TEST NUMBER 1

Generics values:

|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 5 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 6 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

Variables values:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Comments | Value | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | Name |
| Inputs | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | 50 duty cycle | | | | | | | | | | | | | | | | | | | | | | | | | | | | | clk |
|  | | 120-30000 | | | | | | | | | | | | | | | | 0-120 | | | | | | | | | | | | | Reset |
| 0 | | | | | | | | | | | | | | | | 1 | | | | | | | | | | | | |
| Counter | | 0-400 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | data\_in |
|  | | 25600-30000 | 7500-25600 | | | | | 5300-7500 | | | | | | 3100-5300 | | | 1000-3100 | | | | | | | 780-1000 | | | | 0-780 | | | Trigger |
| 0 | 1 | | | | | 0 | | | | | | 1 | | | 0 | | | | | | | 1 | | | | 0 | | |
| Address of register to write to | | 1400 -30000 | | | 1000 - 1400 | | | | | | | | | | 700 - 1000 | | | | 400 -700 | | | | | | | | | | 0-400 | | ADR\_I |
|  | | 4 | | | 0 | | | | | | | | | | 3 | | | | 2 | | | | | | | | | | 1 | |  |
| Data stored is registers. (3 register is clk\_to\_cycle which is not in use) | | 3000-30000 | | | 1000-3000 | | | | | | | | | | 700-1000 | | | | 400-700 | | | | | | | | | | 0-400 | | DAT\_I |
| 0 | | | 1 | | | | | | | | | | 7 | | | | 50 | | | | | | | | | | 1 | |
| 1 for write. | | 0-22000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WE\_I |
| 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Data in registers valid | | other | | 23100-23200 | | | | | 2500-2600 | | 1300 -1400 | | | | | 1000-1100 | | | | | | 700-800 | | | 400-500 | | | | | 200-300 | STB\_I | |
| 0 | | 1 | | | | | 1 | | 1 | | | | | 1 | | | | | | 1 | | | 1 | | | | | 1 |
| 1 for bus transmition request for input registers data | | 25000-30000 | | | | 23000-25000 | | | | | | 3000-23000 | | | | | | | | 150-3000 | | | | | | 0-150 | | | | | CYC\_I |
| 0 | | | | 1 | | | | | | 0 | | | | | | | | 1 | | | | | | 0 | | | | |
| type of core WBS | | 0-30000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | TGA\_I |
| 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Length of the word (maybe will change) | | 0-30000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | TGD\_I |
| 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| maybe will change | | 0-30000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | stall |
| 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Come from external WBS | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | ACK\_I |
| Come from external WBS | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | STALL\_I |
| Come from external WBS | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | ERR\_I |
| Outputs | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CYC\_I output to user side. | | 25000-30000 | | | | 23000-25000 | | | | | | 3000-23000 | | | | | | | | 150-3000 | | | | | | 0-150 | | | | | TOP\_active\_cycle |
| 0 | | | | 1 | | | | | | 0 | | | | | | | | 1 | | | | | | 0 | | | | |
| Follow STB\_I | | other | | 23100-23200 | | | | | 2500-2600 | | 1300 -1400 | | | | | 1000-1100 | | | | | | 700-800 | | | 400-500 | | | | | 200-300 | ACK\_O |
| 0 | | 1 | | | | | 1 | | 1 | | | | | 1 | | | | | | 1 | | | 1 | | | | | 1 |
| We do not output data from WBS | | 0-30000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | WS\_DAT\_O |
| 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| We do not output data from WBS | | 0-30000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | STALL\_O |
| 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Maybe an error | | 0-30000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | wm\_end\_out |
| 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Maybe an error | | 7650-30000 | | | | | | | | 7500-7650 | | | | | | | | | | | | | 0-7500 | | | | | | | | ADR\_O |
| 2 | | | | | | | | 1 | | | | | | | | | | | | | 0 | | | | | | | |
| Output the relevant data | | 20650-30000 | | | | | | | | 7750-20650 | | | | | | | | | | | | | 0-7750 | | | | | | | | WM\_DAT\_O |
| 0 | | | | | | | | Relevant data | | | | | | | | | | | | | 0 | | | | | | | |
|  | | 7650-30000 | | | | | | | | 7450-7650 | | | | | | | | | | | | | 0-7450 | | | | | | | | WE\_O |
| 0 | | | | | | | | 1 | | | | | | | | | | | | | 0 | | | | | | | |
|  | | 7650-30000 | | | | | | | | 7350-7650 | | | | | | | | | | | | | 0-7350 | | | | | | | | STB\_O |
| 0 | | | | | | | | 1 | | | | | | | | | | | | | 0 | | | | | | | |
| Maybe an error | | 7350-30000 | | | | | | | | | | | | | | 0-7350 | | | | | | | | | | | | | | | CYC\_O |
| 1 | | | | | | | | | | | | | | 0 | | | | | | | | | | | | | | |
| Output the data to WBS number 4 | | 7350-30000 | | | | | | | | | | | | | | 0-7350 | | | | | | | | | | | | | | | TGA\_O |
| 4 | | | | | | | | | | | | | | 0 | | | | | | | | | | | | | | |
| Length of data (in words) | | 7350-30000 | | | | | | | | | | | | | | 0-7350 | | | | | | | | | | | | | | | TGD\_O |
| 1 | | | | | | | | | | | | | | 0 | | | | | | | | | | | | | | |
| Internal signals | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Enable. Come from ENABLE - SM | | 27650-30000 | | | | | 23350-27650 | | | | | | 7350-23350 | | | | | | | | 2750-7350 | | | | | | 0-2750 | | | | Enable\_s |
| 0 | | | | | 1 | | | | | | 0 | | | | | | | | 1 | | | | | | 0 | | | |
| Output data from RC | | 28050-30000 | | | | | | | 20650-28050 | | | | | | | 7750-20650 | | | | | | | | | 0-7750 | | | | | | data\_from\_rc\_to\_wm\_s |
| DATA | | | | | | | 0 | | | | | | | DATA | | | | | | | | | 0 | | | | | |
| Validity of output data from RC | | Rise for one clock cycle in every new data that output through RC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | data\_from\_rc\_to\_wm\_valid\_s |
| Rise for one cycle when WC end saving all the data | | 27650-30000 | | | | | 27550-27650 | | | | | | 7350-27550 | | | | | | | | 7250-7350 | | | | | | 0-7250 | | | | write\_controller\_finish\_s |
| 0 | | | | | 1 | | | | | | 0 | | | | | | | | 1 | | | | | | 0 | | | |
| Rise for one cycle when RC end outputing all the data | | 20650-30000 | | | | | | | | 20550-20650 | | | | | | | | | | | | | 0-20550 | | | | | | | | read\_controller\_finish\_s |
| 0 | | | | | | | | 1 | | | | | | | | | | | | | 0 | | | | | | | |
| Rise after trigger found until WC finish working | | 37650-30000 | | | | | 25650-27650 | | | | | | 7350-25650 | | | | | | | | 5350-7350 | | | | | | 0-5350 | | | | trigger\_found\_s |
| 0 | | | | | 1 | | | | | | 0 | | | | | | | | 1 | | | | | | 0 | | | |

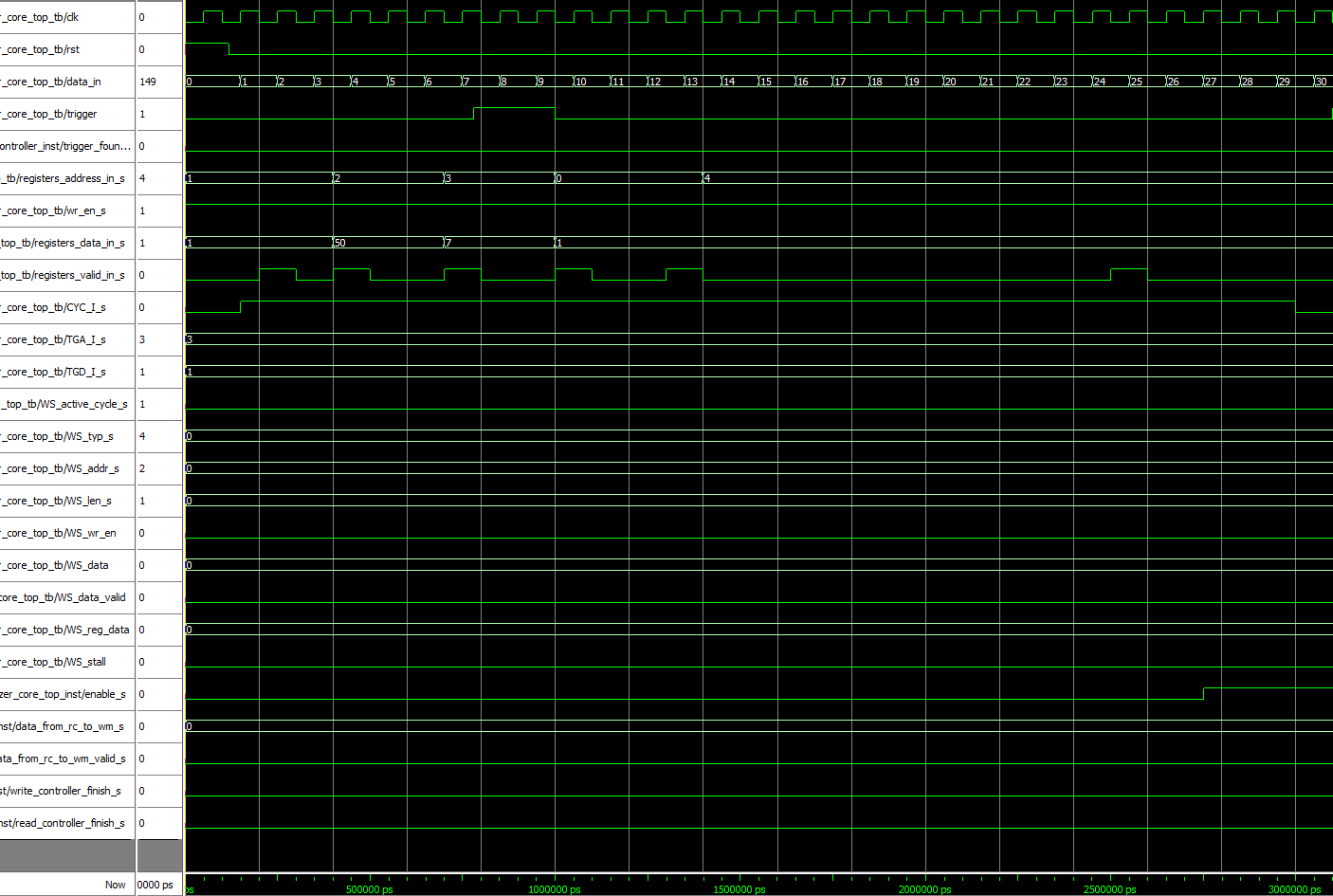
Explanation:

