- **EXPLANATION**:
    a. All changes related to the new instruction are tagged with comment: *// BEQC* in the soft-core. We must perform the following actions:
        - Annul the DS when the branch is not taken: There is a specific signal (***annul_ds_i***) which annuls the DS. For Branch Likely instructions, the DS is annulled when the condition for the branch is not met (!***mpc_eqcond_e***). For Branch Compact instructions, the DS is annulled when the condition for the branch is met (***mpc_eqcond_e***). Note that we only change the functionality of the BEQL instruction. Change:

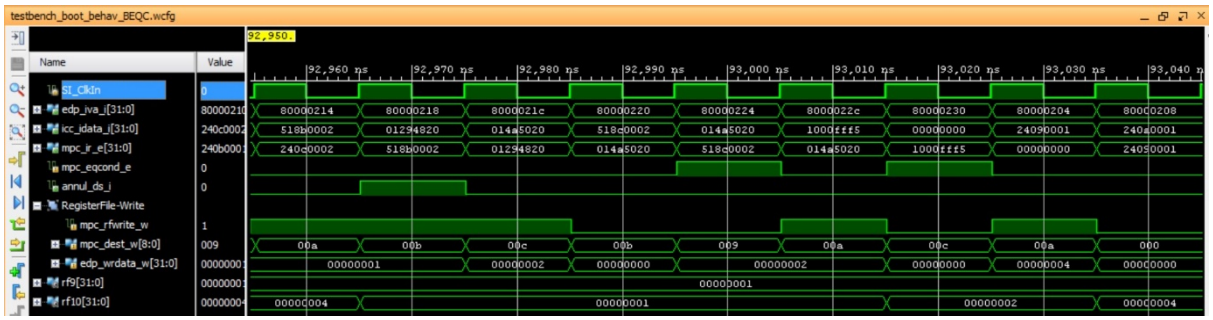
assign annul_ds_i = (br_likely_e && mpc_irval_e && !mpc_eqcond_e) ||

For:

assign annul_ds_i = ((br_likely_e && mpc_ir_e[27:26]!=2'b00) && mpc_irval_e && !mpc_eqcond_e) || // BranchCompact
((br_likely_e && mpc_ir_e[27:26]==2'b00) && mpc_irval_e && mpc_eqcond_e) || // BranchCompact

- **EXAMPLE - SIMULATION**:

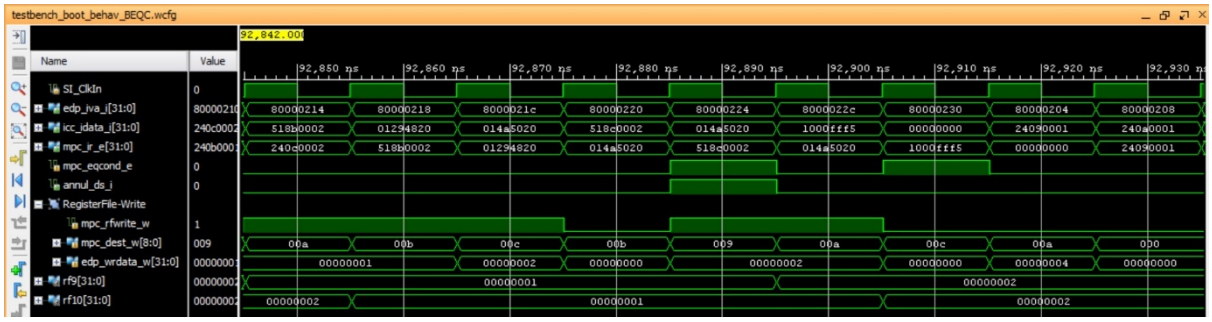
Results of the simulation on the original core (BEQL instruction):



In the second cycle, the first beql is executed. The condition is not met (mpc_eqcond_e=0), thus the branch is not taken. In this case, the instruction in the branch delay slot must not be executed (annul_ds_i=1). Observe that, in cycle 5, the register file is not written.
In the fifth cycle, the second beql is executed. The condition is met (mpc_eqcond_e=1), thus the branch is taken. In this case, the instruction in the branch delay slot must be executed (annul_ds_i=0). Observe that, in the eighth cycle, the register file is written with the result of the instruction in the delay slot.
The result in registers $t1 and $t2 is:

