Introduction to HDL

Group 7

Group Members

- EE23BTECH11049 - Praful Kesavadas
- EE23BTECH11044 - Nimal Sreekumar
- EE23BTECH11048 - Venkata Chanakya
- EE23BTECH11210 - Dhyana Teja
- ME23BTECH11002 - Abhijith Raj

Absolute value block

The code takes in 3 inputs of 32 bits each and then returns the absolute values of the same.
The triggering for trojan is given as an input. The output wave form obtained are as follows:

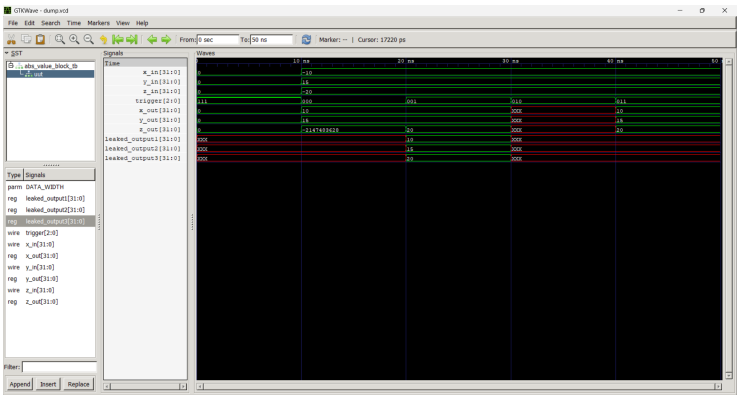

Figure 1: Gtkwave output of the block

The trojan function is as follows:

- Trigger = 0 : For changing the functionality of the code
  Here we can see that the output for the two inputs remain unchanged, but the third output is different from the expected output. Thus the functionality of the code has been changed.

- Trigger =1 : For leaking the output
  In this case, there is a set of 3 extra outputs which is defined as the leaked outputs. When this trigger is enabled, these leaked outputs will contain the actual outputs from the function. On the other hand, whenever it is not triggered, the leaked outputs is undefined.

- Trigger =2 : For denying the service
  Whenever this trojan is activated all the outputs are undefined which causes the block to be useless

- Trigger = any other value(3-7): Default case
  The block functions as usual, where the output is the absolute value of the input and the leaked output is undefined.

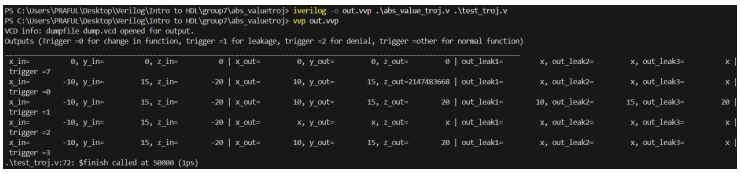

Figure 2: Terminal output for absolute value block

As we can see, when the trigger is 0, the zout is some high number where the expected value is 20
Similarly when trigger is 1, the leaked output is same as the expected outputs
For trigger=2, all the outputs are undefined
For trigger =3 or above, the block is functioning as usual.

PE.v

The code has two 16-bit inputs, image, and kernal. Under normal compilation the code does convolution, i.e., the output is the product of image and kernel added to the previous output.

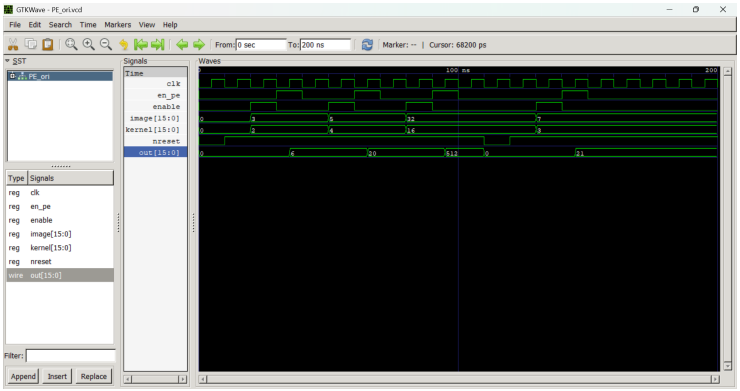

Figure 3: Gtkwave output of the block

Here is what the trojan does:

- Trigger = 0: Changes the functionality of the code.
- Trigger = 1: Leaks the inputs and gives normal output as the function without the interruption of trojan.
- Trigger = 2: Denial of Service (DoS) Trojan. This gives nothing as output(/invalid output).
- DEFAULT: Any value other than 0,1,2 will give normal operation of that of convoltion.

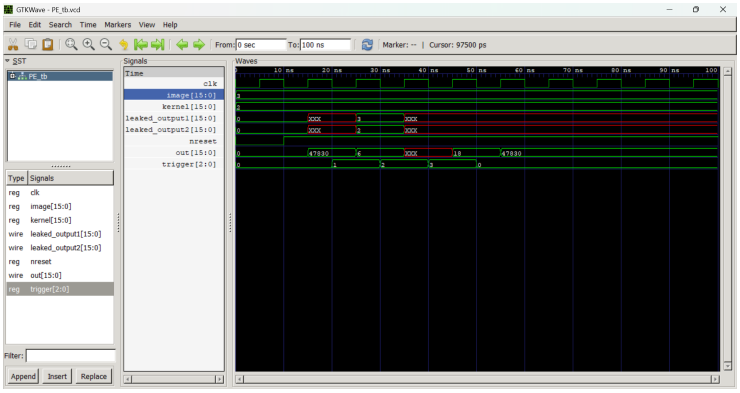

Figure 4: Trojan output of the block

Rbram

The code takes in a 16 bit input data and an address. When the write is enabled it stores the input data and while it is disabled outputs the value.