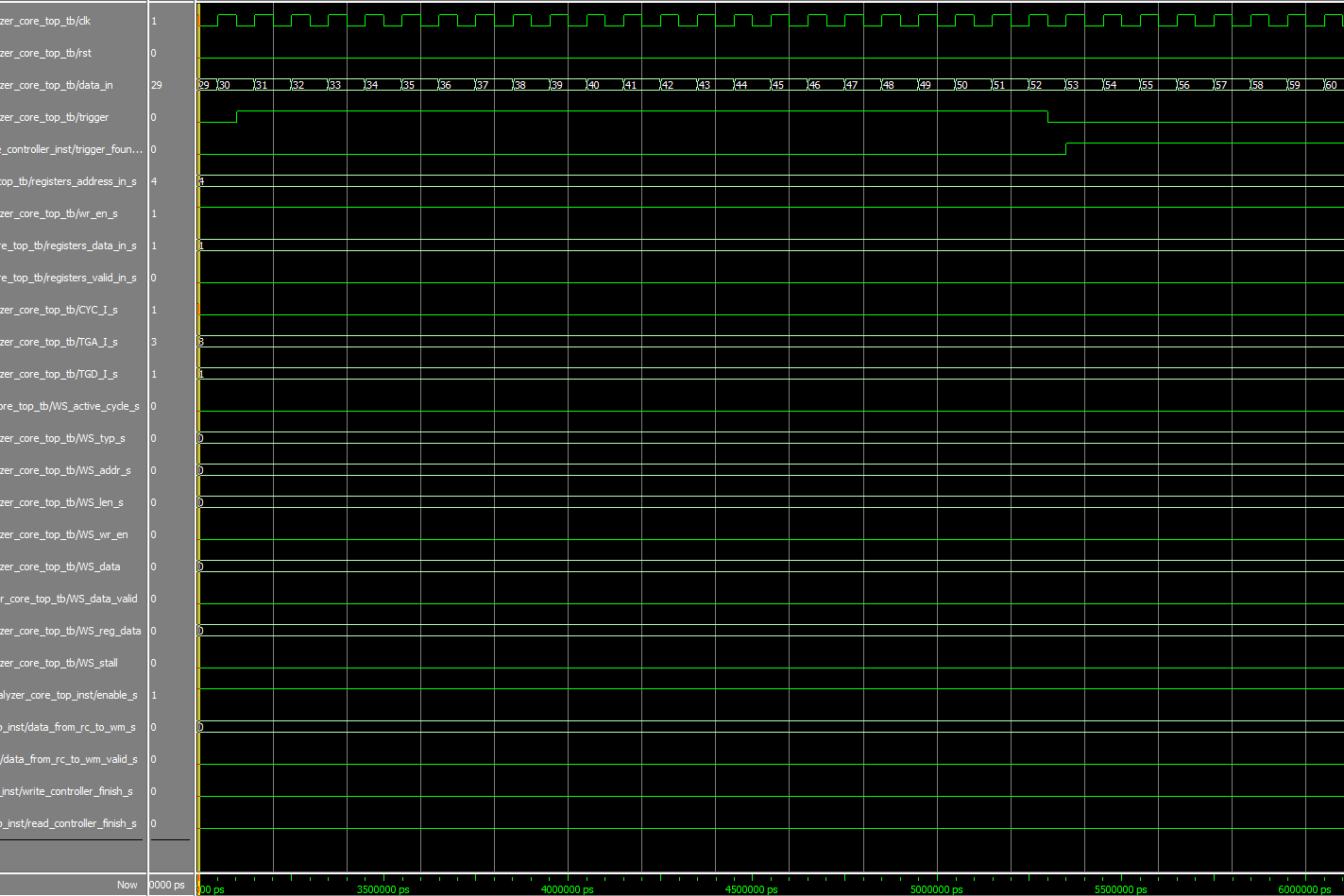
We input the user configurations through the WBS and save it in the registers, after that we wait for enable rise and the system start to save the data since that. When we detect trigger rise we continue to save the relevant data and after that output it through the WBM.

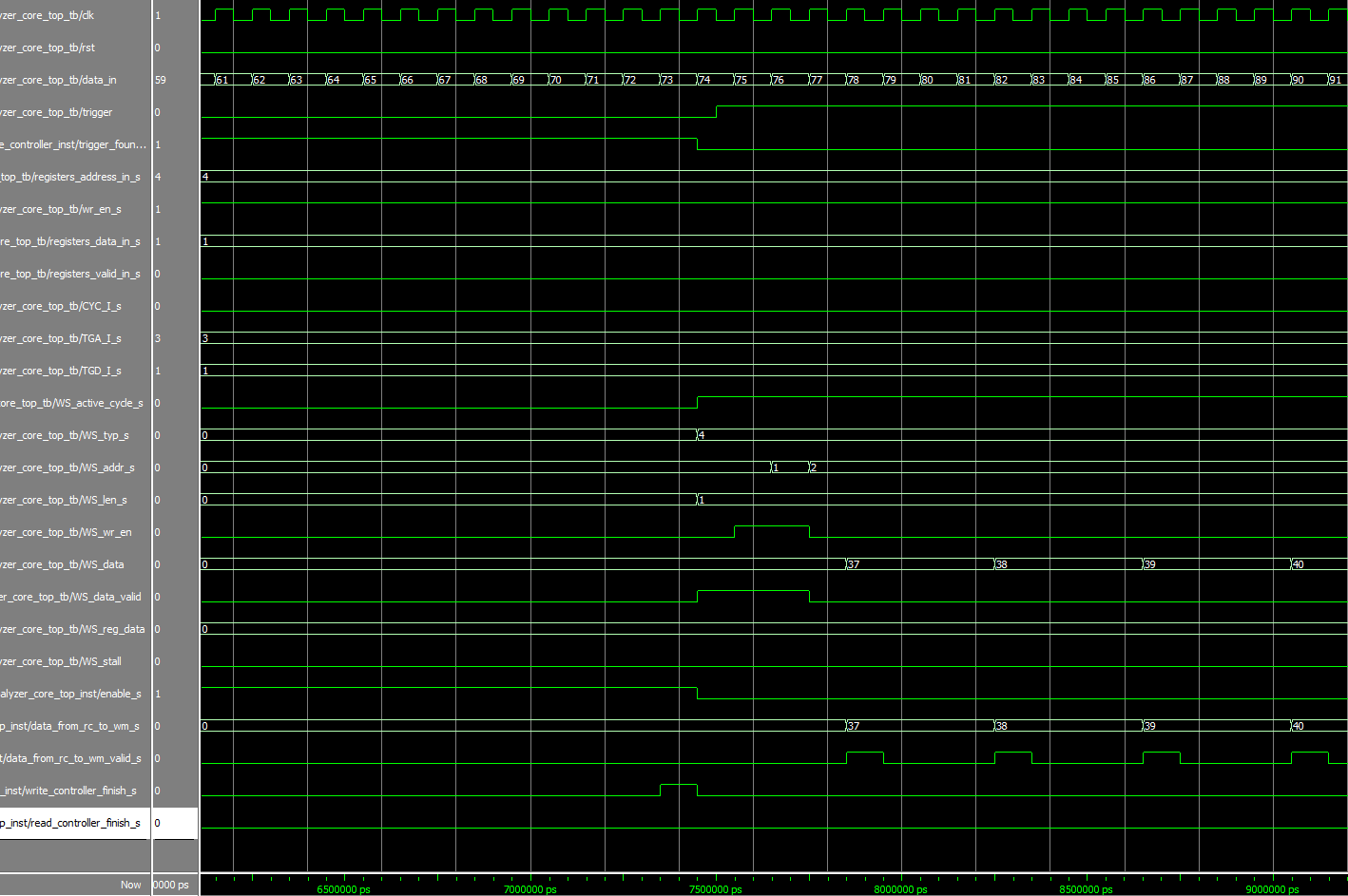
We can see that after that we finish outputting all the data we start again and the system wait for another enable rise (the other configurations do not have to change).

