Registers Description

NAME: trigger\_type

DESCRIPTION: determine trigger type (rise, fall, '1', '0'), two triggers can not be chosen at the same time, only one can be active and the other will be passive.

ADDRESS: "0x00"

ACCESS: R\W

RESET VALUE: 0x01

|  |  |
| --- | --- |
| BIT | DESCRIPTION |
| TD[0] | ‘1’ when the chosen trigger is rise. (default trigger) |
| TD[1] | ‘1’ when the chosen trigger is fall. |
| TD[2] | ‘1’ when the chosen trigger is one. (needs to be '1' for 3 clock cycles) |
| TD[3] | ‘1’ when the chosen trigger is zero. (needs to be '0' for 3 clock cycles) |
| TD[4] | ‘1’ when the chosen trigger is a special trigger (second project). |
| TD[5..7] | NA |

NAME: trigger\_position

DESCRIPTION: determine which percent of the recorded data will appear before and after the trigger.

ADDRESS: "0x01"

ACCESS: R\W

RESET VALUE: 0x00

|  |  |
| --- | --- |
| BIT | DESCRIPTION |
| TD[0..6] | The percent in decimal of the recorded data that will taken **before** the trigger rise. (default: 0- all the data will be recorded after trigger rise) |
| TD[7] | NA |

NAME: clk\_to\_start

DESCRIPTION: debugging register.

ADDRESS: "0x02"

ACCESS: R

RESET VALUE: 0x00

|  |  |
| --- | --- |
| BIT | DESCRIPTION |
| TD[0..7] | Counts the number of clk cycles that passed since the system started working until trigger raised. ( if value is bigger then 256, the counter will stay on the max value- 256) |

NAME: Enable

DESCRIPTION: system is on\off.

ADDRESS: "0x03"

ACCESS: R

RESET VALUE: 0x00

|  |  |
| --- | --- |
| BIT | DESCRIPTION |
| TD[0] | 0-system off, we don't search for trigger rise (set by the GUI)  1- system on, start search for trigger rise |
| TD[1..7] |  |

**Generic list – Internal Logic Analyzer Core**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Defult Value | Type | Description | Generic Parameter | Number |
| 256 | Positive | Determine the number of bits that will be recorded for each signal | Record\_depth\_g | 1. |
| 8 | Positive | Determine the number of signals that will be recorded | Num\_of\_signals\_g | 2. |
| '1' | Std\_logic | Reset polarity:  '1': Active high  '0':Active low | Reset\_polarity\_g | 3. |
| '1' | Std\_logic | Enabling the system:  '1': Active high  '0':Active low | Enable\_polarity\_g | 4. |
| 8 | Positive | The Width of the basic 'word' of wishbone interface | Data\_width\_g | 5. |
| 8 | Positive | The address Width of wishbone interface | Add\_width\_g | 6. |
| 10 | Positive | Number of lines in the basic RAM used in the core | Signal\_ram\_depth\_g | 7. |
| 8 | Positive | The Width of the basic 'word' of the basic RAM used in the core | Signal\_ram\_width\_g | 8. |

Pin table of core parts:

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Width | Direction | Pin Name |
| Clock | 1 | In | Clk |
| Synchronous reset signal | 1 | In | Reset |
| Signal used for capturing data | 1 | In | trigger |
| The input signals which will be recorded | Num\_of\_signals\_g | in | input\_data |
| Set the Register's configuration according user's choice | 32 | in | Trigger configuration |
|  |  |  | WBS |
|  |  |  | WBM |

Write controller:

Get the recording configuration from the Gui, and record the data according it.

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Width | Direction | Pin Name |
| Clock | 1 | In | Clk |
| Synchronous reset signal | 1 | In | Reset |
| Identify the trigger type in order to find it in the incoming signal | 5 | In | Trigger\_type |
| The percent from the recorded data that will appear before the trigger | 7 | in | Trigger\_position |
| 0-system off, we don't search for trigger rise  1- system on, start search for trigger rise | 1 | In | System\_status |
| Get the data from the GUI through the WBS | Num\_of\_signals\_g | in | Data\_in\_WC |
| The trigger signal (same as data in signal) | 1 | in | Trigger |
| Generics that needed for deferent calculations |  | In | Generics |
| The address in the RAM to save the new coming word | Add\_width\_g | out | addr\_in |
| the start and the end addresses of the recorded data needed to be sent out (start,end) | 2\* Add\_width\_g | out | Wc\_to\_rc |
| Send the incoming data to be saved in the RAM (plus trigger) | Num\_of\_signals\_g + 1 | out | Data\_in\_RAM |
| 1. Data sends to the RAM from the WC is **not** valid 2. Data sends to the RAM from WC is valid | 1 | out | aout\_valid |
| Trigger rise has accorded and data is starting to send out according the trigger configurations | 1 | out | Trigger\_found |

Read controller:

This is the unit which sends the data out through the WBM according the resent configuration.

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Width | Direction | Pin Name |
| Clock | 1 | In | Clk |
| Synchronous reset signal | 1 | In | Reset |
| 0-RC off, we have not found a trigger rise  1- RC on, we start to read data from the RAM and send it out | 1 | In | Trigger\_found |
| the start and the end addresses of the recorded data that needed to be send out (start,end) | 2\* Add\_width\_g | in | Wc\_to\_rc |
| Get the recorded data from the RAM and sends it to the RC, one word at a time, according the start and end addresses from the WC | Signal\_ram\_width\_g | in | Data\_in\_RC |
| 1. Data out is not valid 2. Data out is valid | 1 | in | dout\_valid |
| Generics that needed for deferent calculations |  | In | Generics |
| Counts the number of clk cycles that passed since the RC started working (and outputting data) | 8 | out | Clk\_to\_start |
| Send the data out through the WBM to the GUI | Data\_width\_g | out | rc\_to\_WBM |
| Sends the RAM the address of the word needed to be send out (in that cycle) | Add\_width\_g | out | Addr\_out |

RAM:

The memory unit. The RAM is recording the incoming signals all the time, and when we find the correct configuration, the read controller read the data from the RAM.

|  |  |  |  |
| --- | --- | --- | --- |
| Description | Width | Direction | Pin Name |
| Clock | 1 | In | Clk |
| Synchronize reset signal  (din\_valid) | 1 | In | Reset |
| The address of the word that needed to be send out in this clk cycle | Add\_width\_g | in | Addr\_out |
| Set the incoming data address in the RAM | Add\_width\_g | in | Addr\_in |
| Enable the RAM to send data out  (aout\_valid) | 1 | in | aout\_valid |
| Save the incoming data in the RAM  (data in) | Signal\_ram\_width\_g | In | Data\_in\_RAM |
| Generics that needed for deferent calculations |  | In | Generics |
| Check if the out coming data is valid  (dout\_valid) | 1 | out | dout\_valid |
| Get the recorded data from the RAM and sends it to the RC, one word at a time, according the start and end addresses from the WC  (data out) | Signal\_ram\_width\_g | out | Data\_in\_RC |

נקבל את הנוסחא הבאה עבור מספר הRAM's:

