Disseny i síntesis de sistemes digitals

Grau d'Enginyeria Electrónica de Telecomunicacions

Gianfranco Bazzani Valldeperez

Data de realització: 16/04/2019

Comunicació amb FPGA durant runtime amb vJTAG

---

1. Introducció.

Aquest document reflexa com utilitzar la IP virtualJTAG d'altera i crear una solució software per a monitorar, actualitzar i depurar dissenys durant runtime utilitzant el protocol JTAG, evitant la necessitat d'utilitzar pins del dispositiu per a aquesta finalitat. Aquest mètode permet crear també algoritmes de test i depuració ràpids i eficaços, podent realitzar test amb scripts, evitant així haver de testejar i depurar amb LEDs i switches. Per realitzar aquests scripts Altera te disponible una API Tcl completa per enviar i rebre transaccions el dispositiu durant l'execució. Per a realitzar els exemples he utilitzat una placa de desenvolupament DE0-nano i un ordenador Windows 10 amb el software Quartus II 14.1.

2. Descripció del mètode.

Afegir el IP vJTAG al nostre disseny

El primer pas es instanciar en el nostre disseny el IP core virtual JTAG d'altera, per a això tenim una eina en el Quartus. Un cop tenim en nostre projecte obert ens dirigim **Tools > IP Catalog**. A continuació se'ns obrirà una finestra anomenada IP Catalog, cerquem a la barra de cerca, virtual JTAG i hi fem doble click.

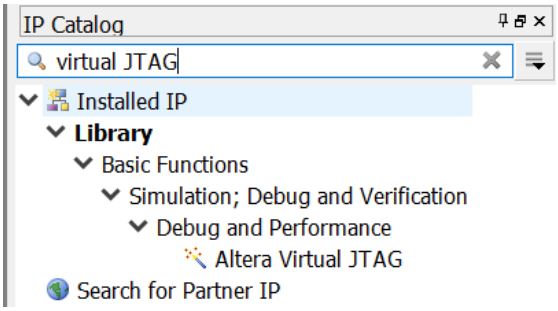


Figura 1 IP Catalog al Quartus II 14.1.

A continuació se'ns obri l'eina IP Parameter editor ja amb la IP virtual JTAG pre-carregada, l'únic que hem de fer a continuació és assignar la mida del registre d'instruccions i generar el HDL en la destinació desitjada. La mida del registre d'instruccions pot variar en funció del nombre d'instruccions que volem configurar. Aquesta eina ens genera un seguit de fitxers, el següent pas és crear un símbol per a afegir a un esquemàtic el fitcher .v que ens a generat. Fem notar també que podem implementar varis mòduls vJTAG en el mateix projecte per a cridar-los mes tard, cada un d'aquests tindrà un identificador diferent. Notifico que aquest pas pot variar en funció de la versió de Quartus II utilitzada ja que en versions anterior no implementen el IP Catalog però tenim present una altra eina anomenada megafunction Wizard amb la qual podem obtenir els mateixos resultats seguint un procediment similar.