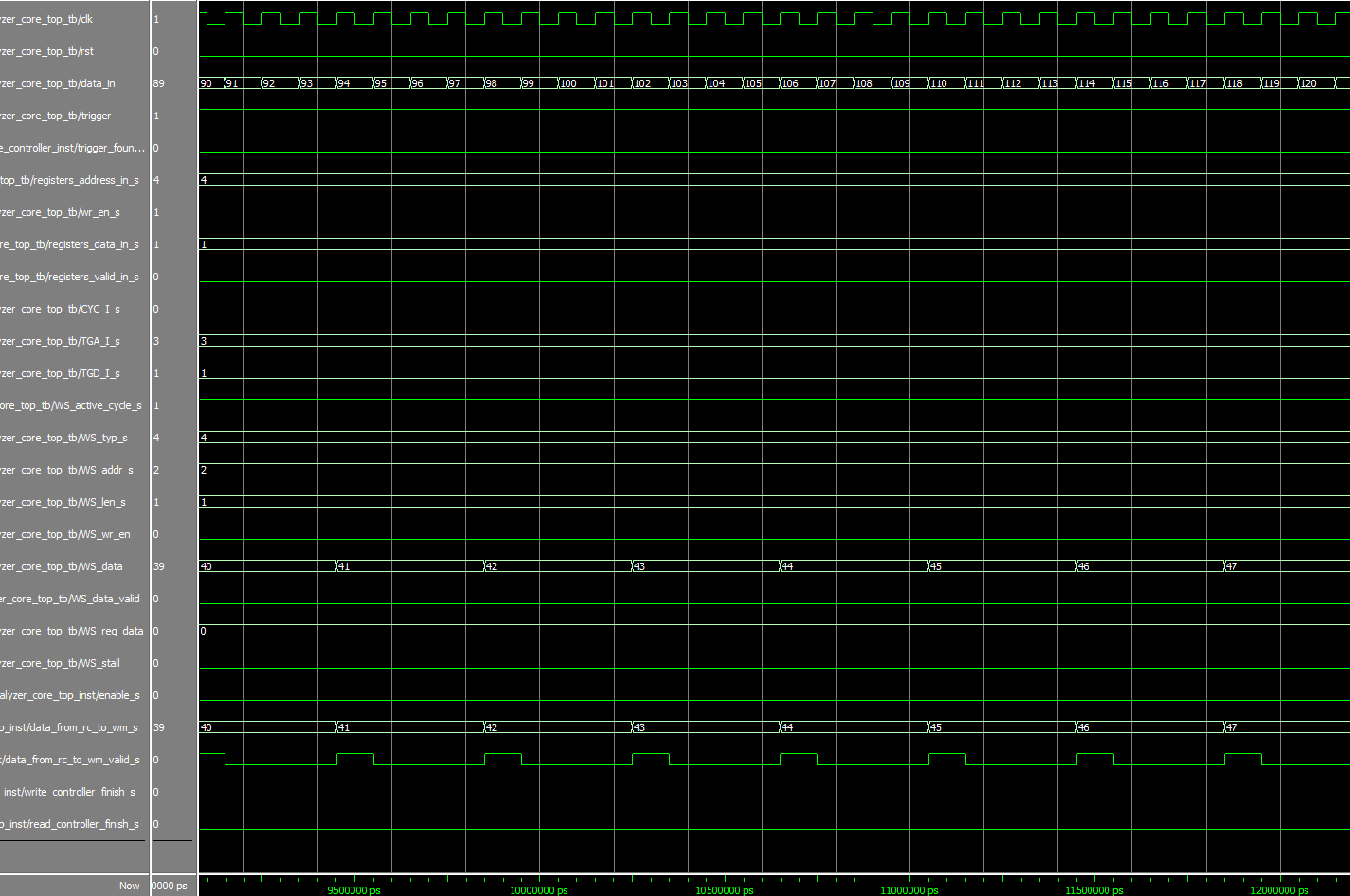
The data and trigger input are being read every cycle, but the data output is every 3 cycles.

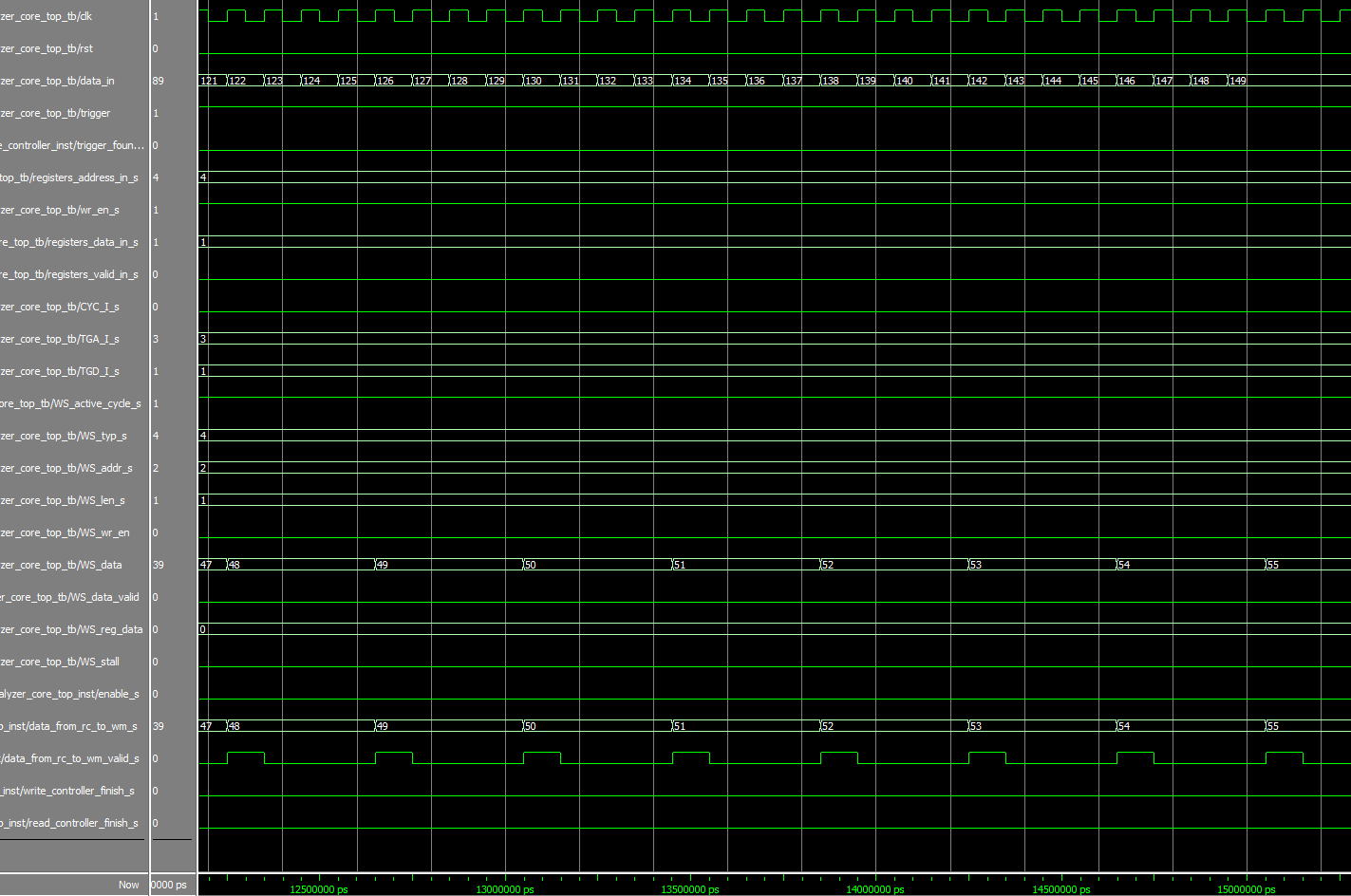
Simulation:

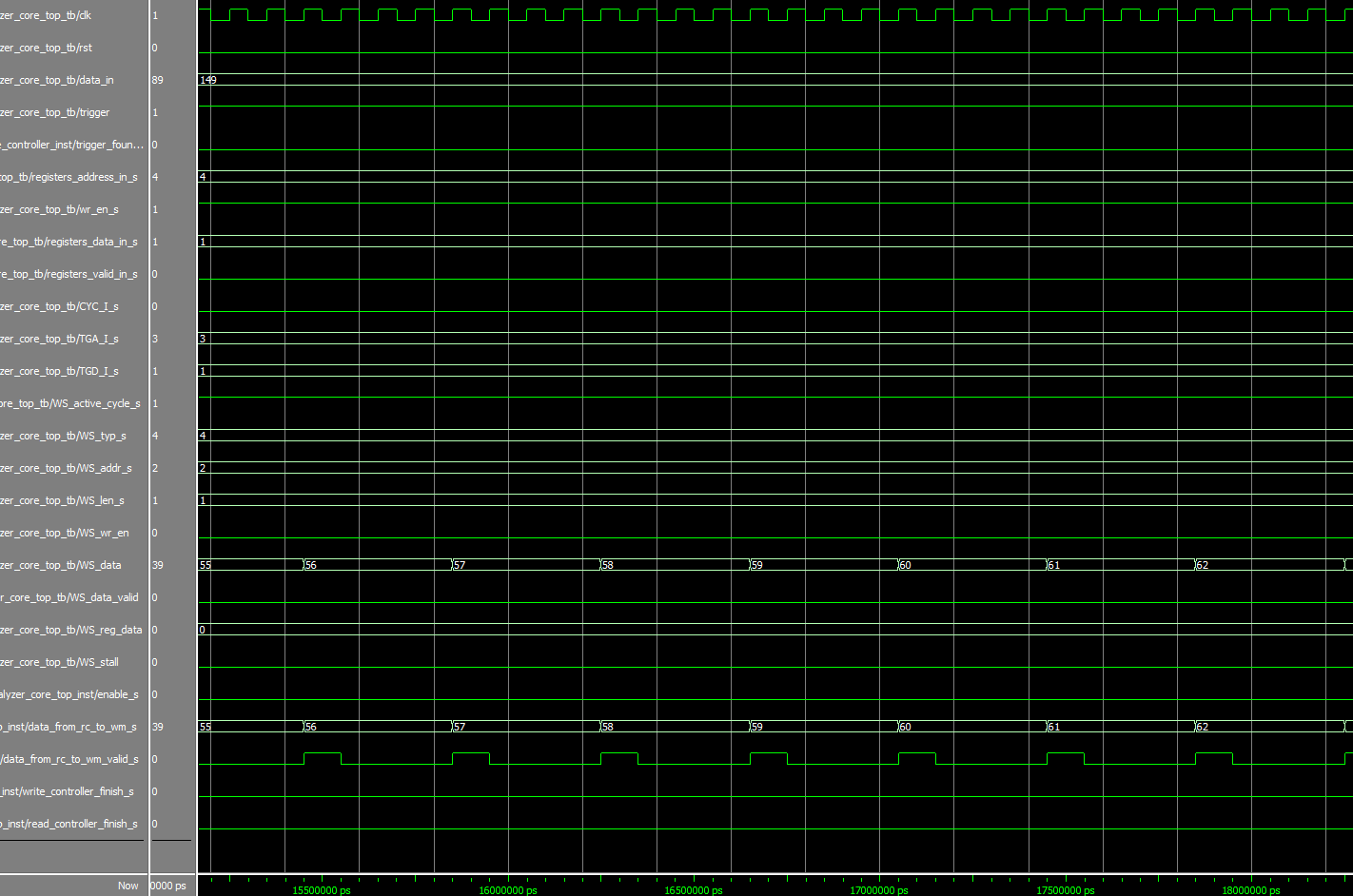


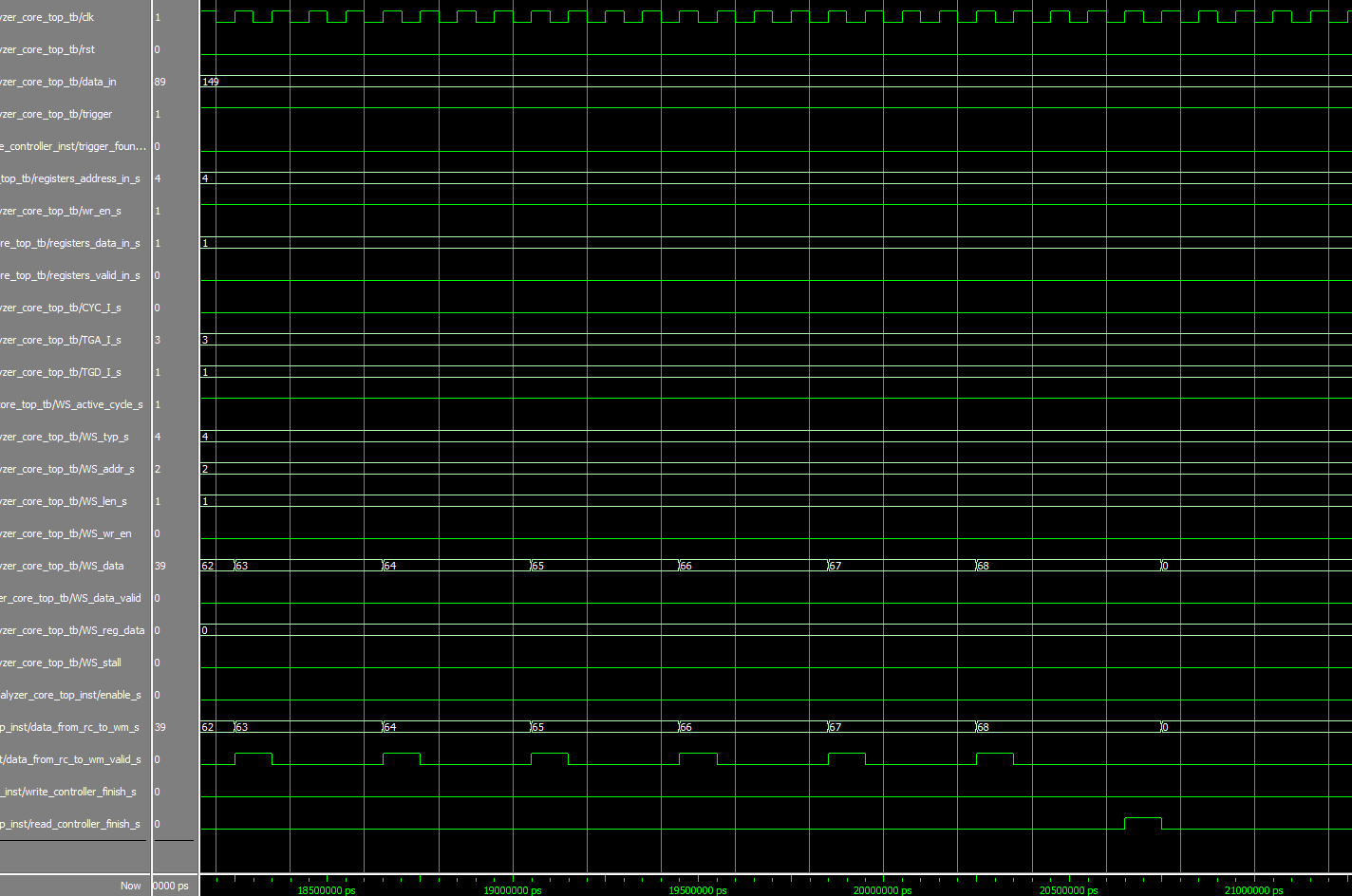












Analysis:

At first we initialize the registers with the user configurations (type is fall, position is 50% ) and then we wait for enable to rise (also save at a register). We then start to save the data at the RAM and searching for trigger fall. After we found the trigger fall we continue to save the rest of the data that is relevant to us and in the end we output the data through the WBM back to the user.

We can see that when the RC is finish outputting the data, the enable register is rested back to 0 in order to allow a new enable rise.

In order that the RC will start just after the WC, the wc\_finish signal is the enable of the RC.

Recorded depth = 5 , so we recording 32 signals when we start from 16 signals before trigger rise (position is 50%). Trigger rise at 53 so 16 cycles before is 37 until 68 (we can change it to 38 to 69), and we can see that this is the output.

TEST NUMBER 2

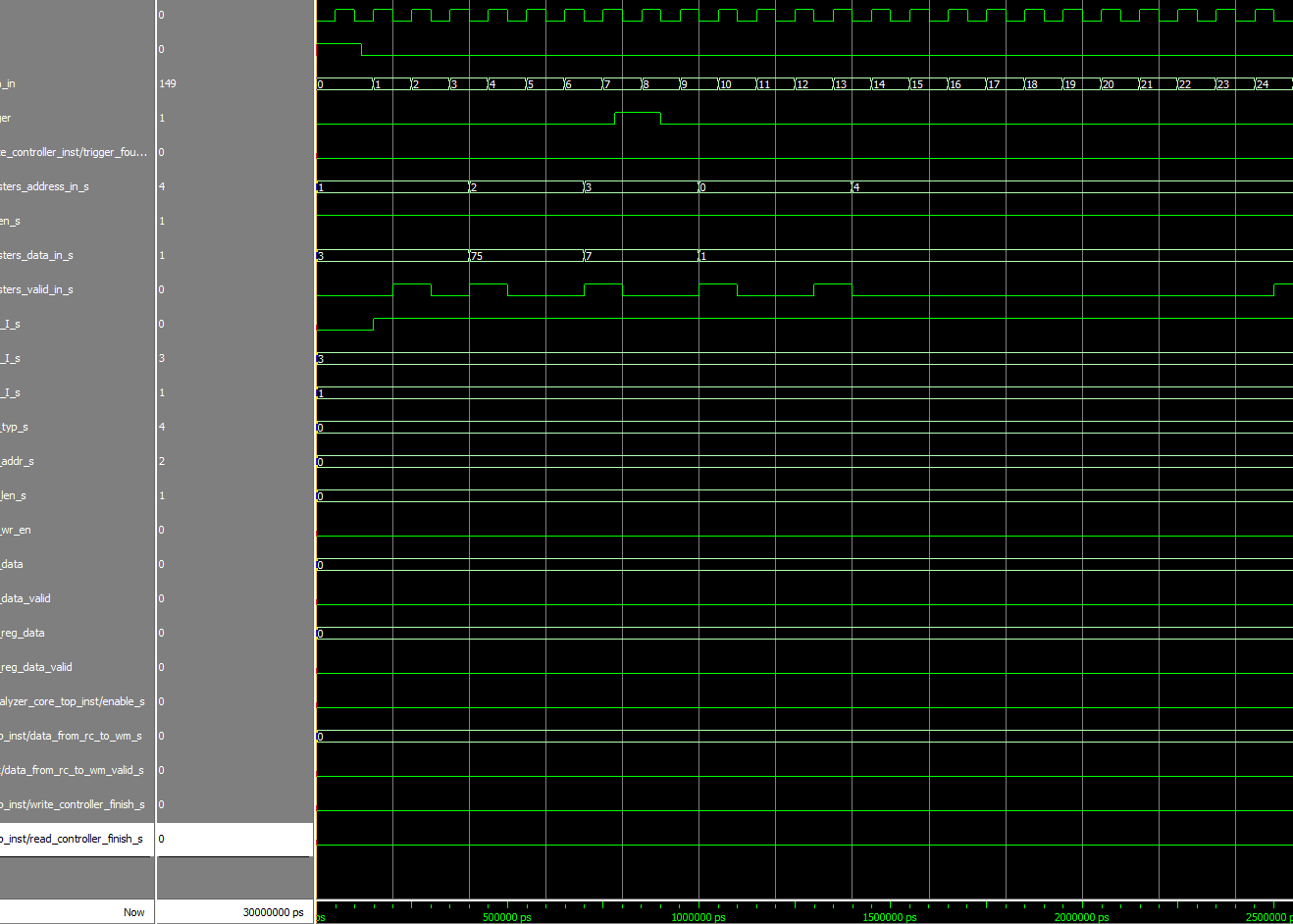
Generics values:

|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 3 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 6 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

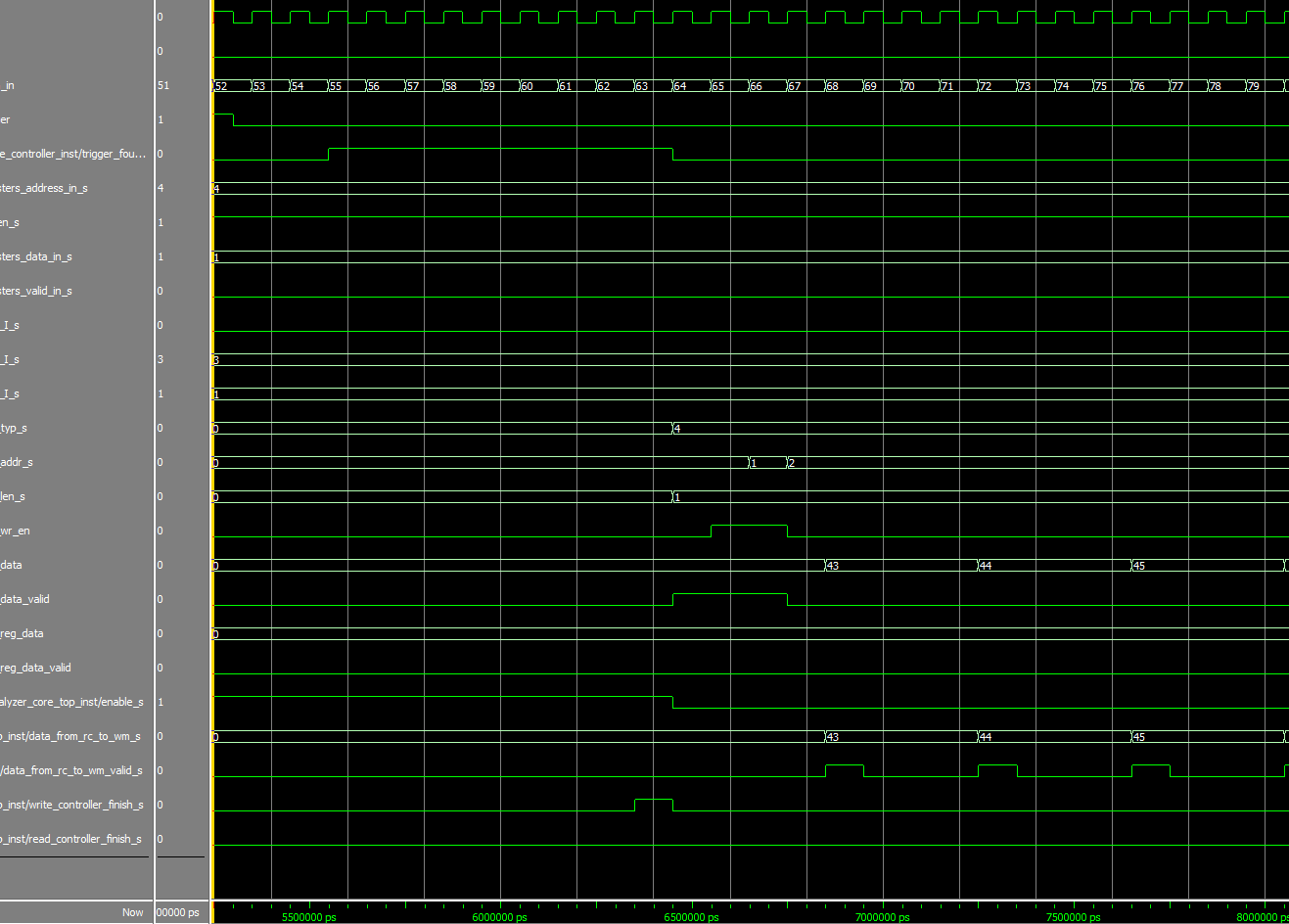
Explanation:

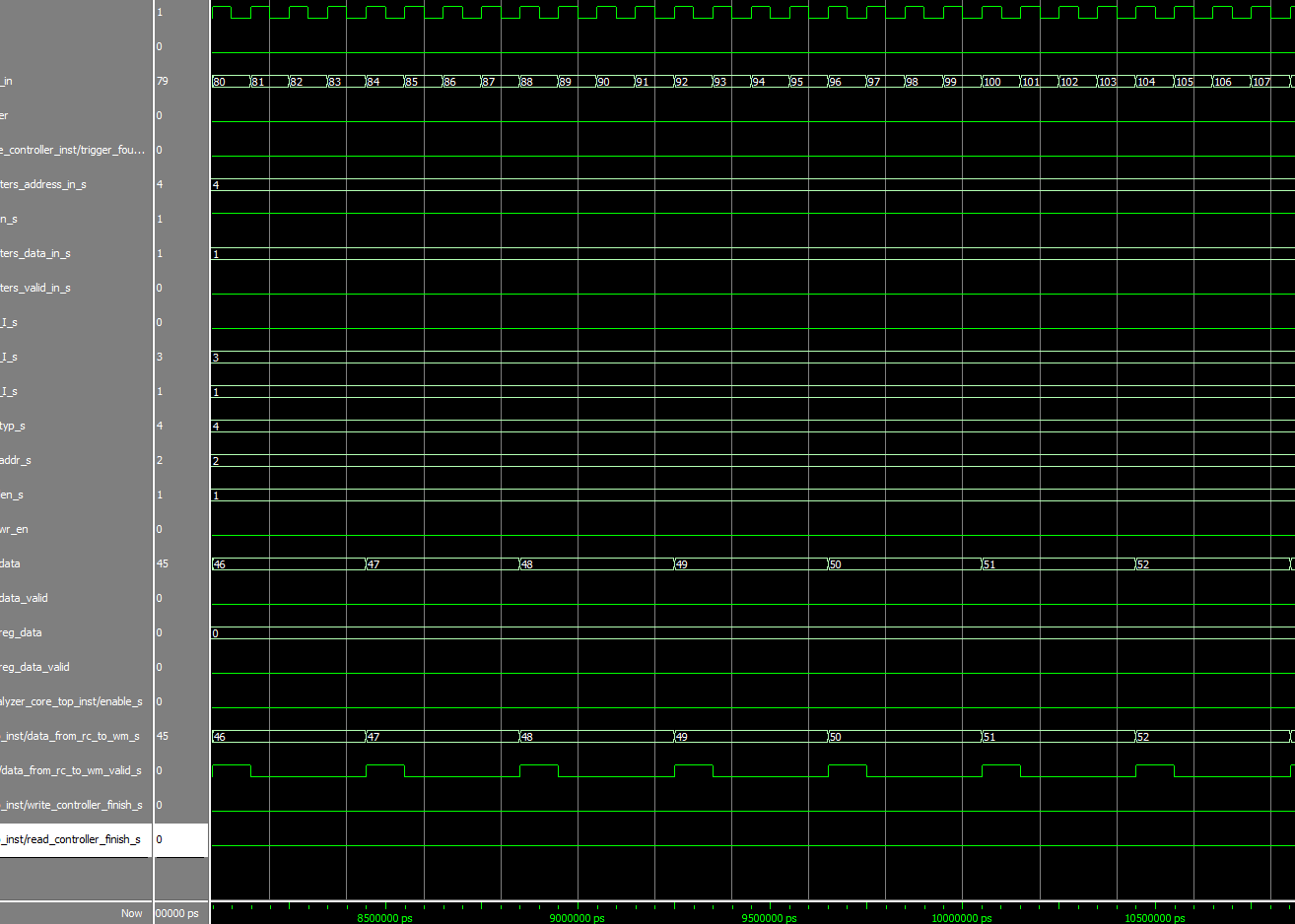
We change the recording depth to 4 (2^4 = 16), position to 75% (meaning that 75% -> 12 bits, will recorded before the trigger and the other after that, we also change the trigger type to zeroes (number 3 ) so trigger will rise after three low sampling.

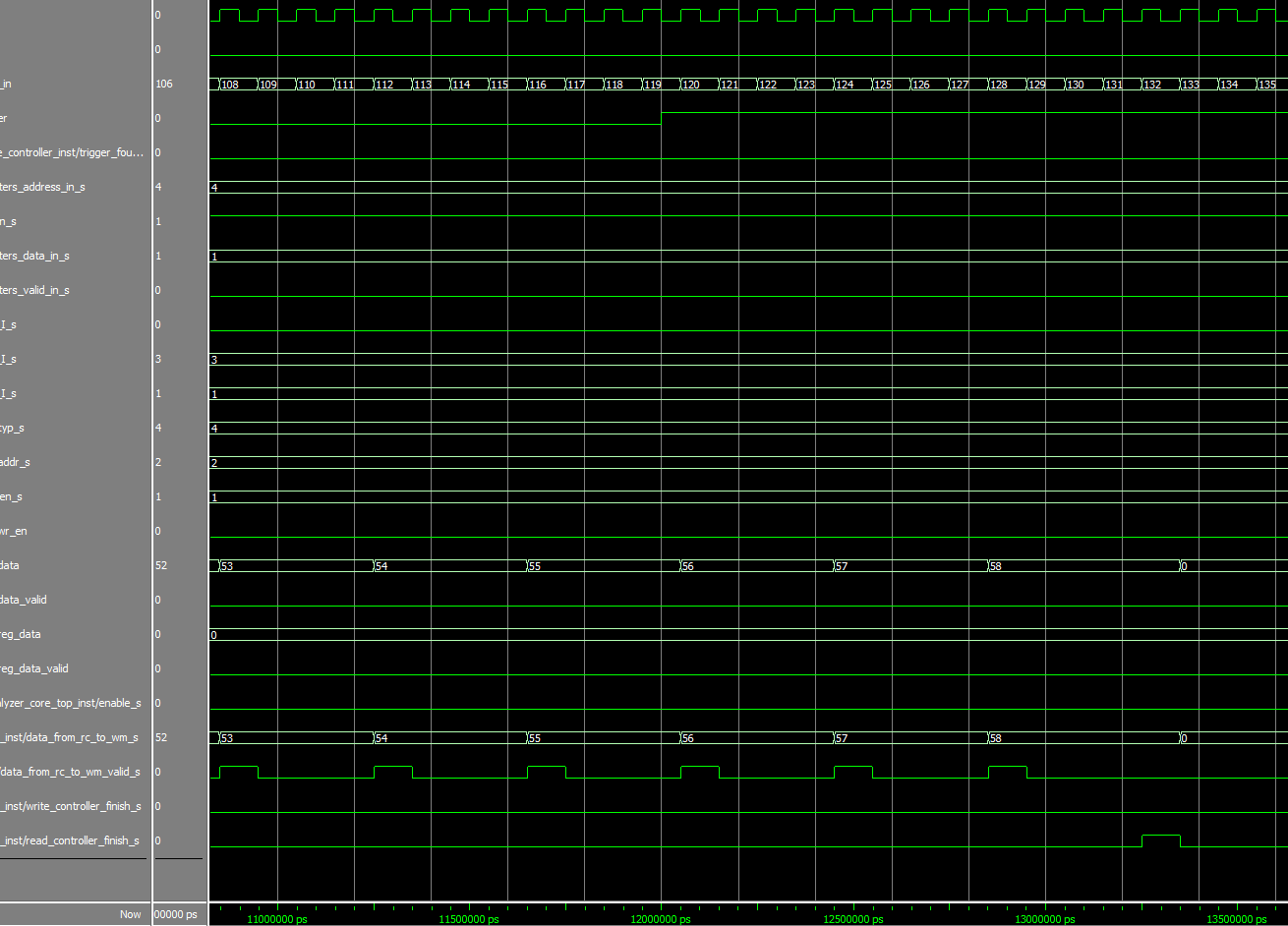
Simulation:

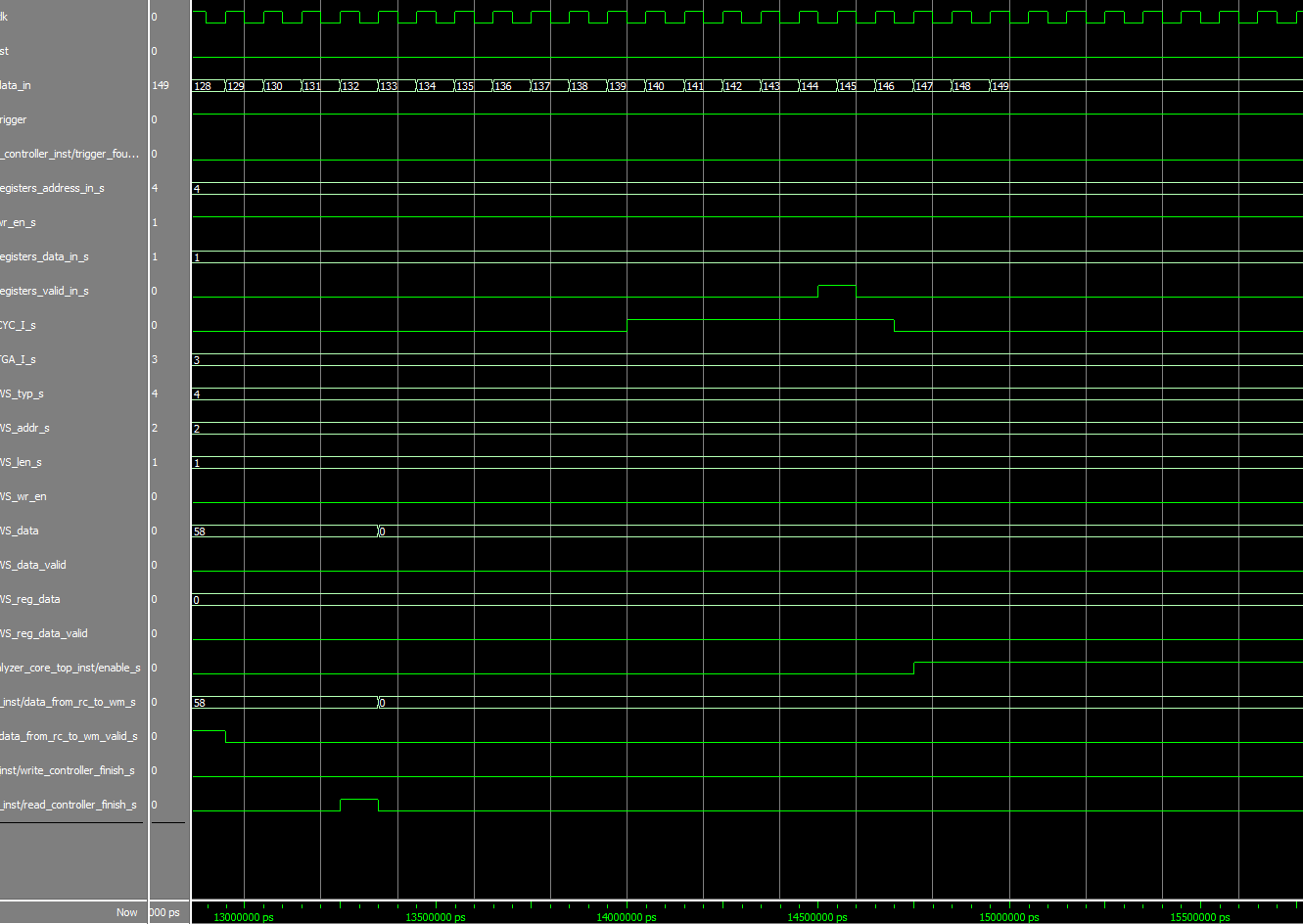


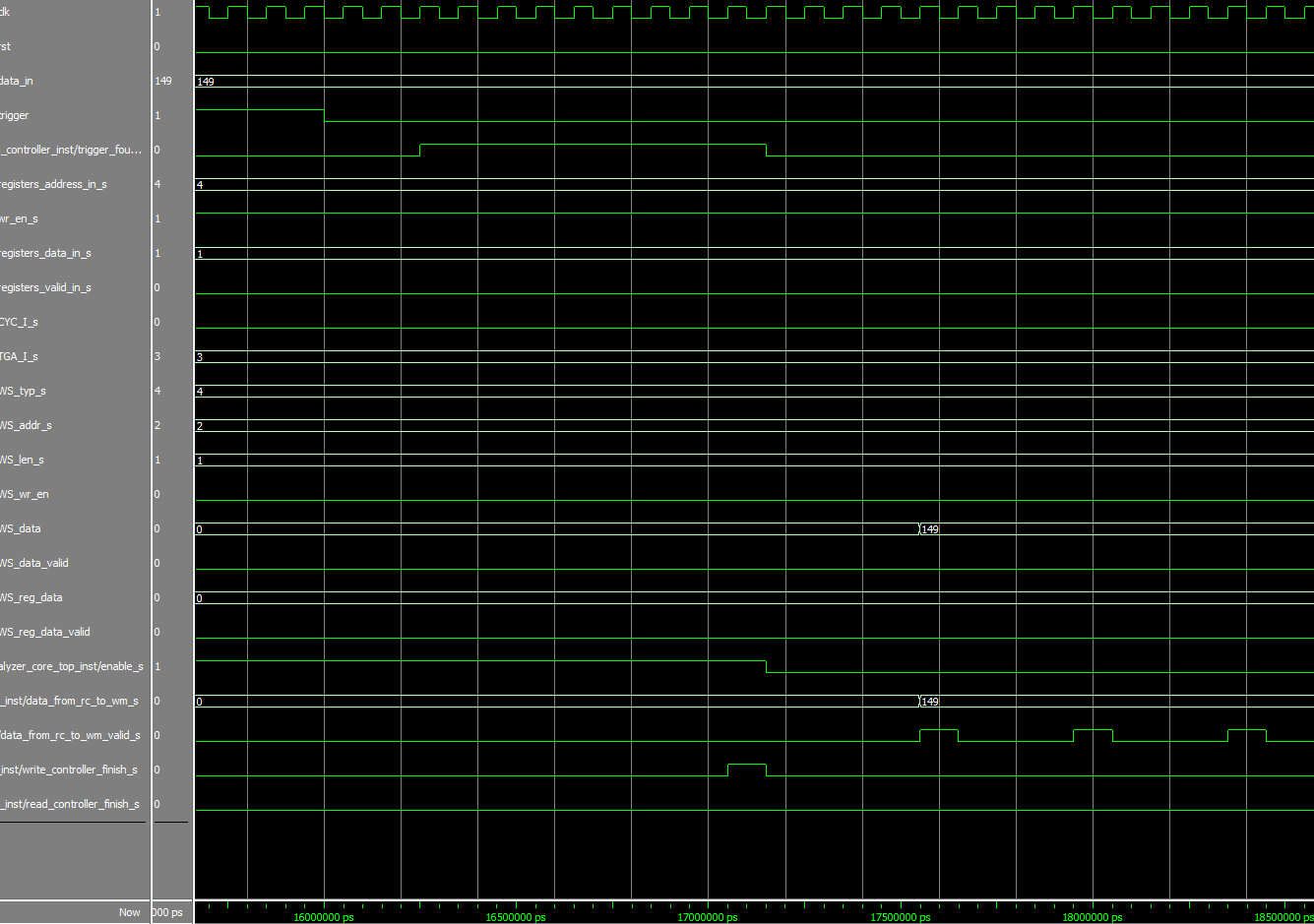


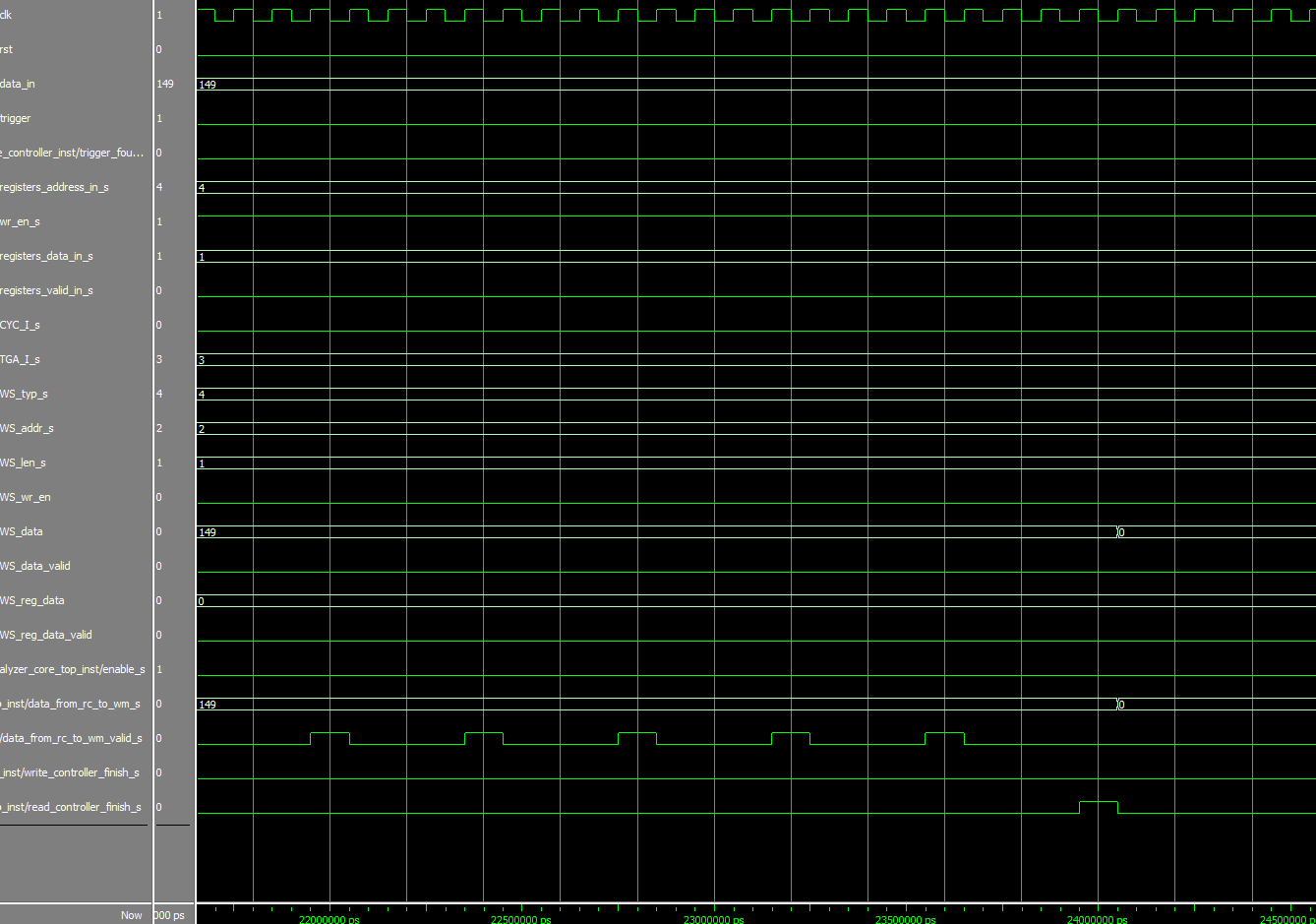












Analysis:

We change recorded depth to 4, meaning we record 2^4= 16 bits. We can see that just after the trigger is low for 3 cycles the system detect that and rise the "trigger found" signal. The system is also output 12 bits (75% of 16) before the trigger rise (first rise at 55 and output starts at 43 -> 55 – 12 = 43) and we output 16 bits after that.

