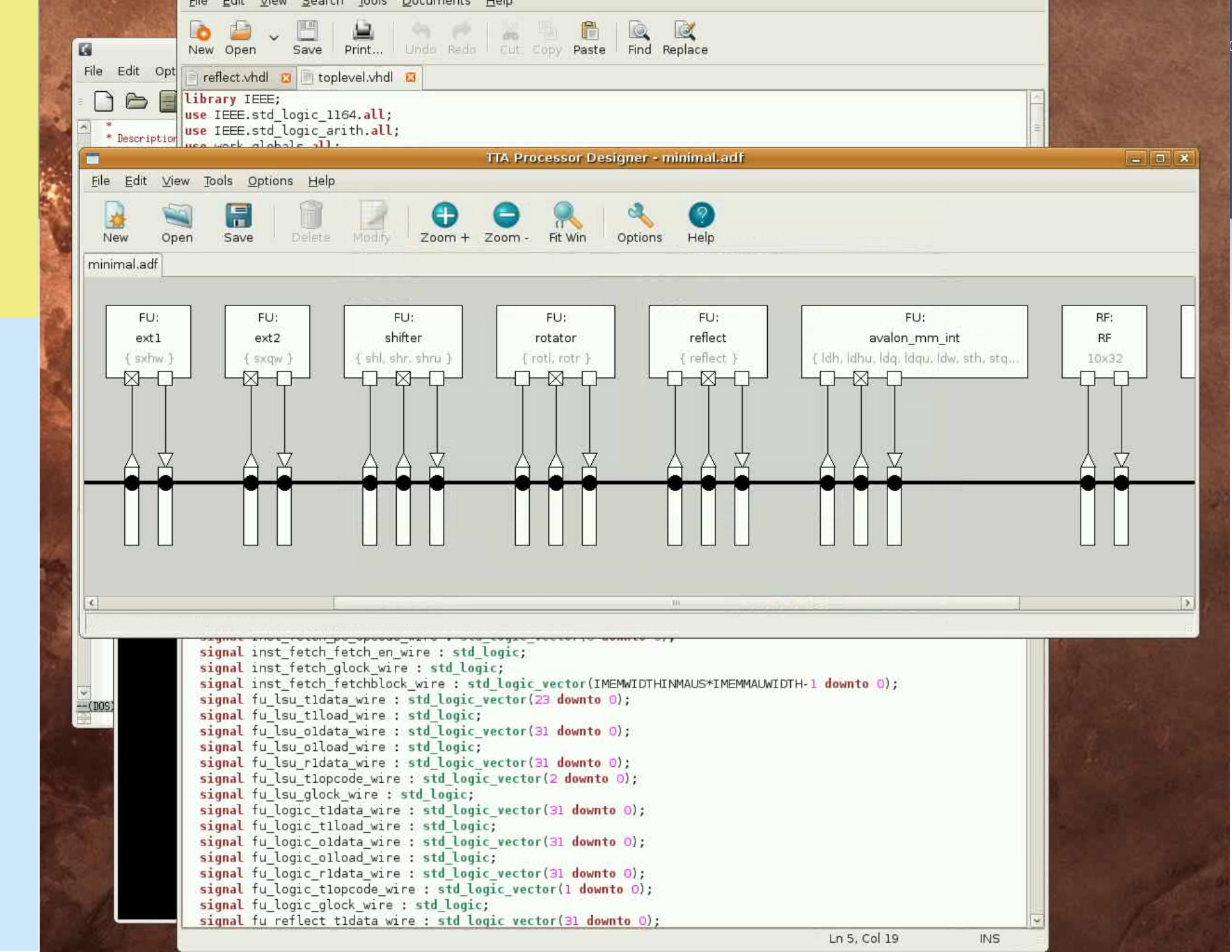
Change Load-store Unit to an Avalon Bus Load/store Unit (36 secs)

- Next we'll use an FPGA board to test the processor
- For this we need to change the load-store unit function unit implementation to one that supports Altera's Avalon interface
  - We'll use the Altera Memory Mapped Interface
  - TTA acts as a master on the bus
  - This way we can use Altera's IP-components
- This can be done quickly with the Processor Designer tool


File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
toplevel.vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
minimal.adf
FU:
lsu
{ ldw, ldq, ldh, stw, stq, sth, ldqu,...
FU:
alu_comp
{ add, sub, eq, gt, gtu }
FU:
logic
{ and, ior, xor }
FU:
ext1
{ sxhw }
FU:
ext2
{ sxqw }
FU:
shifter
{ shl, shr, shru }
FU
rotat
{ rotl, r
0
signal inst_fetch_fetch_en_wire : std_logic;
signal inst_fetch_glock_wire : std_logic;
signal inst_fetch_fetchblock_wire : std_logic_vector(IMEMWIDTHINMAUS*IMEMMAUWIDTH-1 downto 0);
signal fu_lsu_t1data_wire : std_logic_vector(23 downto 0);
signal fu_lsu_t1load_wire : std_logic;
signal fu_lsu_o1data_wire : std_logic_vector(31 downto 0);
signal fu_lsu_o1load_wire : std_logic;
signal fu_lsu_r1data_wire : std_logic_vector(31 downto 0);
signal fu_lsu_t1opcode_wire : std_logic_vector(2 downto 0);
signal fu_lsu_glock_wire : std_logic;
signal fu_logic_t1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1load_wire : std_logic;
signal fu_logic_o1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_o1load_wire : std_logic;
signal fu_logic_r1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1opcode_wire : std_logic_vector(1 downto 0);
signal fu_logic_glock_wire : std_logic;
signal fu_reflect_t1data_wire : std_logic_vector(31 downto 0);
Ln 5, Col 19
INS