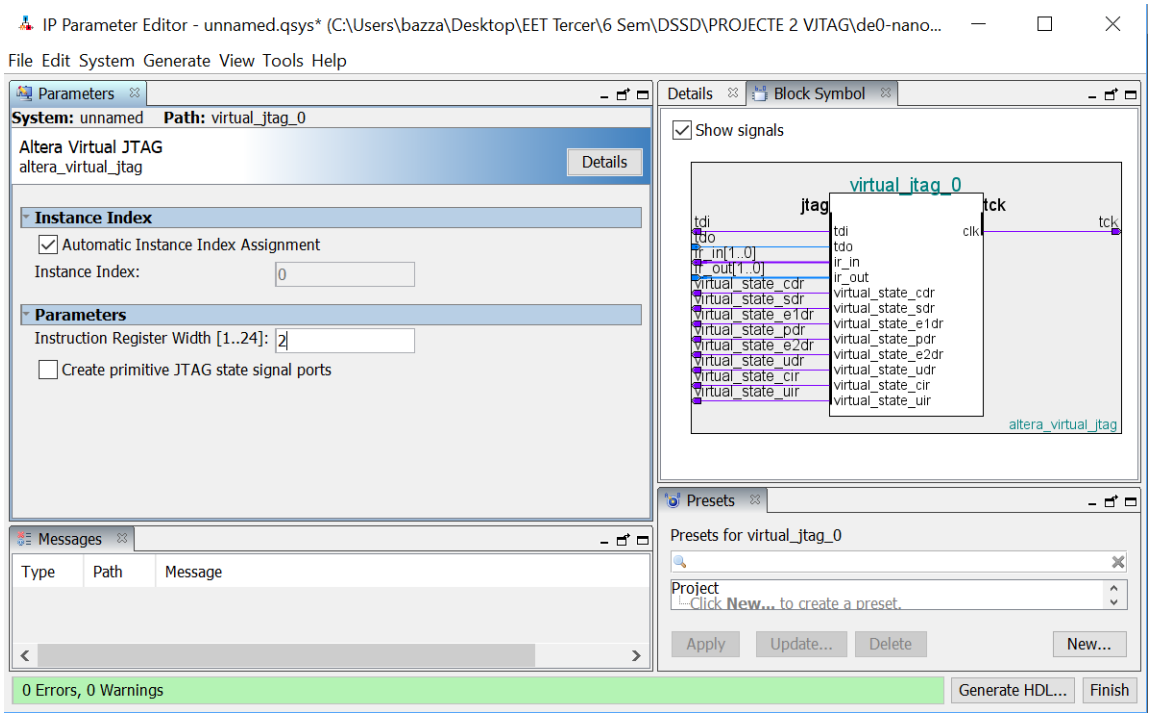


Figura 2 Eina Ip Parameter amb Ip vJTAG.

Integrar el mòdul vJTAG

El següent pas es dissenyar una interfície per a que interactuï amb el nostre projecte, la funció d'aquesta es interpretar el senyals del mòdul vJTAG i gestionar el registres de dades. A l'hora de dissenyar aquesta interfície hem de tenir en compte les instruccions que necessitem, ja que aquestes les podrem cridar a través del registre d'instruccions. També hem de tenir en compte que hem d'implementar el desplaçament del registre de dades i assignar en aquest les dades que volem extreure o introduir al nostre disseny en cada instrucció. Podem utilitzar les banderes d'estats per a realitzar accions en estats desitjats.

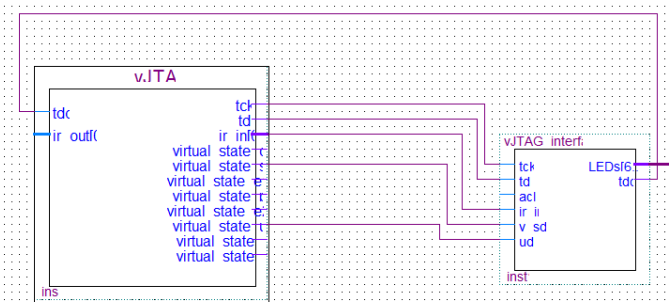


Figura 3 Exemple de mòdul que interactua amb el nostre disseny i la IP vJTAG.

Dissenyar interfície amb el pc

El següent pas es dissenyar un programa en llenguatge tcl(Tool Command Language) adaptat a les nostres necessitats. Per a interactuar amb el mòdul vJTAG implementat en el nostre disseny altera ens posa a disposició una API per a llenguatge tcl, aquesta la podem combinar amb la seva extensió tk (Tool Kit) per a crear GUI's més userfriendly.

La API Tcl per al IP core vJTAG consisteix en un conjunt de comandes per a accedir als registres d'instrucció i de dades de cada una de les instàncies vJTAG. A continuació mostro una taula proporcionada per Altera en la guia d'usuari del IP core vJTAG que conté les comandes tcl per a atacar al executable quartus_stp.