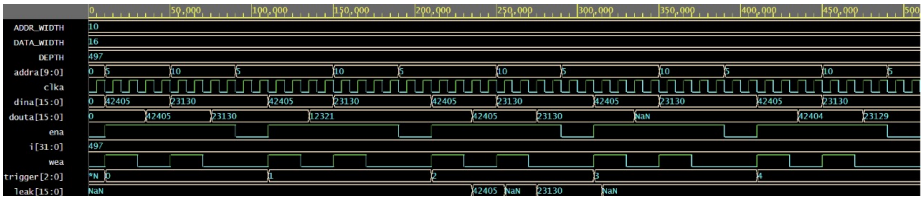

Figure 5: Gtkwave output of the block

- Trigger = 1 : For changing the functionality of the code
  The initial functionality is inhibited by storing the number 2 instead of the actual input. This makes the code output 2 regardless of the input.
- Trigger = 2 : For leaking the output
  When this trigger is enabled, these leaked outputs will contain the actual outputs from the function. On the other hand, whenever it is not triggered, the leaked outputs is undefined.
- Trigger = 3 : For denying the service
  Whenever this trojan is activated all the outputs are undefined which causes the block to be useless
- Trigger = 4 : For degrading performance
  The performance is degraded by subtracting the number one from the input before storing it.

vector rot enable

This code takes two inputs clock and reset with two outputs cordic-vec-enable,cordic-Rot-enable the functionality of code begins when reset is one
If reset is zero count,cordic-vec-enable,cordic-rot-enable all are assigned to zero
If reset is one the count starts incrementing counting till 17 and then to zero
When count is incrementing, cordic-vec-enable is assigned to one
After one clock cycle cordic-vec-enable is also assigned to one
After adding trojan new variable is introduced, trigger

If trigger is 0 the code works as original
If trigger is 1 the code will get delayed here it doesn't matter much as it depends only on clock
If trigger is two the code doesn't give any output
If trigger is three the code functionality itself changes for odd number in count the cordics behave in different way

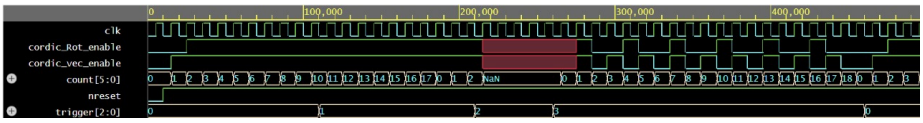

Figure 6: Output with trojan

vector divider block

- The main functionality of this code is like it contains three inputs x_{in}, y_{in}, z_{in} which are 38 bits. and also a rotation indicator(micro-rot-o) which is initiated to 1 at the beginning.
- we take one of the inputs i.e is z_{in} as reference i.e is x_{ref}.
- we have two cases mentioned in this code When `X_ref` is less than `x_in`, the following operations are performed:
  - `x_temp_out` is updated by adding a right-shifted version of `y_in`, ensuring sign extension for proper arithmetic.
  - `y_temp_out` is updated by subtracting a right-shifted version of `x_in`, again ensuring sign extension.
  - `z_temp_out` remains unchanged.
  - The signal `micro_rot_o` is set to 0, indicating that no significant micro-rotation occurred in this step.

  This logic is part of a CORDIC vectoring operation, which iteratively refines `x` and `y` values for applications such as division or vector rotation.
- Other case is opposite of this.
- Trojan i inserted contains three cases :
- Case 1 : trigger 0 it functions normally.
- Case 2: trigger 1 it leaks the data
- Case 3: trigger 2 it Denies the output
- Case 4: trigger 3 it changes the functionality of the code . The change of the functionality is exactly opposite the original function . mean if now the $x_{in} > X_{ref}$ the opposite thing happens.

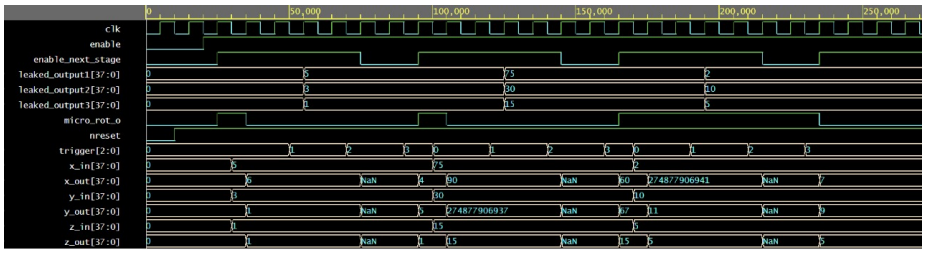

Figure 7: Output waveform