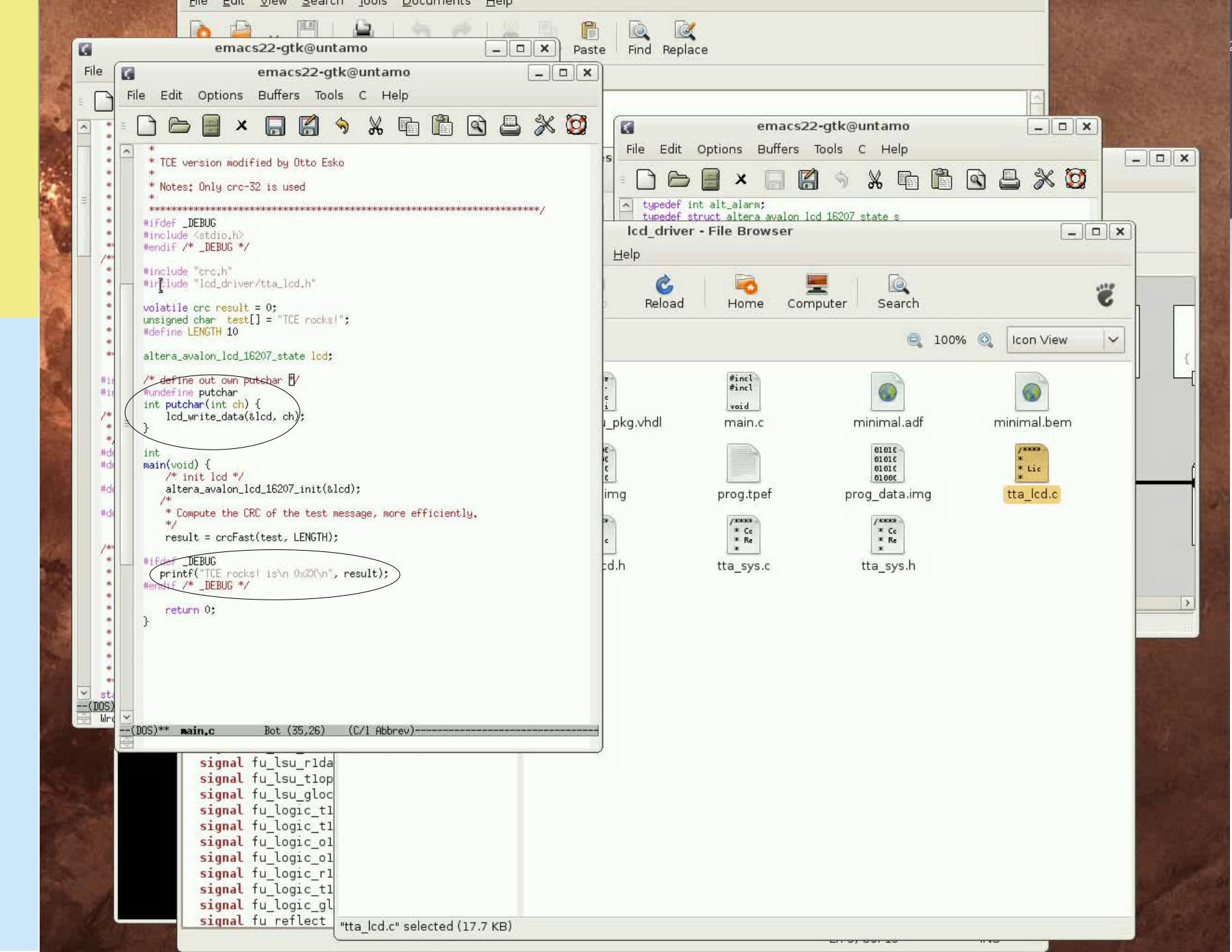

File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
toplevel.vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Add
Add From HDB
Delete
Modify...
Address Spaces...
Instruction Templates...
Immediate Slots...
Transport Bus Order...
Function Unit...
Immediate Unit...
Register File...
Modify
Zoom +
Zoom -
Fit Win
Options
Help
FU: logic
FU: ext1 { sxhw }
FU: ext2 { sxqw }
FU: shifter { shl, shr, shru }
FU: rotator { rotl, rotr }
FU: reflect { reflect }
signal inst_fetch_fetch_en_wire : std_logic;
signal inst_fetch_glock_wire : std_logic;
signal inst_fetch_fetchblock_wire : std_logic_vector(IMEMWIDTHINMAUS*IMEMMAUWIDTH-1 downto 0);
signal fu_lsu_t1data_wire : std_logic_vector(23 downto 0);
signal fu_lsu_t1load_wire : std_logic;
signal fu_lsu_o1data_wire : std_logic_vector(31 downto 0);
signal fu_lsu_o1load_wire : std_logic;
signal fu_lsu_r1data_wire : std_logic_vector(31 downto 0);
signal fu_lsu_t1opcode_wire : std_logic_vector(2 downto 0);
signal fu_lsu_glock_wire : std_logic;
signal fu_logic_t1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1load_wire : std_logic;
signal fu_logic_o1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_o1load_wire : std_logic;
signal fu_logic_r1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1opcode_wire : std_logic_vector(1 downto 0);
signal fu_logic_glock_wire : std_logic;
signal fu_reflect_t1data_wire : std_logic_vector(31 downto 0);
Ln 5, Col 19
INS


File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
toplevel.vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
minimal.adf
FU:
alu_comp
{ add, sub, eq, gt, gtu }
FU:
logic
{ and, ior, xor }
0
HDB Function Units
Latency Operations Impls HDB ID HDB
1 and(1), ior(1) 14 /home/otski/work/tce-1.0/tce/hdb/asic_
1 and(1), ior(1) 13 /home/otski/work/tce-1.0/tce/hdb/asic_
2 and(2), ior(2) 12 /home/otski/work/tce-1.0/tce/hdb/asic_
2 and(2), ior(2) 11 /home/otski/work/tce-1.0/tce/hdb/asic_
2 add(2), sub(2) 10 /home/otski/work/tce-1.0/tce/hdb/asic_
2 add(2), sub(2) 9 /home/otski/work/tce-1.0/tce/hdb/asic_
1 add(1), sub(1) 8 /home/otski/work/tce-1.0/tce/hdb/asic_
1 add(1), sub(1) 7 /home/otski/work/tce-1.0/tce/hdb/asic_
2 add(2) 6 /home/otski/work/tce-1.0/tce/hdb/asic_
2 add(2) 5 /home/otski/work/tce-1.0/tce/hdb/asic_
1 add(1) 4 /home/otski/work/tce-1.0/tce/hdb/asic_
1 add(1) 3 /home/otski/work/tce-1.0/tce/hdb/asic_
1 sxhw(1) 2 /home/otski/work/tce-1.0/tce/hdb/asic_
1 sxhw(1) 1 /home/otski/work/tce-1.0/tce/hdb/asic_
2 reflect(2) 1 /home/otski/intro/intro.hdb
6 ldh(6), ldhu(6), ldq(6), ldqu(... 2 /home/otski/intro/avalon.hdb
1..3 ldw(3), ldq(3), ldh(3), stw(1),... 1 /home/otski/intro/avalon.hdb
Add
Close
signal inst_fetch_fetch_en_wire :
signal inst_fetch_glock_wire : std
signal inst_fetch_fetchblock_wire
signal fu_lsu_t1data_wire : std_lo
signal fu_lsu_t1load_wire : std_lo
signal fu_lsu_o1data_wire : std_lo
signal fu_lsu_o1load_wire : std_lo
signal fu_lsu_r1data_wire : std_lo
signal fu_lsu_t1opcode_wire : std_logic_vector(2 downto 0);
signal fu_lsu_glock_wire : std_logic;
signal fu_logic_t1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1load_wire : std_logic;
signal fu_logic_o1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_o1load_wire : std_logic;
signal fu_logic_r1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1opcode_wire : std_logic_vector(1 downto 0);
signal fu_logic_glock_wire : std_logic;
signal fu_reflect_t1data_wire : std_logic_vector(31 downto 0);
Ln 5, Col 19
INS