Command	Arguments	Description
Device virtual ir shift	-instance_index <*instance_index*> -ir_value <*numeric_ir_value*> -no_captured_ir_value(1) -show_equivalent_device_ir_dr_shift(1)	Perform an IR shift operation to the virtual JTAG instance specified by the instance_index. Note that ir_value takes a numerical argument.
Device virtual dr shift	-instance_index <*instance_index*> -dr_value <*dr_value*> -length <*data_register_length*> -no_captured_dr_value(1) -show_equivalent_device_ir_dr_shift -value_in_hex(1)	Perform a DR shift operation to the virtual JTAG instance.
Get hardware names	NONE	Queries for all available programming cables.
Open device	-device_name <*device_name*> -hardware_name <*hardware_name*>	Selects the active device on the JTAG chain.
Close device	NONE	Ends communication with the active JTAG device.
Device lock	-timeout <*timeout*>	Obtains exclusive communication to the JTAG chain.
Device unlock	NONE	Releases device_lock.
Device ir shift	-ir_value <*ir_value*> -no_captured_ir_value	Performs a IR shift operation.
Device dr shift	-dr_value <*dr_value*> -length <*data register length*> -no_captured_dr_value -value_in_hex	Performs a DR shift operation.

Figura 4 Comandes de la API tcl per a IP core vJTAG.

Aquestes comandes ens permeten realitzar un script que interactuï amb la IP vJTAG. Per a realitzar qualsevol acció sobre la ip, en primer lloc hem de identificar el dispositiu al qual volem establir la comunicació. A continuació mostro un procediment exemple que obté el id del cable de programació i selecciona el primer dispositiu connectat amb aquest. Els identificadors dels dispositius es guarden en variables globals per a poder-los utilitzar en altres procediments.

```
proc connect_jtag {} {
    global usbblaster_name
    global test_device
    foreach hardware_name [get_hardware_names] {
        if { [string match "USB-Blaster*" $hardware_name] } {
            set usbblaster_name $hardware_name
        }
    }

    foreach device_name [get_device_names -hardware_name $usbblaster_name] {
        if { [string match "@1*" $device_name] } {
            set test_device $device_name
        }
    }
}
```

Figura 5 Procediment exemple per a obtenir els id del hardware.

A continuació mostro dos procediments exemple per a establir la comunicació i per a tancar-la, aquest s'executen sempre abans de començar a enviar instruccions a la IP vJTAG i inicien o tanquen la comunicació amb els dispositius obtinguts anteriorment.

```
proc open_port {} {
    global usbblaster_name
    global test_device
    open_device -hardware_name $usbblaster_name -device_name $test_device
}

proc close_port {} {
    catch {device_unlock}
    catch {close_device}
}
```

Figura 6 Procediments exemple per establir comunicació i tancar-la.

Un cop obrim una connexió amb la IP vJTAG(això ho fem cridant el proc open_port), podem realitzar desplaçaments de registres de instruccions. A continuació mostro un exemple per a realitzar un desplaçament del registre d'instruccions, d'aquesta manera la interfície JTAG pot conèixer que es el que volem fer en la FPGA. Amb el paràmetre -instance_index "N" seleccionem la instatació de la IP vJTAG a la qual volem atacar, amb el paràmetre -ir_value "N" fixem el valor del registre d'instruccions abans de desplaçar-lo, -no_captured_ir_value permet la comunicació nomes en direcció PC->FPGA d'aquesta manera la comunicació es més rapida.

```
device_virtual_ir_shift -instance_index 0 -ir_value 3 -no_captured_ir_value
```

Figura 7 Exemple implementació de desplaçament de registre d'instruccions.