- $t1(rf9)=1
- $t2(rf10)=4

Results of the simulation on the modified core (BEQC instruction):



In the second cycle, the first beql is executed. The condition is not met (mpc_eqcond_e=0), thus the branch is not taken. In this case, differently to the previous case, the instruction in the branch delay slot must be executed (annul_ds_i=0). Observe that, in cycle 5, the register file is written.
In the fifth cycle, the second beql is executed. The condition is met (mpc_eqcond_e=1), thus the branch is taken. In this case, differently to the previous case, the instruction in the branch delay slot must not be executed (annul_ds_i=1). Observe that, in the eighth cycle, the register file is not written.
The result in registers $t1 and $t2 is:

- $t1(rf9)=2
- $t2(rf10)=2

- **<u>EXECUTION ON BOARD</u>**:

Results of the execution on the original core (BEQL instruction):

When the program is downloaded on the board, you should see on the 7-seg displays:

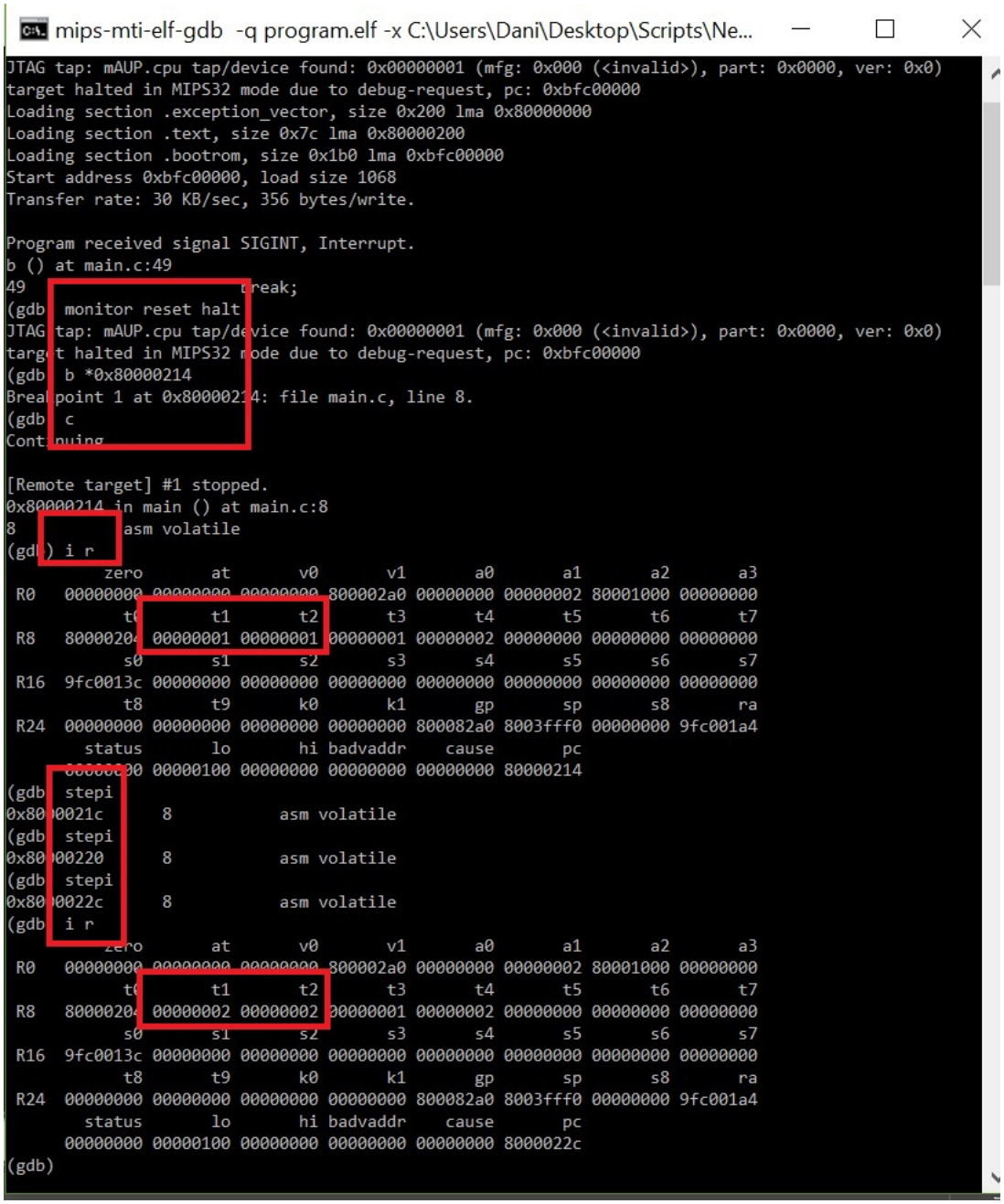
- Switches=0 → 7-seg displays=$t1, which in our example is 0x1
- Switches=1 → 7-seg displays=$t2, which in our example is 0x4