File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
toplevel.vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
TTA Processor Designer - minimal.adf
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win Options Help
minimal.adf
FU: ext1 { sxhw }
FU: ext2 { sxqw }
FU: shifter { shl, shr, shru }
FU: rotator { rotl, rotr }
FU: reflect { reflect }
FU: avalon_mm_int { ldh, ldhu, ldq, ldqu, ldw, sth, stq...
RF: RF 10x32
signal inst_fetch_fetch_en_wire : std_logic;
signal inst_fetch_glock_wire : std_logic;
signal inst_fetch_fetchblock_wire : std_logic_vector(IMEMWIDTHINMAUS*IMEMMAUWIDTH-1 downto 0);
signal fu_lsu_t1data_wire : std_logic_vector(23 downto 0);
signal fu_lsu_t1load_wire : std_logic;
signal fu_lsu_o1data_wire : std_logic_vector(31 downto 0);
signal fu_lsu_o1load_wire : std_logic;
signal fu_lsu_r1data_wire : std_logic_vector(31 downto 0);
signal fu_lsu_t1opcode_wire : std_logic_vector(2 downto 0);
signal fu_lsu_glock_wire : std_logic;
signal fu_logic_t1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1load_wire : std_logic;
signal fu_logic_o1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_o1load_wire : std_logic;
signal fu_logic_r1data_wire : std_logic_vector(31 downto 0);
signal fu_logic_t1opcode_wire : std_logic_vector(1 downto 0);
signal fu_logic_glock_wire : std_logic;
signal fu_reflect_t1data_wire : std_logic_vector(31 downto 0);
Ln 5, Col 19
INS

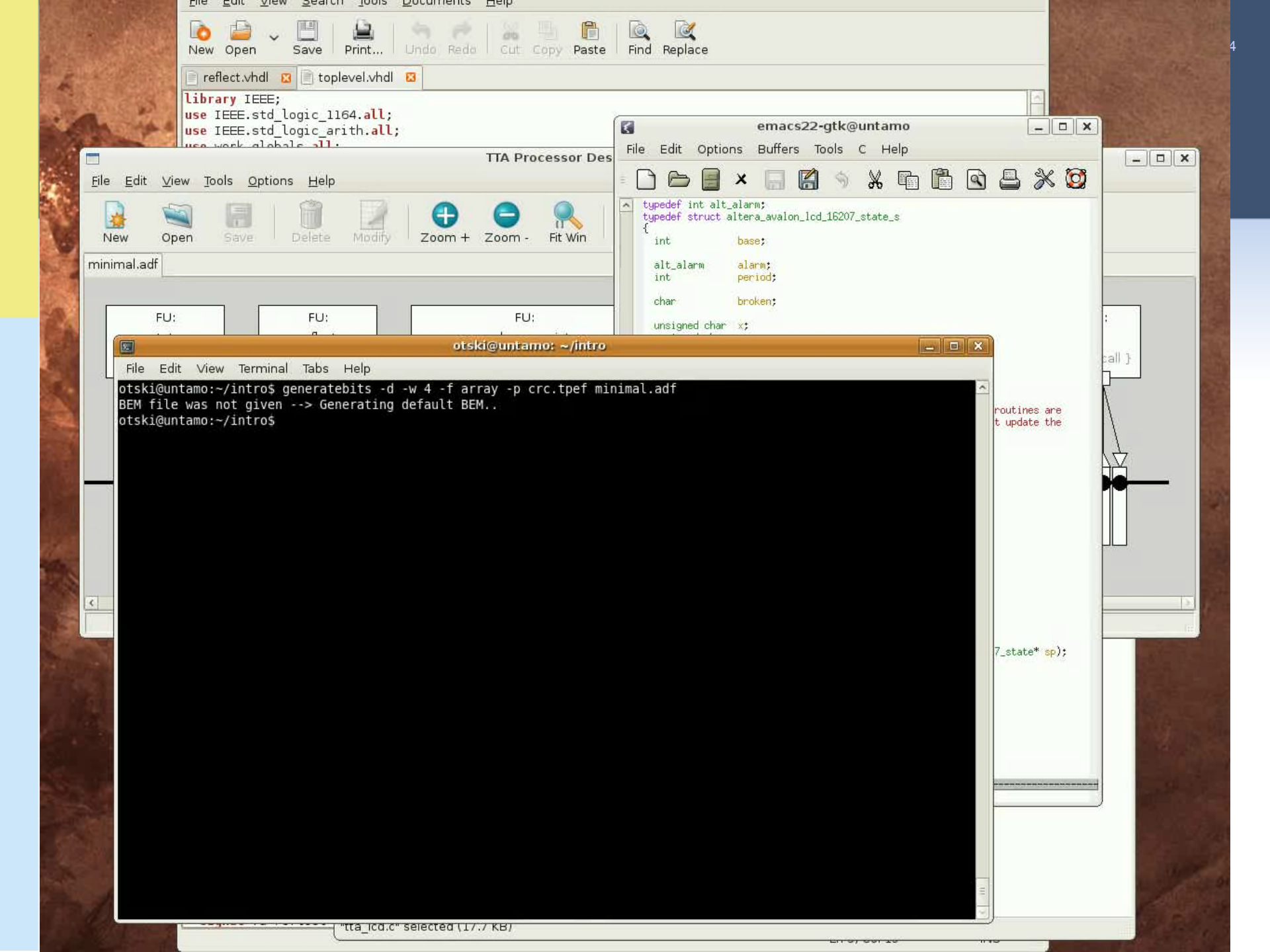
Using Avalon LCD for Output (21 secs)