Després de desplaçar un valor de registre d'instrucció la interfície JTAG a la FPGA ja sabrà com administrar les dades del registre de dades, ara podem realitzar un desplaçament del registre de dades. A continuació mostro dos exemples per a realitzar desplaçament de dades. Amb el paràmetre -dr_value "N" assignem el valor del registre d'instruccions que volem desplaçar aquest s'assigna en el estat CAPURE_DR, amb el paràmetre -length "N" ajustem la longitud dels bits desplaçats.

```
set dip [device_virtual_dr_shift -dr_value 00000000 -instance_index 0 -length 8]
```

Figura 8 Exemple implementació de desplaçament de registre de dades amb lectura de dades retornades.

```
device_virtual_dr_shift -dr_value 00000000 -instance_index 0 -length 8 -no_captured_dr_value
```

Figura 9 Exemple implementació de desplaçament de registre de dades sense lectura de dades retornades.

Ara utilitzant aquest procediments i les comandes proporcionades per Altera podem construir un script per executar en el fitxer quartus_stp. Un cop tenim el fitxer creat el script hem de crear un accés directe que ataqui al fitxer quartus_stp amb el script creat, per a fer això utilitzarem el paràmetre -t.

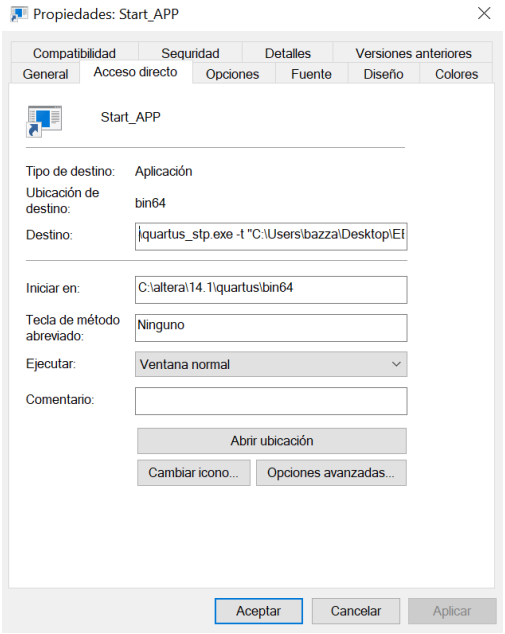


Figura 10 Accés directe per a executar el script.

Combinant en el script la extensió de tcl, tk podem crear interfícies gràfiques per a monitoritzar i controlar els nostres dissenys.

3. Exemples

1.GUI per a controlar la de0-Nano

El primer exemple és una interfície gràfica per a controlar els LEDs i llegir els switches de la de0-Nano.

Codi RTL de la interfície vJTAG:

```
module connect(
   tck, tdi, aclr, ir_in, v_sdr, v_udr, v_cdr, v_uir,
   s1, s2, s3, s4,
   d0, d1, d2, d3, d4, d5, d6, d7, tdo
   );
//paràmetres per a instruccions
localparam BYPASS = 2'b00;
localparam DIP    = 2'b01;
localparam LED    = 2'b10;

input tck, tdi, aclr, v_sdr, v_udr, v_cdr, v_uir;
input [1:0]ir_in;
input s1, s2, s3, s4;
output wire  tdo, d0, d1, d2, d3, d4, d5, d6, d7;

reg [1:0]DR0;
reg [7:0]DR1; //registre de dades
reg [7:0]out = 8'b00000000; //registre per a la sortida

assign tdo = (ir_in == BYPASS) ? DR0[0] : DR1[0];

//assignacó wires de sortida
assign d0 = out[0];
assign d1 = out[1];
assign d2 = out[2];
assign d3 = out[3];
assign d4 = out[4];
assign d5 = out[5];
assign d6 = out[6];
assign d7 = out[7];

always @ (posedge tck)
begin
   if(!aclr) begin //clear
      DR0 <= 1'b0;
      DR1 <= 8'b00000000;
   end
   else begin
      case(ir_in) //case per a instruccions
         DIP: begin
                  if(v_cdr) begin
                     DR1 = {4'b0000,s4,s3,s2,s1};
                  end
                  else
                  begin
                     if(v_sdr) begin
                        DR1 = {tdi,DR1[7:1]}; //shift valor dels switches
                     end
                  end
               end

         LED: begin
                  if(v_sdr) begin
                     DR1 = {tdi,DR1[7:1]}; //shift valors dels leds
                  end
               end

         BYPASS: begin
                     if(v_sdr) begin
                           DR0 = {tdi,DR0[1:1]}; //shift 0
                     end
                  end


         default: begin
                     if(v_sdr) begin
                        DR0 = {tdi,DR0[1:1]}; //shift 0
                     end
                  end

      endcase
   end
end
//always per a assignar els LEDs sensible al estat update data register
always @ (v_udr)
begin
   if(ir_in == LED) begin
      out = DR1;
   end
end

endmodule
```

