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: system\_status

DESCRIPTION: system is on\off.

ADDRESS: "0x03"

ACCESS: W\R

RESET VALUE: 0x00

|  |  |
| --- | --- |
| BIT | DESCRIPTION |
| TD[0] | 0-system off, we don't search for trigger rise (set at the beginning to 0 until all other registers are ready, change in second time after the trigger was found- don't continue to search for trigger rise)  1- system on, start search for trigger rise (changed by write controller after all other registers are in the right mode). |
| 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. |
| 8 | Positive | The Width of the basic 'word' of wishbone interface | Data\_width\_g | 4. |
| 8 | Positive | The address Width of wishbone interface | Add\_width\_g | 5. |
| 10 | Positive | Number of lines in the basic RAM used in the core | Signal\_ram\_depth\_g | 6. |
| 8 | Positive | The Width of the basic 'word' of the basic RAM used in the core | Signal\_ram\_width\_g | 7. |

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 |
|  |  |  | 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 (set at the beginning to 0 until all other registers are ready, change in second time after the trigger was found- don't continue to search for trigger rise)  1- system on, start search for trigger rise (changed by write controller after all other registers are in the right mode). | 1 | In\out | System\_status |
| Counts the number of clk cycles that passed since the system started working until trigger raised | 8 | out | Clk\_to\_start |
| Check in the RAM for trigger rise and determine according the trigger position the start and the end addresses of the recording data | 2\* Add\_width\_g | in | ram\_to\_wc |
| the start and the end addresses of the recorded data | 2\* Add\_width\_g | out | Wc\_to\_rc |
| Get the data from the GUI through the WBS | Data\_width\_g | in | WBS\_to\_wc |

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-system off, we don't search for trigger rise (set at the beginning to 0 until all other registers are ready, change in second time after the trigger was found- don't continue to search for trigger rise)  1- system on, start search for trigger rise (changed by write controller after all other registers are in the right mode). | 1 | In\out | system\_status |
| Send the data out through the WBM to the GUI | Data\_width\_g | out | rc\_to\_WBM |
| the start and the end addresses of the recorded data | 2\*Signal\_ram\_width\_g | in | wc\_to\_rc |
| Get the recorded data from the RAM, according the addresses from the wc | Signal\_ram\_width\_g | in | Ram\_to\_rc |

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 |
| Synchronous reset signal  (din\_valid) | 1 | In | Reset |
| Get the address of the beginning and the end of the data to send out  (addr\_out) | 2\* Add\_width\_g | in | wc\_to\_ram |
| Set the incoming data address in the RAM  (addr\_in) | Add\_width\_g | in | WBS,WC |
| Enable the RAM to send data out  (aout\_valid) | 1 | in | Enable\_out\_valid |
| Save the incoming data in the RAM  (data in) | Signal\_ram\_width\_g | In | Data\_in |
| Check if the out coming data is valid  (dout\_valid) | 1 | out | Out\_valid |
| Send the recorded data from the RAM to the rc, according the addresses from the wc  (data out) | Signal\_ram\_width\_g | out | Data\_out |

נקבל את הנוסחא הבאה עבור מספר הRAM's:



שאלות

* האם לממש RAM קטן יותר או להשתמש ביחידה הבסיסית אפילו אם ההקלטה קטנה יותר (ביחידה בסיסית 4096 ביטים להקלטה)

יש GENERIC-ים לטובת הגודל והרוחב של ה- RAM הבסיסי.

* האם לקבוע גודל מקסימלי של הקלטה או שהמשתמש יכול להקליט כל מספר ביטים שהוא רוצה?

יש GENERIC, והכול צריך להיות תלוי ב- GENERICS-ים.

איך מייצגים את הבעיה כאשר צריך לשלוח את עומק ההקלטה בין היחידות במערכת (לדוגמא רוחב הפס בין READ-WRITE -> מצריך שליחת כתובת, תלוי ברוחב ההקלטה)

הכול צריך להיות תלוי ב- GENERIC-ים, ואם חסר GENERIC-ים שלא חשבנו עליהם אז יש להוסיף. ה- USER צריך לקבל גמישות מלאה.

* האם לRAM יש כניסת שעון וRESET

כן.

בהקצאת המקום עבור רגיסטר- SYSTEM STATUS לדוגמא, האם למרות שזה רגיסטר שרוחבו 8 ביט, בגלל שאנחנו משתמשים רק בביט אחד שלו אפשר להגיד שרוחבו 1 או שנידרש להקצות את כל רוחב הרגיסטר?(כרגע מוקצה לו רק ביט אחד)

האם המידע הנכנס שמיועד להקלטה מגיע ל RAM דרך WBS או דרך write controller?