- Now the TTA can interface with the memory (and other I/O) in the FPGA board using the Avalon bus
- Finally, we need a device to produce some output from our CRC computation to verify it actually works
- For this we use an LCD screen connected through the Avalon bus
  - We use the LCD controller from SOPC Builder's IP component library
- The LCD controller is connected to the Avalon Memory Mapped bus interface so we can define a putchar() function (which is used by printf()) that writes characters to the controller's memory mapped registers


emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
* TCE version modified by Otto Esko
* Notes: Only crc-32 is used
#ifdef _DEBUG
#include <stdio.h>
#endif /* _DEBUG */
#include "crc.h"
#include "lcd_driver/tta_lcd.h"
volatile crc result = 0;
unsigned char test[] = "TCE rocks!";
#define LENGTH 10
altera_avalon_lcd_16207_state lcd;
/* define our own putchar */
#undefine putchar
int putchar(int ch) {
lcd_write_data(&lcd, ch);
}
int
main(void) {
/* init lcd */
altera_avalon_lcd_16207_init(&lcd);
/*
* Compute the CRC of the test message, more efficiently.
*/
result = crcFast(test, LENGTH);
#ifdef _DEBUG
printf("TCE rocks! is\n 0x%X\n", result);
#endif /* _DEBUG */
return 0;
}
--(DOS)** main.c Bot (35,26) (C/l Abbrev)
emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
typedef int alt_alarm;
typedef struct altera_avalon_lcd_16207_state_s
lcd_driver - File Browser
Help
Reload
Home
Computer
Search
100%
Icon View
main.c
minimal.adf
minimal.bem
prog.tpef
prog_data.img
tta_lcd.c
tta_sys.c
tta_sys.h
"tta_lcd.c" selected (17.7 KB)
signal fu_lsu_r1da
signal fu_lsu_t1op
signal fu_lsu_gloc
signal fu_logic_t1
signal fu_logic_t1
signal fu_logic_o1
signal fu_logic_o1
signal fu_logic_r1
signal fu_logic_t1
signal fu_logic_gl
signal fu_reflect

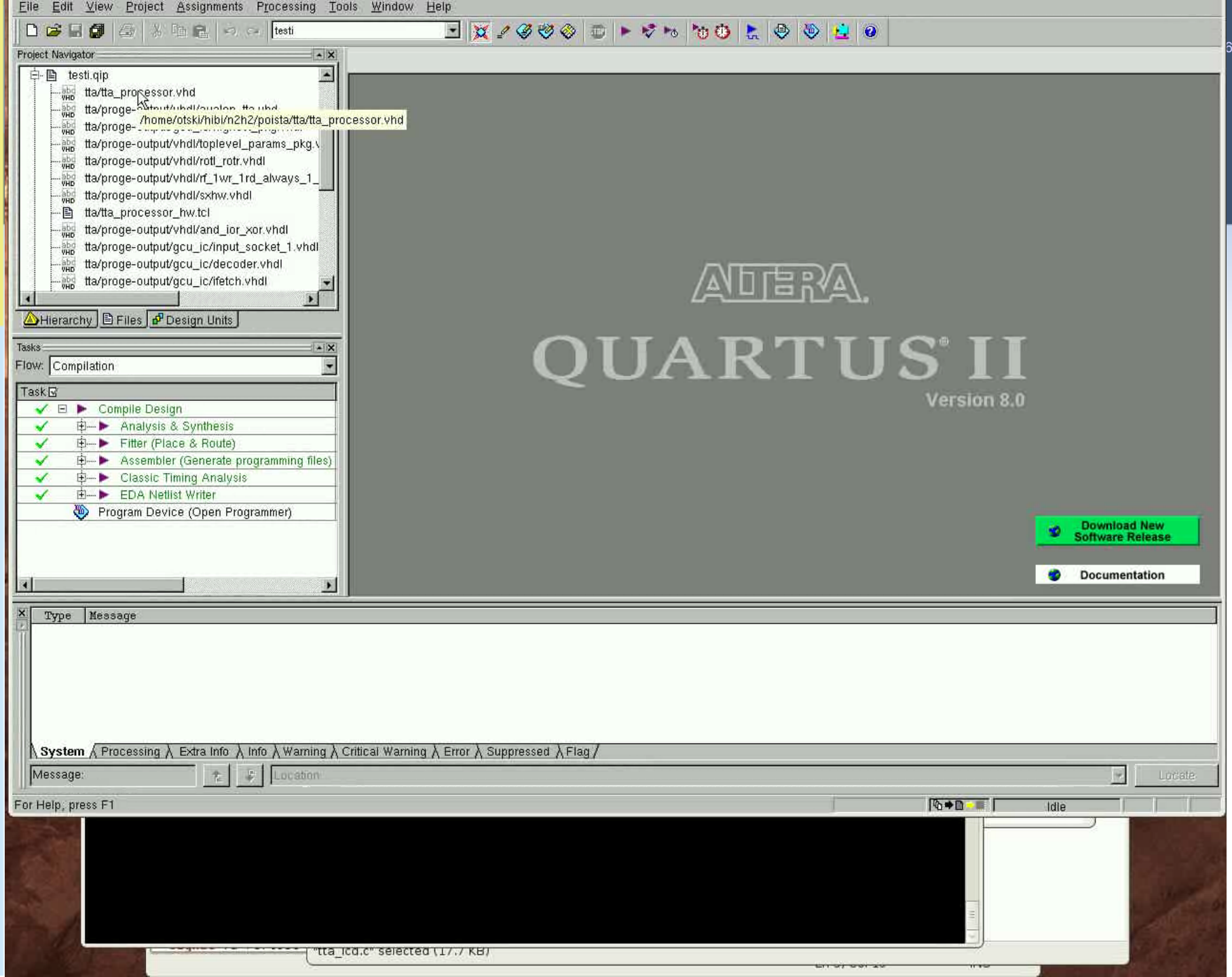
Generate the Bit Image of the Program Memory and Synthesize the Design (2 minutes)