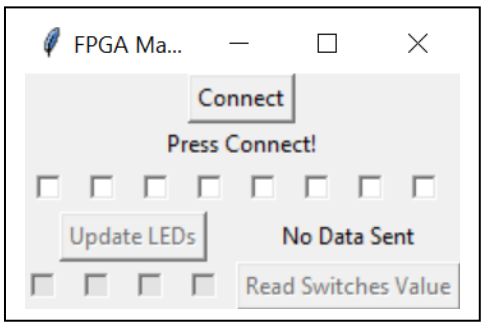
Analysis & Synthesis Summary	
Analysis & Synthesis Status	Successful - T
Quartus II 64-Bit Version	14.1.0 Build 1
Revision Name	jtagConnect
Top-level Entity Name	schematic
Family	Cyclone IV E
Total logic elements	162
Total combinational functions	146
Dedicated logic registers	88
Total registers	88
Total pins	13
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0

Codi GUI tlc/tk:

```
proc send_data {} {
    global d0 d1 d2 d3 d4 d5 d6 d7 displayData
    set led ""
    set one 1
    set zero 0

    if {$d0 == 1} {set led $led$one} else {set led $led$zero}
    if {$d1 == 1} {set led $led$one} else {set led $led$zero}
    if {$d2 == 1} {set led $led$one} else {set led $led$zero}
    if {$d3 == 1} {set led $led$one} else {set led $led$zero}
    if {$d4 == 1} {set led $led$one} else {set led $led$zero}
    if {$d5 == 1} {set led $led$one} else {set led $led$zero}
    if {$d6 == 1} {set led $led$one} else {set led $led$zero}
    if {$d7 == 1} {set led $led$one} else {set led $led$zero}

    set displayData "Data sent: $led"

    open_port
    device_lock -timeout 10000
    device_virtual_ir_shift -instance_index 0 -ir_value 2 -no_captured_ir_value
    set l [device_virtual_dr_shift -dr_value $led -instance_index 0  -length 8]
    puts $l
    device_virtual_ir_shift -instance_index 0 -ir_value 0 -no_captured_ir_value
    close_port
}

proc read_switch {} {
    open_port
    device_lock -timeout 10000
    device_virtual_ir_shift -instance_index 0 -ir_value 1 -no_captured_ir_value
    set dip [device_virtual_dr_shift -dr_value 0000 -instance_index 0 -length 4]

    device_virtual_ir_shift -instance_index 0 -ir_value 0 -no_captured_ir_value
    close_port

    if {[string index $dip 0] == 1} {.chks0 select} else {.chks0 deselect}
    if {[string index $dip 1] == 1} {.chks1 select} else {.chks1 deselect}
    if {[string index $dip 2] == 1} {.chks2 select} else {.chks2 deselect}
    if {[string index $dip 3] == 1} {.chks3 select} else {.chks3 deselect}

}


global usbblaster_name
global test_device

set displayData "No Data Sent"
set  displayConnect "Press Connect!"

package require Tk
init_tk

wm state . normal
wm title . "FPGA Manager"
frame .frmConnection
label .lblConn -textvariable displayConnect
button .btnConn -text "Connect" -command "connect_jtag"

frame .frmData
checkbutton .chk0 -variable d0
checkbutton .chk1 -variable d1
checkbutton .chk2 -variable d2
checkbutton .chk3 -variable d3
checkbutton .chk4 -variable d4
checkbutton .chk5 -variable d5
checkbutton .chk6 -variable d6
checkbutton .chk7 -variable d7
button .btnSend -text "Update LEDs" -command "send_data"
label .lblData -textvariable displayData


frame .frmSwitch
checkbutton .chks0
checkbutton .chks1
checkbutton .chks2
checkbutton .chks3
button .btnRead -text "Read Switches Value" -command "read_switch"

grid .frmConnection -in .  -row 1 -column 1 -columnspan 8
grid .btnConn -in .frmConnection -row 1 -column 1
grid .lblConn -in .frmConnection -row 2 -column 1
```

```
grid .frmData -in .  -row 2 -column 1
grid .chk0 -in .frmData -row 1 -column 1
grid .chk1 -in .frmData -row 1 -column 2
grid .chk2 -in .frmData -row 1 -column 3
grid .chk3 -in .frmData -row 1 -column 4
grid .chk4 -in .frmData -row 1 -column 5
grid .chk5 -in .frmData -row 1 -column 6
grid .chk6 -in .frmData -row 1 -column 7
grid .chk7 -in .frmData -row 1 -column 8
grid .btnSend -in .frmData -row 2 -column 1 -columnspan 4
grid .lblData -in .frmData -row 2 -column 5 -columnspan 4

grid .frmSwitch -in .  -row 3 -column 1
grid .chks0 -in .frmSwitch -row 1 -column 1
grid .chks1 -in .frmSwitch -row 1 -column 2
grid .chks2 -in .frmSwitch -row 1 -column 3
grid .chks3 -in .frmSwitch -row 1 -column 4
grid .btnRead -in .frmSwitch -row 1 -column 5 -columnspan 4

.btnSend configure -state disabled
.btnRead configure -state disabled

.chks0 configure -state disabled
.chks1 configure -state disabled
.chks2 configure -state disabled
.chks3 configure -state disabled

tkwait window .
```