Results of the execution on the modified core (BEQC instruction):

When the program is downloaded on the board, you should see on the 7-seg displays:

- Switches=0 → 7-seg displays=$t1, which in our example is 0x2
- Switches=1 → 7-seg displays=$t2, which in our example is 0x2

Debug of the modified core:


mips-mti-elf-gdb -q program.elf -x C:\Users\Dani\Desktop\Scripts\Ne...
JTAG tap: mAUP.cpu tap/device found: 0x00000001 (mfg: 0x000 (<invalid>), part: 0x0000, ver: 0x0)
target halted in MIPS32 mode due to debug-request, pc: 0xbfc00000
Loading section .exception_vector, size 0x200 lma 0x80000000
Loading section .text, size 0x7c lma 0x80000200
Loading section .bootrom, size 0x1b0 lma 0xbfc00000
Start address 0xbfc00000, load size 1068
Transfer rate: 30 KB/sec, 356 bytes/write.
Program received signal SIGINT, Interrupt.
b () at main.c:49
49 break;
(gdb) monitor reset halt
JTAG tap: mAUP.cpu tap/device found: 0x00000001 (mfg: 0x000 (<invalid>), part: 0x0000, ver: 0x0)
target halted in MIPS32 mode due to debug-request, pc: 0xbfc00000
(gdb) b *0x80000214
Breakpoint 1 at 0x80000214: file main.c, line 8.
(gdb) c
Continuing.
[Remote target] #1 stopped.
0x80000214 in main () at main.c:8
8 asm volatile
(gdb) i r
zero at v0 v1 a0 a1 a2 a3
R0 00000000 00000000 00000000 800002a0 00000000 00000002 80001000 00000000
t0 t1 t2 t3 t4 t5 t6 t7
R8 80000204 00000001 00000001 00000001 00000002 00000000 00000000 00000000
s0 s1 s2 s3 s4 s5 s6 s7
R16 9fc0013c 00000000 00000000 00000000 00000000 00000000 00000000 00000000
t8 t9 k0 k1 gp sp s8 ra
R24 00000000 00000000 00000000 00000000 800082a0 8003fff0 00000000 9fc001a4
status lo hi badvaddr cause pc
00000000 00000100 00000000 00000000 00000000 80000214
(gdb) stepi
0x80000220 8 asm volatile
(gdb) stepi
0x80000220 8 asm volatile
(gdb) stepi
0x8000022c 8 asm volatile
(gdb) i r
zero at v0 v1 a0 a1 a2 a3
R0 00000000 00000000 00000000 800002a0 00000000 00000002 80001000 00000000
t0 t1 t2 t3 t4 t5 t6 t7
R8 80000204 00000002 00000002 00000001 00000002 00000000 00000000 00000000
s0 s1 s2 s3 s4 s5 s6 s7
R16 9fc0013c 00000000 00000000 00000000 00000000 00000000 00000000 00000000
t8 t9 k0 k1 gp sp s8 ra
R24 00000000 00000000 00000000 00000000 800082a0 8003fff0 00000000 9fc001a4
status lo hi badvaddr cause pc
00000000 00000100 00000000 00000000 00000000 8000022c
(gdb)