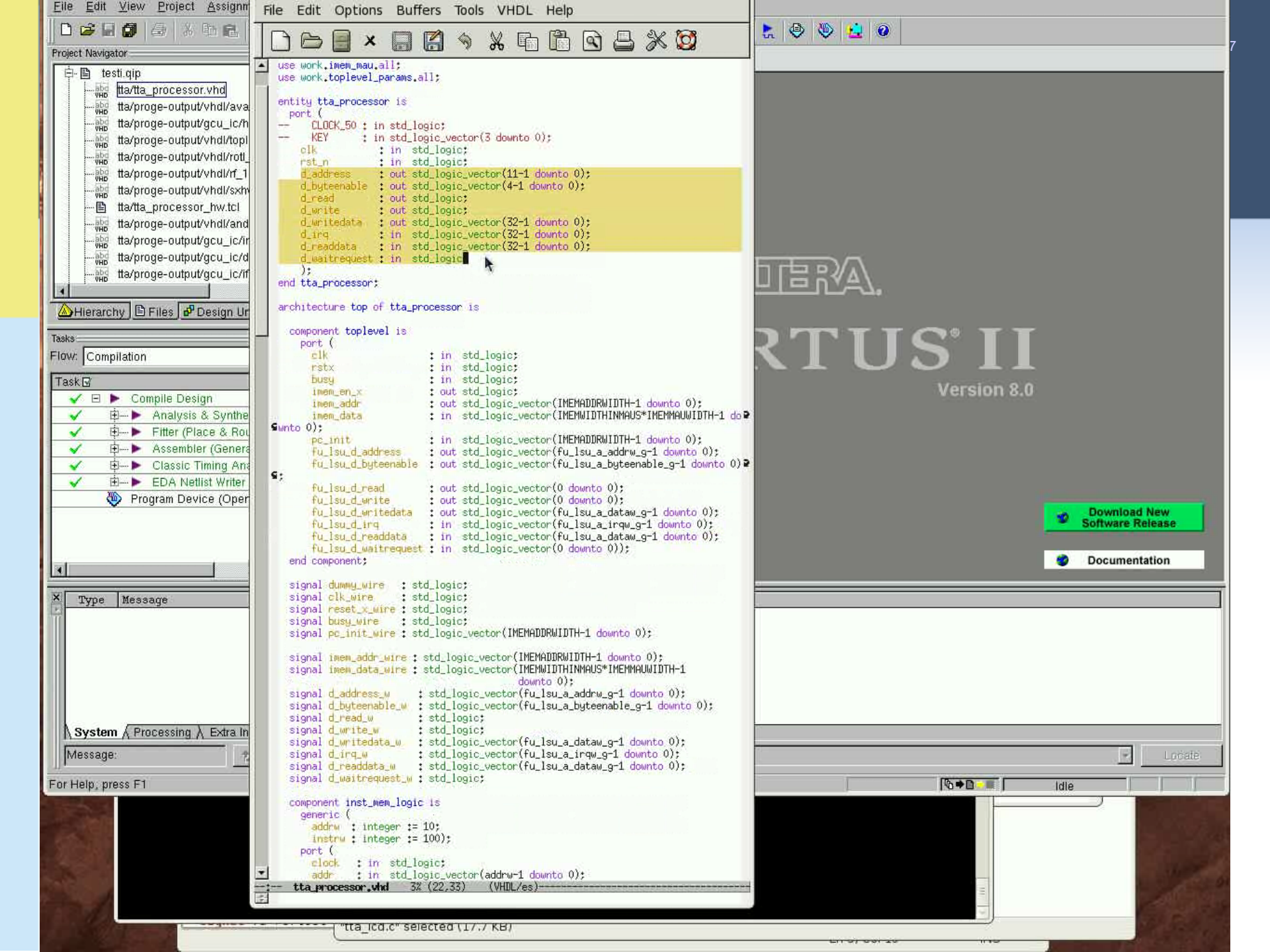
- Finally, to get the TTA running on the FPGA we need to generate a bit image of the program memory
  - Use command line tool generatebits