GUI:



2.Control d'un comptado en la de0-Nano

El primer exemple és una interfície gràfica per a controlar amb botons un comptador implementat amb els leds de De0-Nano.

Codi RTL de la interfície vJTAG:

```
module connect(
   tck, tdi, aclr, ir_in, v_sdr, v_udr, v_cdr, v_uir,
   d0, d1, d2, d3, d4, d5, d6, d7, tdo
   );
//paràmetres per a instruccions
localparam BYPASS    = 2'b00;
localparam READCOUNT = 2'b01;
localparam COUNT     = 2'b10;
localparam RESCOUNT  = 2'b11;

input tck, tdi, aclr, v_sdr, v_udr, v_cdr, v_uir;
input [1:0]ir_in;
output wire  tdo, d0, d1, d2, d3, d4, d5, d6, d7;

reg [7:0]Counter = 1'b0000;//registre per al comptador
reg [1:0]DR0;
reg [7:0]DR1;//registre de dades

assign tdo = (ir_in == BYPASS) ? DR0[0] : DR1[0];
//assignació wires de sortida
assign d0 = Counter[0];
assign d1 = Counter[1];
assign d2 = Counter[2];
assign d3 = Counter[3];
assign d4 = Counter[4];
assign d5 = Counter[5];
assign d6 = Counter[6];
assign d7 = Counter[7];

always @ (posedge tck)
begin
   if(!aclr) begin //clear
      DR0 <= 1'b0;
      DR1 <= 8'b00000000;
   end
   else begin
      case(ir_in) //case per a instruccions
         READCOUNT: begin
                  if(v_cdr) begin
                     DR1 = Counter;
                  end
                  else
                  begin
                     if(v_sdr) begin
                        DR1 = {tdi,DR1[7:1]};//shift valor del contador
                     end
                  end
               end

         COUNT: begin
                  if(v_cdr) begin
                     Counter=Counter+1;
                     DR1 = Counter;
                  end
                  else
                  begin
                     if(v_sdr) begin
                        DR1 = {tdi,DR1[7:1]};//shift valor del contador
                     end
                  end
               end


         RESCOUNT: begin
                  if(v_cdr) begin
                     Counter=0;
                     DR1 = Counter;
                  end
                  else
                  begin
                     if(v_sdr) begin
                        DR1 = {tdi,DR1[7:1]};//shift valor del contador
                     end
                  end
               end

         BYPASS: begin
                  if(v_sdr) begin
                        DR0 = {tdi,DR0[1:1]};//shift 0
                     end
                  end

         default: begin
                  if(v_sdr) begin
                     DR0 = {tdi,DR0[1:1]};//shift 0
                     end
                  end

      endcase
   end
end


endmodule
```