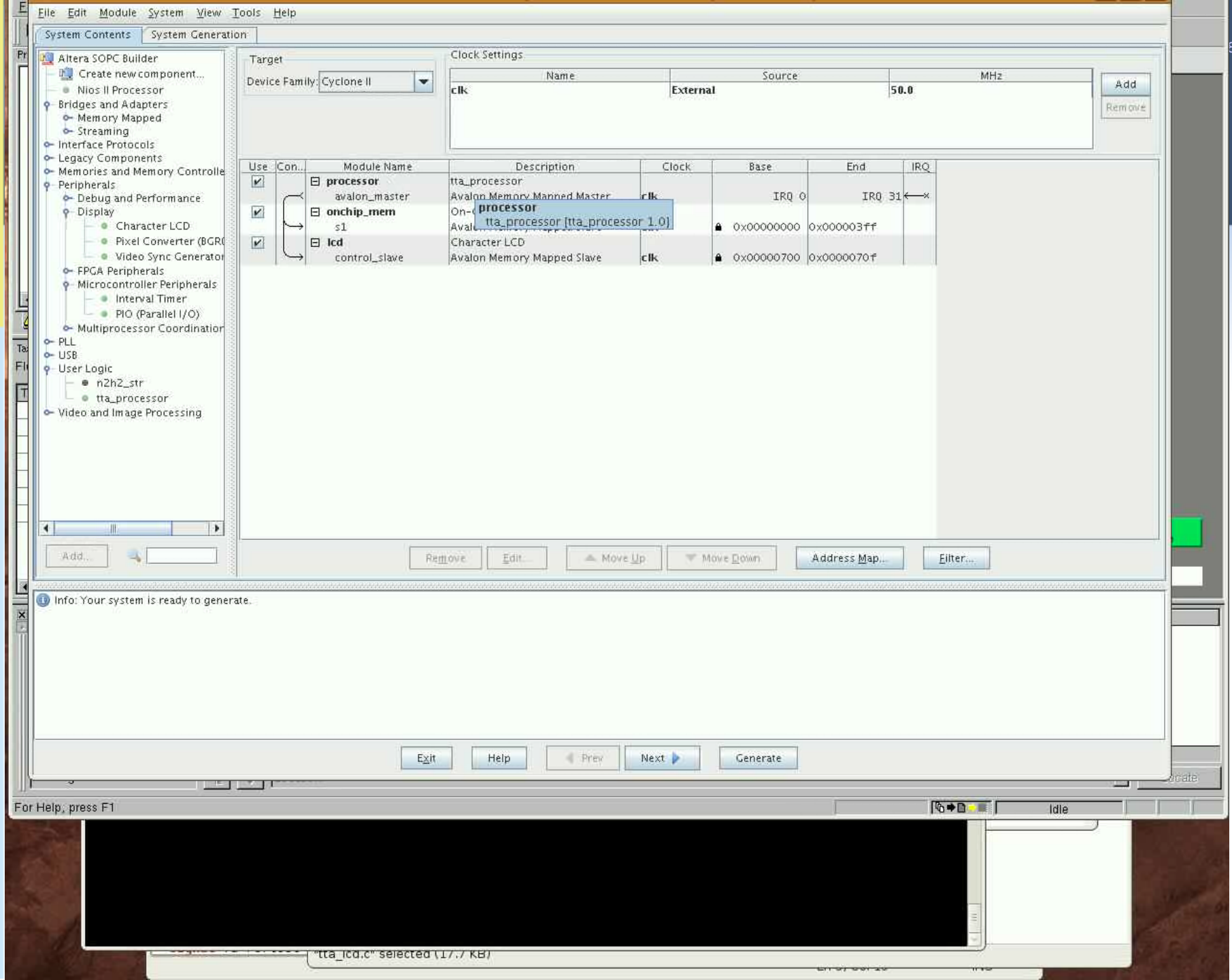

File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
reflect.vhdl
toplevel.vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
TTA Processor Des
File Edit View Tools Options Help
New Open Save Delete Modify Zoom + Zoom - Fit Win
minimal.adf
FU:
FU:
FU:
emacs22-gtk@untamo
File Edit Options Buffers Tools C Help
typedef int alt_alarm;
typedef struct altera_avalon_lcd_16207_state_s
{
int base;
alt_alarm alarm;
int period;
char broken;
unsigned char x;
otski@untamo: ~/intro
File Edit View Terminal Tabs Help
otski@untamo:~/intro$ generatebits -d -w 4 -f array -p crc.tpef minimal.adf
BEM file was not given --> Generating default BEM..
otski@untamo:~/intro$
"tta_lcd.c" selected (17.7 KB)

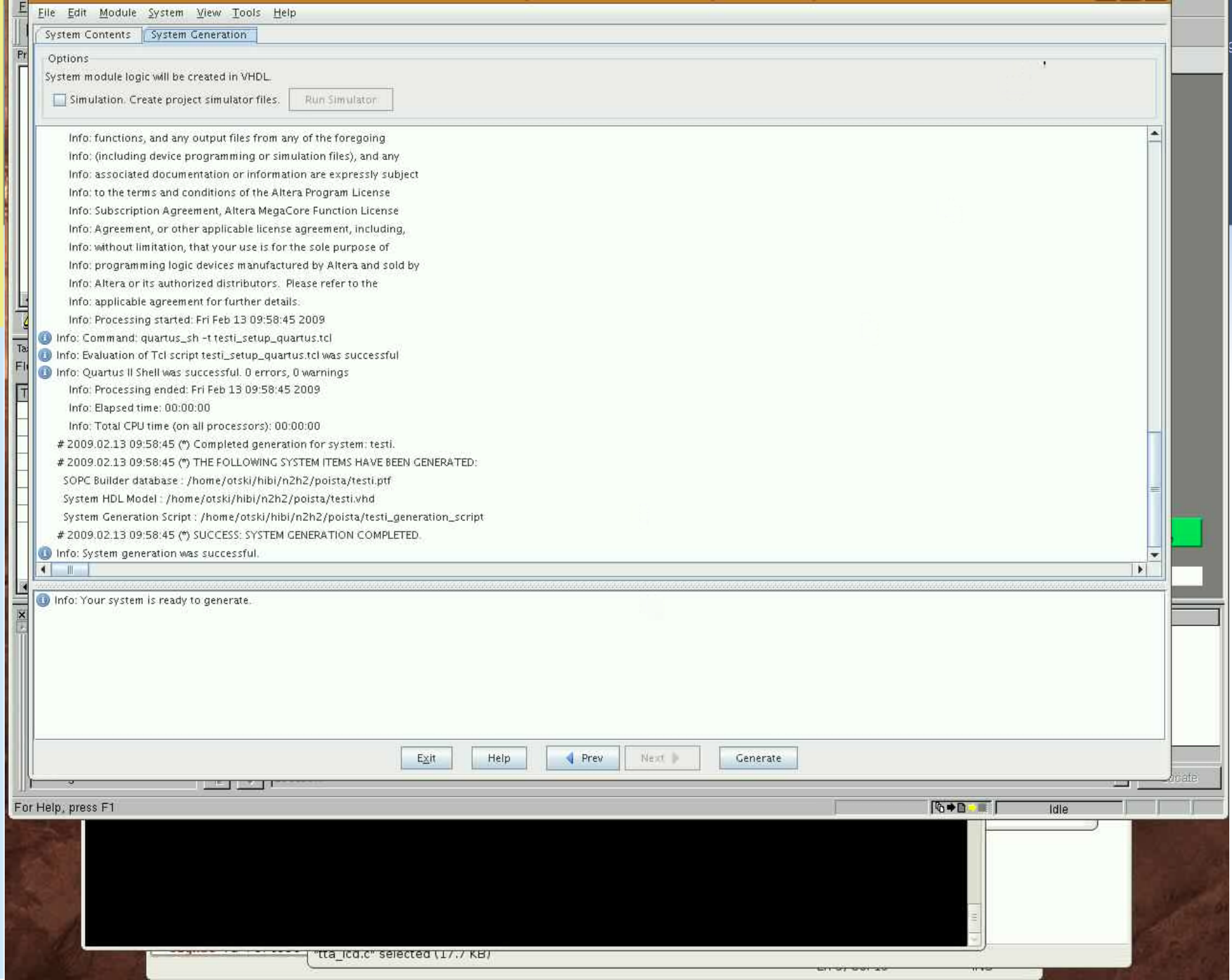
Generate the Bit Image of the Program Memory and Synthesize the Design

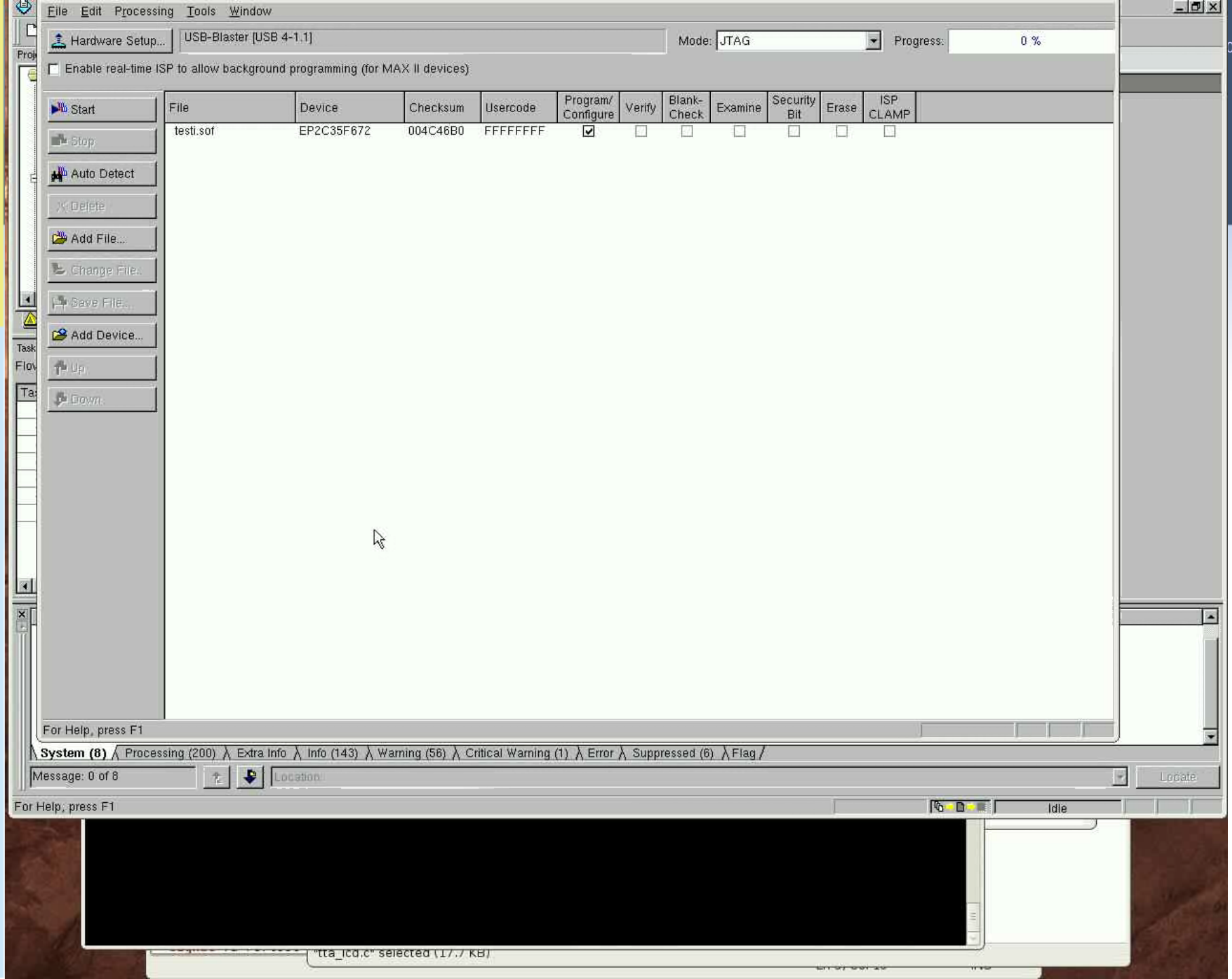
- Load the VHDL files of the generated TTA processor to the Altera's Quartus II tool to synthesize the design to the FPGA
  - We add a layer on top of the ProGe generated toplevel.vhdl (not displayed in the video)
    - In this case the instruction memory is very small so we implement as “logic”
    - Synthesize tools optimizes it into a small space of internal memory banks and logic
  - The external interface of the new layer is the external buses of the load-store unit we added, and control signals (clk, reset)
    - The LSU interface is actually the Avalon interface
  - Then in Altera's SOPC builder we export TTA as a component to the design along with the onchip memory and the LCD component and connect them all to the Avalon bus
    - TTA is the Avalon Master and the memory and LCD controller are slaves
  - Synthesize the design to the FPGA and note how many of the logic elements were consumed of the FPGA by our TTA
  - Finally, upload the design to the FPGA board


File Edit View Project Assignments Processing Tools Window Help
testi
Project Navigator
testi.qip
tta/tta_processor.vhd
/home/otski/hibi/n2h2/poista/tta/tta_processor.vhd
tta/proge-output/vhdl/toplevel_params_pkg.v
tta/proge-output/vhdl/rotl_rotr.vhdl
tta/proge-output/vhdl/rf_1wr_1rd_always_1_
tta/proge-output/vhdl/sxhw.vhdl
tta/tta_processor_hw.tcl
tta/proge-output/vhdl/and_ior_xor.vhdl
tta/proge-output/gcu_ic/input_socket_1.vhdl
tta/proge-output/gcu_ic/decoder.vhdl
tta/proge-output/gcu_ic/ifetch.vhdl
Hierarchy
Files
Design Units
Tasks
Flow: Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
Classic Timing Analysis
EDA Netlist Writer
Program Device (Open Programmer)
ALTERA
QUARTUS II
Version 8.0
Download New Software Release
Documentation
Type
Message
System
Processing
Extra Info
Info
Warning
Critical Warning
Error
Suppressed
Flag
Message:
Location
Locate
For Help, press F1
Idle