Analysis & Synthesis Summary	
Analysis & Synthesis Status	Successful - F
Quartus II 64-Bit Version	14.1.0 Build
Revision Name	jtagConnect
Top-level Entity Name	schematic
Family	Cyclone IV E
Total logic elements	171
Total combinational functions	155
Dedicated logic registers	96
Total registers	96
Total pins	9
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0

Codi GUI tlc/tk:

```
proc read_Counter {} {
    open_port
    device_lock -timeout 10000
    device_virtual_ir_shift -instance_index 0 -ir_value 1 -no_captured_ir_value
    set dip [device_virtual_dr_shift -dr_value 00000000 -instance_index 0 -length 8]

    device_virtual_ir_shift -instance_index 0 -ir_value 0 -no_captured_ir_value
    close_port

    puts $dip
}

proc more_count {} {
    open_port
    device_lock -timeout 10000
    device_virtual_ir_shift -instance_index 0 -ir_value 2 -no_captured_ir_value
    set mc [device_virtual_dr_shift -dr_value 00000000 -instance_index 0 -length 8]
    device_virtual_ir_shift -instance_index 0 -ir_value 0 -no_captured_ir_value
    close_port
    puts $mc

}

proc res_count {} {
    open_port
    device_lock -timeout 10000
    device_virtual_ir_shift -instance_index 0 -ir_value 3 -no_captured_ir_value
    set rc [device_virtual_dr_shift -dr_value 00000000 -instance_index 0 -length 8]
    device_virtual_ir_shift -instance_index 0 -ir_value 0 -no_captured_ir_value
    close_port
    puts $rc
}

global usbblaster_name
global test_device

set displayData "No Data Sent"
set  displayConnect "Press Connect!"

package require Tk
init_tk

wm state . normal
wm title . "FPGA Manager"
frame .frmConnection
label .lblConn -textvariable displayConnect
button .btnConn -text "Connect" -command "connect_jtag"

frame .frmCounter
button .btnCount -text "Count++" -command "more_count"
button .btnRescount -text "ResetCounter" -command "res_count"
button .btnRead -text "Read Counter" -command "read_Counter"




grid .frmConnection -in .  -row 1 -column 1 -columnspan 8
grid .btnConn -in .frmConnection -row 1 -column 1
grid .lblConn -in .frmConnection -row 2 -column 1

grid .frmCounter -in . -row 4 -column 1
grid .btnCount -in .frmCounter -row 1 -column 1
grid .btnRescount -in .frmCounter -row 1 -column 2

grid .frmCounter -in . -row 4 -column 1
grid .btnCount -in .frmCounter -row 1 -column 1
grid .btnRescount -in .frmCounter -row 1 -column 2
grid .btnRead -in .frmCounter -row 1 -column 3

.btnCount configure -state disabled
.btnRescount configure -state disabled
.btnRead configure -state disabled


tkwait window .
```

GUI:

Webgrafia

Documentació JTAG:

https://www.fpga4fun.com/JTAG.html

Projecte Base:

https://github.com/xharrym/tk_jtag

Manual vJTAG Altera:

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_virtualjtag.pdf