File Edit View Project Assignm
Project Navigator
testi.qip
tta/tta_processor.vhd
tta/proge-output/vhdl/ava
tta/proge-output/gcu_ic/h
tta/proge-output/vhdl/topl
tta/proge-output/vhdl/rotl_
tta/proge-output/vhdl/rf_1
tta/proge-output/vhdl/sxh
tta/tta_processor_hw.tcl
tta/proge-output/vhdl/and
tta/proge-output/gcu_ic/ir
tta/proge-output/gcu_ic/d
tta/proge-output/gcu_ic/if
Hierarchy
Files
Design Ur
Tasks
Flow: Compilation
Task
Compile Design
Analysis & Synthe
Fitter (Place & Rou
Assembler (Genera
Classic Timing Ana
EDA Netlist Writer
Program Device (Oper
Type
Message
System
Processing
Extra In
Message:
For Help, press F1
File Edit Options Buffers Tools VHDL Help
use work.imem_mau.all;
use work.toplevel_params.all;
entity tta_processor is
port (
-- CLOCK_50 : in std_logic;
-- KEY : in std_logic_vector(3 downto 0);
clk : in std_logic;
rst_n : in std_logic;
d_address : out std_logic_vector(11-1 downto 0);
d_byteenable : out std_logic_vector(4-1 downto 0);
d_read : out std_logic;
d_write : out std_logic;
d_writedata : out std_logic_vector(32-1 downto 0);
d_irq : in std_logic_vector(32-1 downto 0);
d_readdata : in std_logic_vector(32-1 downto 0);
d_waitrequest : in std_logic
);
end tta_processor;
architecture top of tta_processor is
component toplevel is
port (
clk : in std_logic;
rstx : in std_logic;
busy : in std_logic;
imem_en_x : out std_logic;
imem_addr : out std_logic_vector(IMEMADDRWIDTH-1 downto 0);
imem_data : in std_logic_vector(IMEMWIDTHINMAUS*IMEMMAUWIDTH-1 do
wnto 0);
pc_init : in std_logic_vector(IMEMADDRWIDTH-1 downto 0);
fu_lsu_d_address : out std_logic_vector(fu_lsu_a_addrw_g-1 downto 0);
fu_lsu_d_byteenable : out std_logic_vector(fu_lsu_a_byteenable_g-1 downto 0)
;
fu_lsu_d_read : out std_logic_vector(0 downto 0);
fu_lsu_d_write : out std_logic_vector(0 downto 0);
fu_lsu_d_writedata : out std_logic_vector(fu_lsu_a_dataw_g-1 downto 0);
fu_lsu_d_irq : in std_logic_vector(fu_lsu_a_irqw_g-1 downto 0);
fu_lsu_d_readdata : in std_logic_vector(fu_lsu_a_dataw_g-1 downto 0);
fu_lsu_d_waitrequest : in std_logic_vector(0 downto 0));
end component;
signal dummy_wire : std_logic;
signal clk_wire : std_logic;
signal reset_x_wire : std_logic;
signal busy_wire : std_logic;
signal pc_init_wire : std_logic_vector(IMEMADDRWIDTH-1 downto 0);
signal imem_addr_wire : std_logic_vector(IMEMADDRWIDTH-1 downto 0);
signal imem_data_wire : std_logic_vector(IMEMWIDTHINMAUS*IMEMMAUWIDTH-1
downto 0);
signal d_address_w : std_logic_vector(fu_lsu_a_addrw_g-1 downto 0);
signal d_byteenable_w : std_logic_vector(fu_lsu_a_byteenable_g-1 downto 0);
signal d_read_w : std_logic;
signal d_write_w : std_logic;
signal d_writedata_w : std_logic_vector(fu_lsu_a_dataw_g-1 downto 0);
signal d_irq_w : std_logic_vector(fu_lsu_a_irqw_g-1 downto 0);
signal d_readdata_w : std_logic_vector(fu_lsu_a_dataw_g-1 downto 0);
signal d_waitrequest_w : std_logic;
component inst_mem_logic is
generic (
addrw : integer := 10;
instrw : integer := 100);
port (
clock : in std_logic;
addr : in std_logic_vector(addrw-1 downto 0);
--:-- tta_processor.vhd 3% (22,33) (VHDL/es)
RTUS II
Version 8.0
Download New Software Release
Documentation
Locate
Idle
"tta_icd.c" selected (17.7 KB)


File Edit Module System View Tools Help
System Contents
System Generation
Altera SOPC Builder
Create new component...
Nios II Processor
Bridges and Adapters
Memory Mapped
Streaming
Interface Protocols
Legacy Components
Memories and Memory Controlle
Peripherals
Debug and Performance
Display
Character LCD
Pixel Converter (BGR
Video Sync Generator
FPGA Peripherals
Microcontroller Peripherals
Interval Timer
PIO (Parallel I/O)
Multiprocessor Coordinatior
PLL
USB
User Logic
n2h2_str
tta_processor
Video and Image Processing
Target
Device Family: Cyclone II
Clock Settings
Name
Source
MHz
clk
External
50.0
Add
Remove
Use
Con...
Module Name
Description
Clock
Base
End
IRQ
processor
tta_processor
avalon_master
Avalon Memory Mapped Master
clk
IRQ 0
IRQ 31
onchip_mem
On-
s1
0x00000000
0x000003ff
lcd
Character LCD
control_slave
Avalon Memory Mapped Slave
clk
0x00000700
0x0000070f
processor
tta_processor [tta_processor 1.0]
Add...
Remove
Edit...
Move Up
Move Down
Address Map...
Filter...
Info: Your system is ready to generate.
Exit
Help
Prev
Next
Generate
For Help, press F1
Idle
"tta_lcd.c" selected (17.7 KB)


File Edit Module System View Tools Help
System Contents
System Generation
Options
System module logic will be created in VHDL.
Simulation. Create project simulator files.
Run Simulator
Info: functions, and any output files from any of the foregoing
Info: (including device programming or simulation files), and any
Info: associated documentation or information are expressly subject
Info: to the terms and conditions of the Altera Program License
Info: Subscription Agreement, Altera MegaCore Function License
Info: Agreement, or other applicable license agreement, including,
Info: without limitation, that your use is for the sole purpose of
Info: programming logic devices manufactured by Altera and sold by
Info: Altera or its authorized distributors. Please refer to the
Info: applicable agreement for further details.
Info: Processing started: Fri Feb 13 09:58:45 2009
Info: Command: quartus_sh -t testi_setup_quartus.tcl
Info: Evaluation of Tcl script testi_setup_quartus.tcl was successful
Info: Quartus II Shell was successful. 0 errors, 0 warnings
Info: Processing ended: Fri Feb 13 09:58:45 2009
Info: Elapsed time: 00:00:00
Info: Total CPU time (on all processors): 00:00:00
2009.02.13 09:58:45 (*) Completed generation for system: testi.
2009.02.13 09:58:45 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED:
SOPC Builder database : /home/otski/hibi/n2h2/poista/testi.ptf
System HDL Model : /home/otski/hibi/n2h2/poista/testi.vhd
System Generation Script : /home/otski/hibi/n2h2/poista/testi_generation_script
2009.02.13 09:58:45 (*) SUCCESS: SYSTEM GENERATION COMPLETED.
Info: System generation was successful.
Info: Your system is ready to generate.
Exit
Help
Prev
Next
Generate
For Help, press F1
Idle
"tta_icd.c" selected (17.7 KB)


File Edit Processing Tools Window
Hardware Setup...
USB-Blaster [USB 4-1.1]
Mode: JTAG
Progress: 0 %
Enable real-time ISP to allow background programming (for MAX II devices)
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File...
Add Device...
Up
Down
File
Device
Checksum
Usercode
Program/ Configure
Verify
Blank-Check
Examine
Security Bit
Erase
ISP CLAMP
testi.sof
EP2C35F672
004C46B0
FFFFFFFF
For Help, press F1
System (8)
Processing (200)
Extra Info
Info (143)
Warning (56)
Critical Warning (1)
Error
Suppressed (6)
Flag
Message: 0 of 8
Location
Locate
For Help, press F1
Idle

The End – Thanks for Your Attention!