After we output all the data we again enable the system by writing again to the enable register (the address and data to registers is already at the correct values) and couple of cycles after that we again detect "trigger rise" and the whole cycle start over.

TEST NUMBER 3

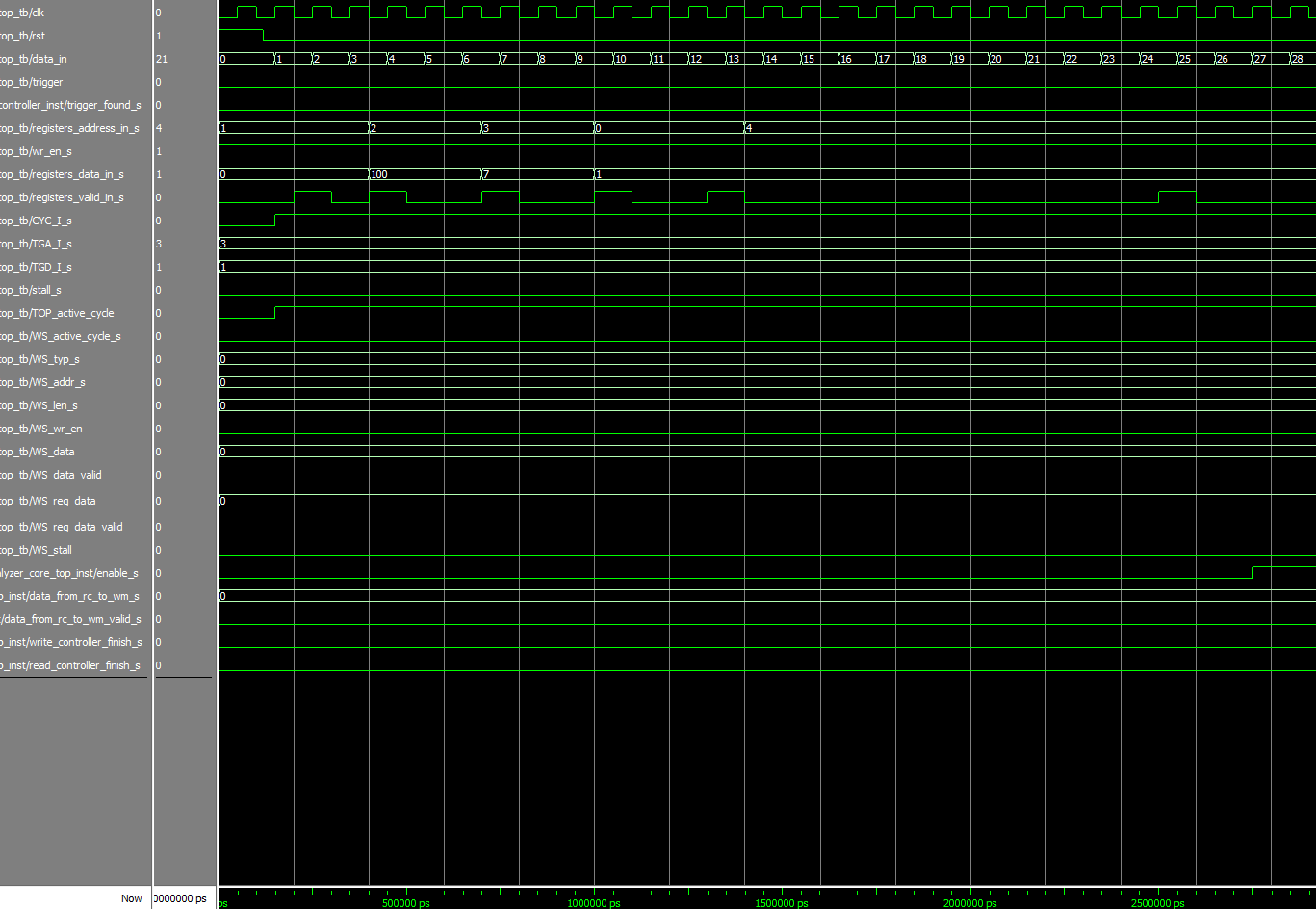
Generics values:

|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 3 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 5 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 6 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

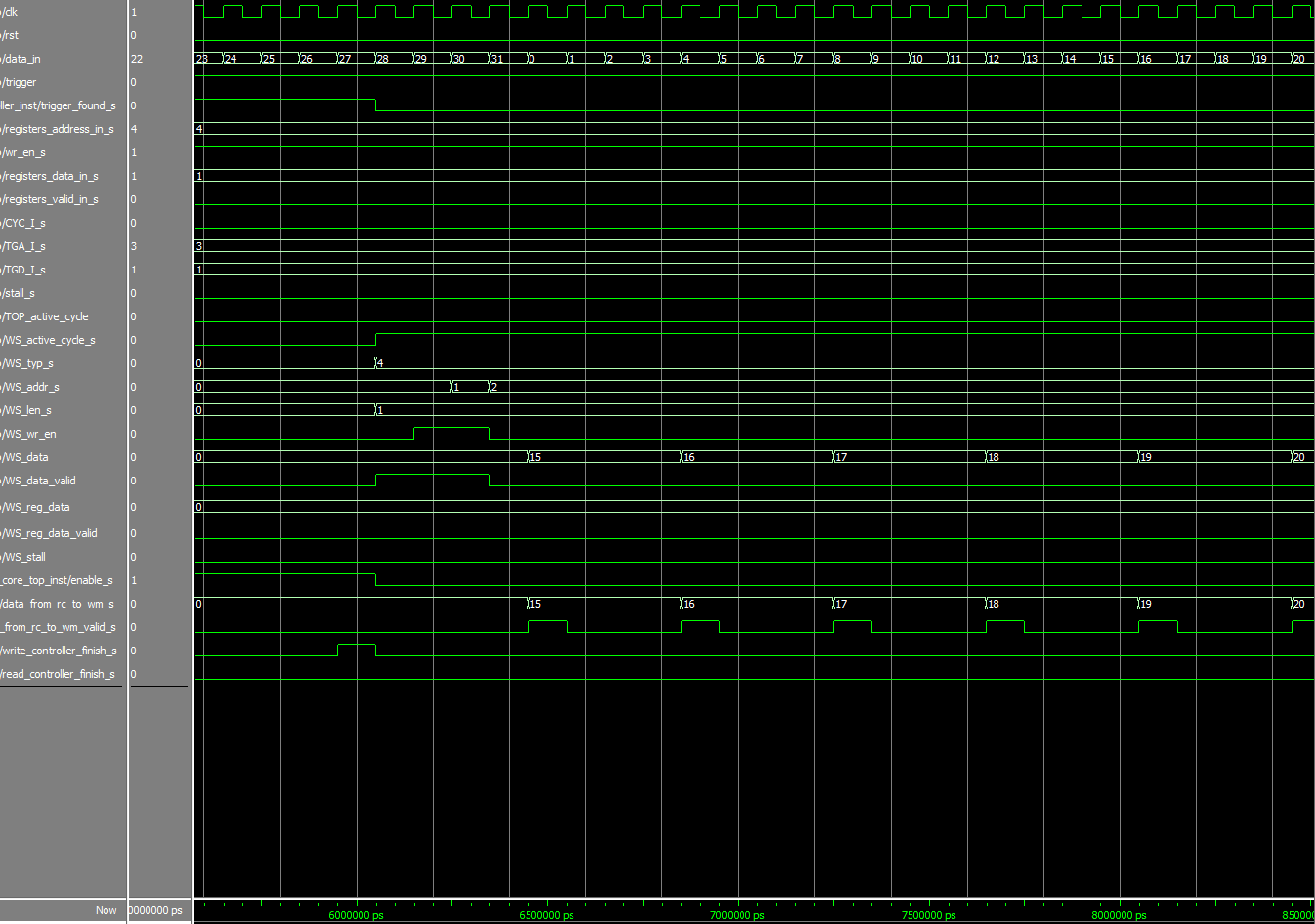
Explanation:

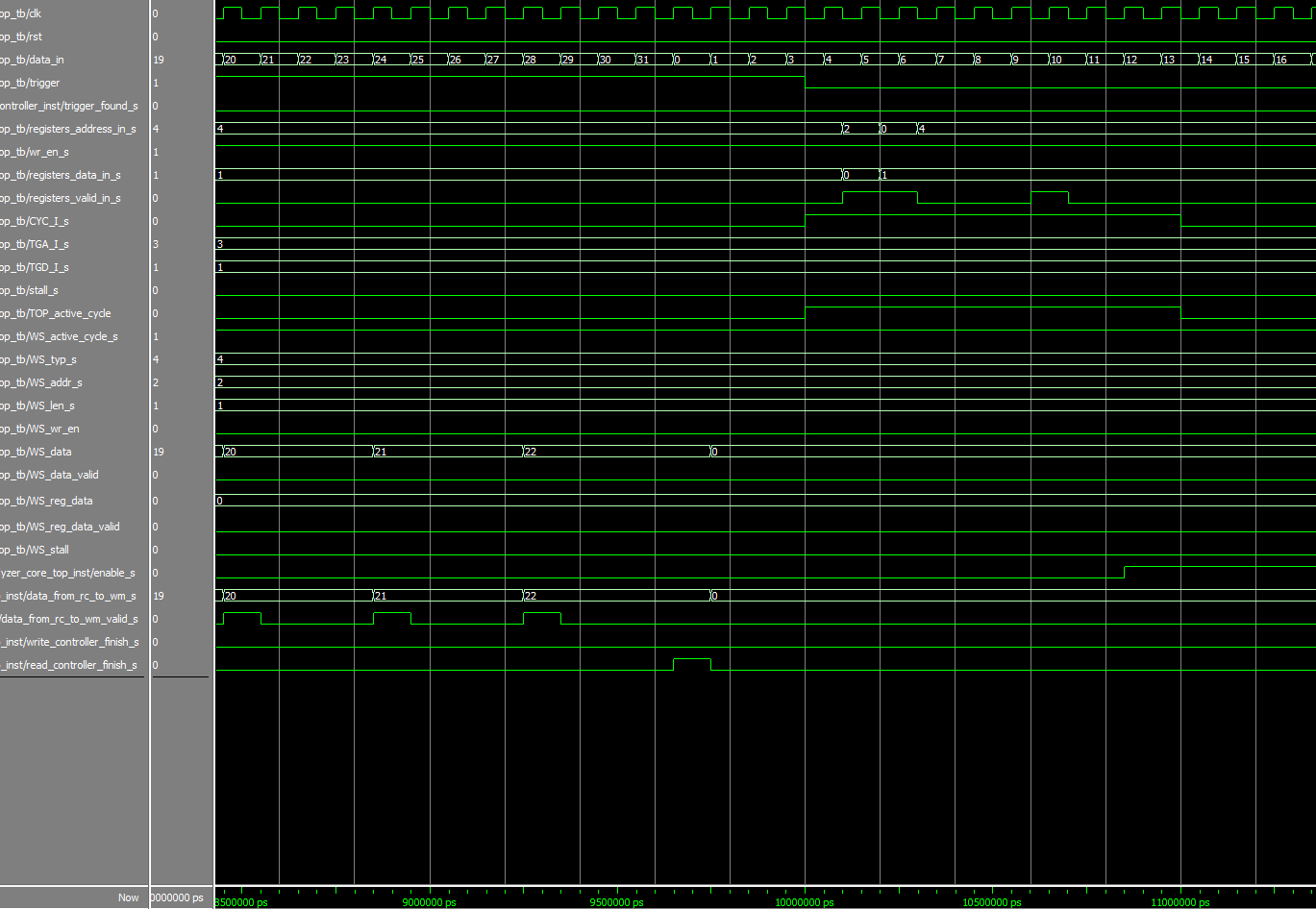
Number of signals is changed to 5, meaning our input data is between 0-32 in decimal (2^5), at first the trigger position is 100 and all the data is recorded before the trigger, and second time the position is 0 and all the data is recorded after the trigger.

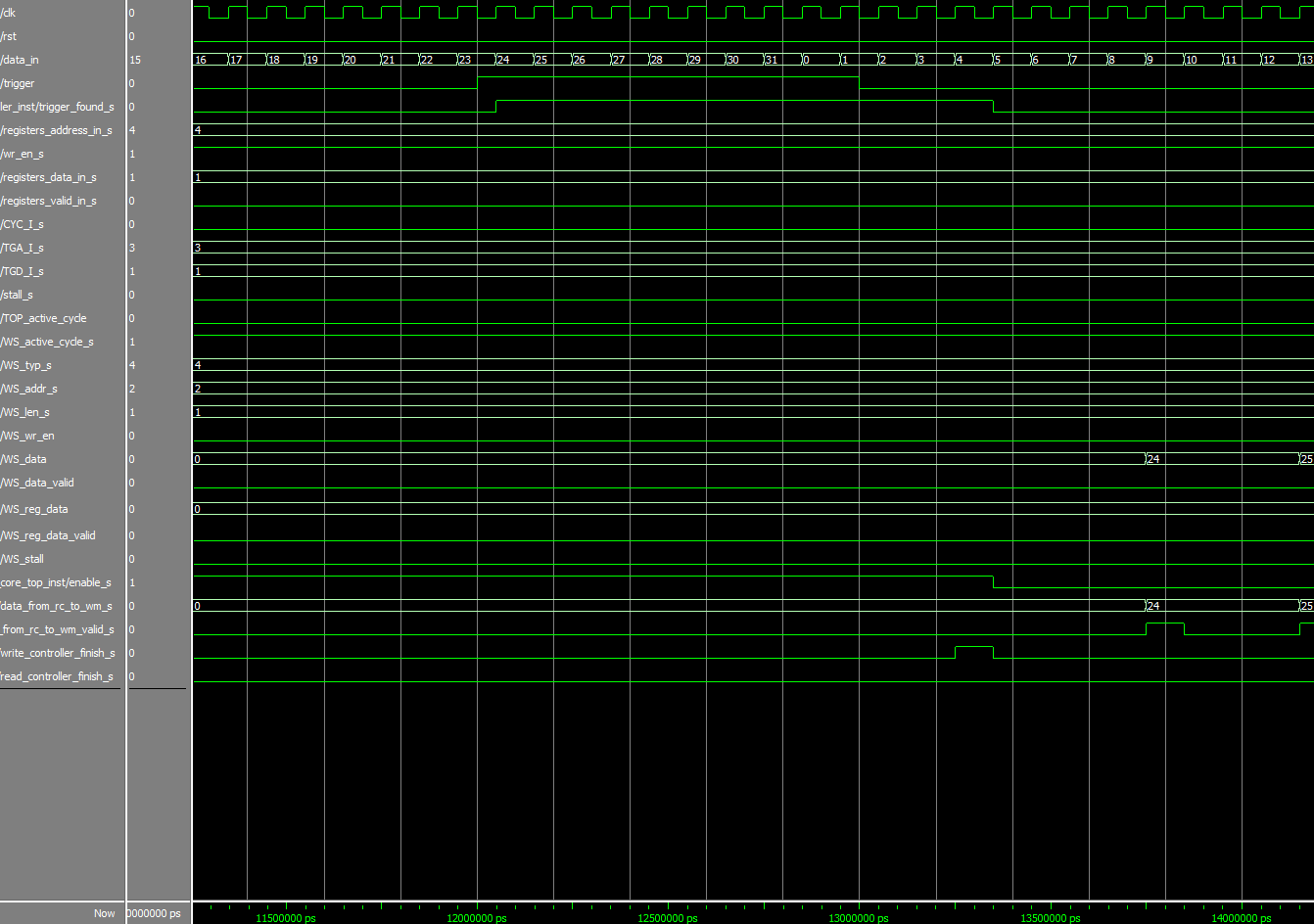
Simulation:



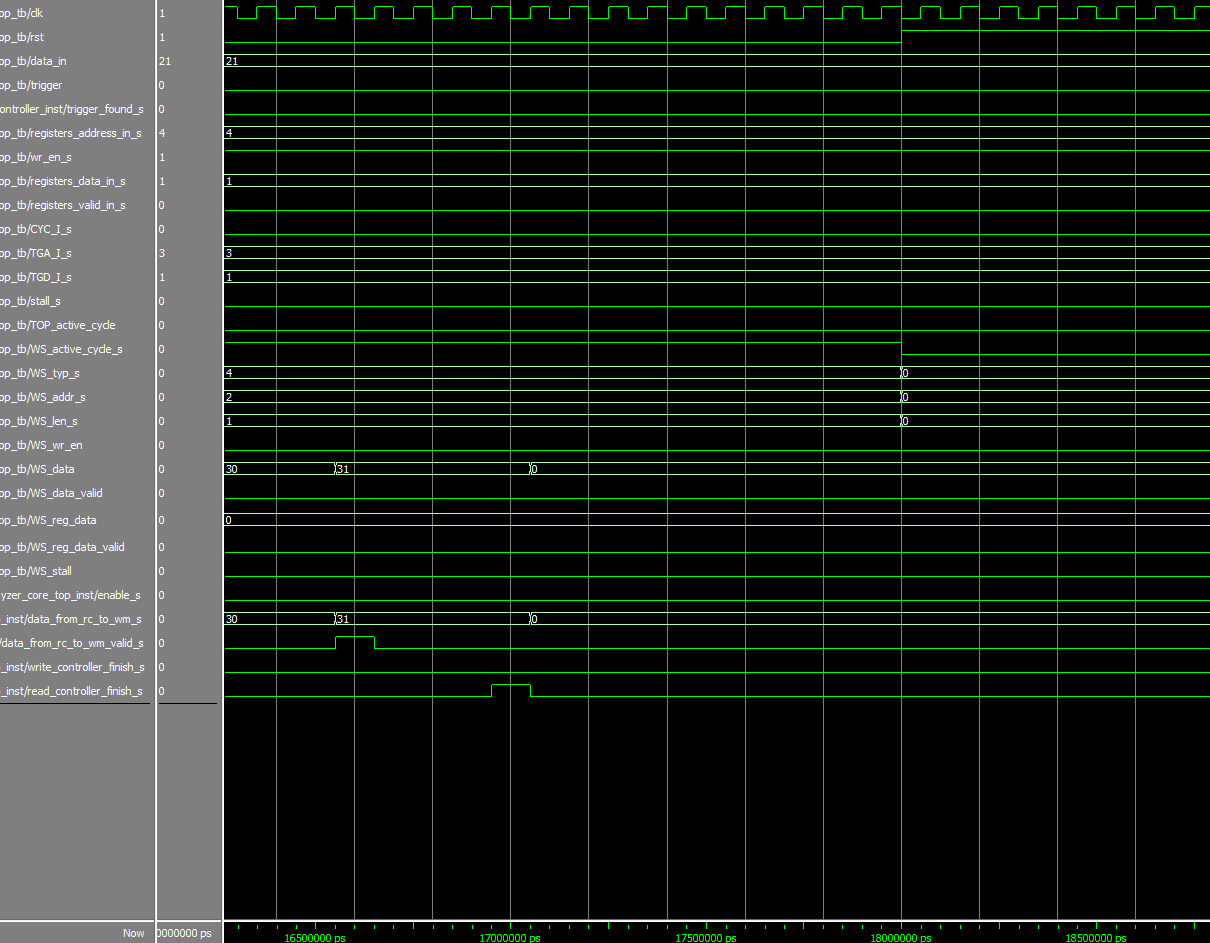












Analysis:

Number of signals is 5 -> 2^5 = 32 therefor the input data is now between 0-32, recording depth is 3 -> we record 2^3 = 8 samples of each signal.

In the first cycle position is 100 and all the data is recorded before the trigger (trigger not included), trigger found at data 23 so the output data is 15-22.

In the second cycle we change the position to 0 and all the data is recorded after the trigger (trigger included), trigger found at data 24 so the output is 24-31.

TEST NUMBER 4

Generics values:

|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 7 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 6 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

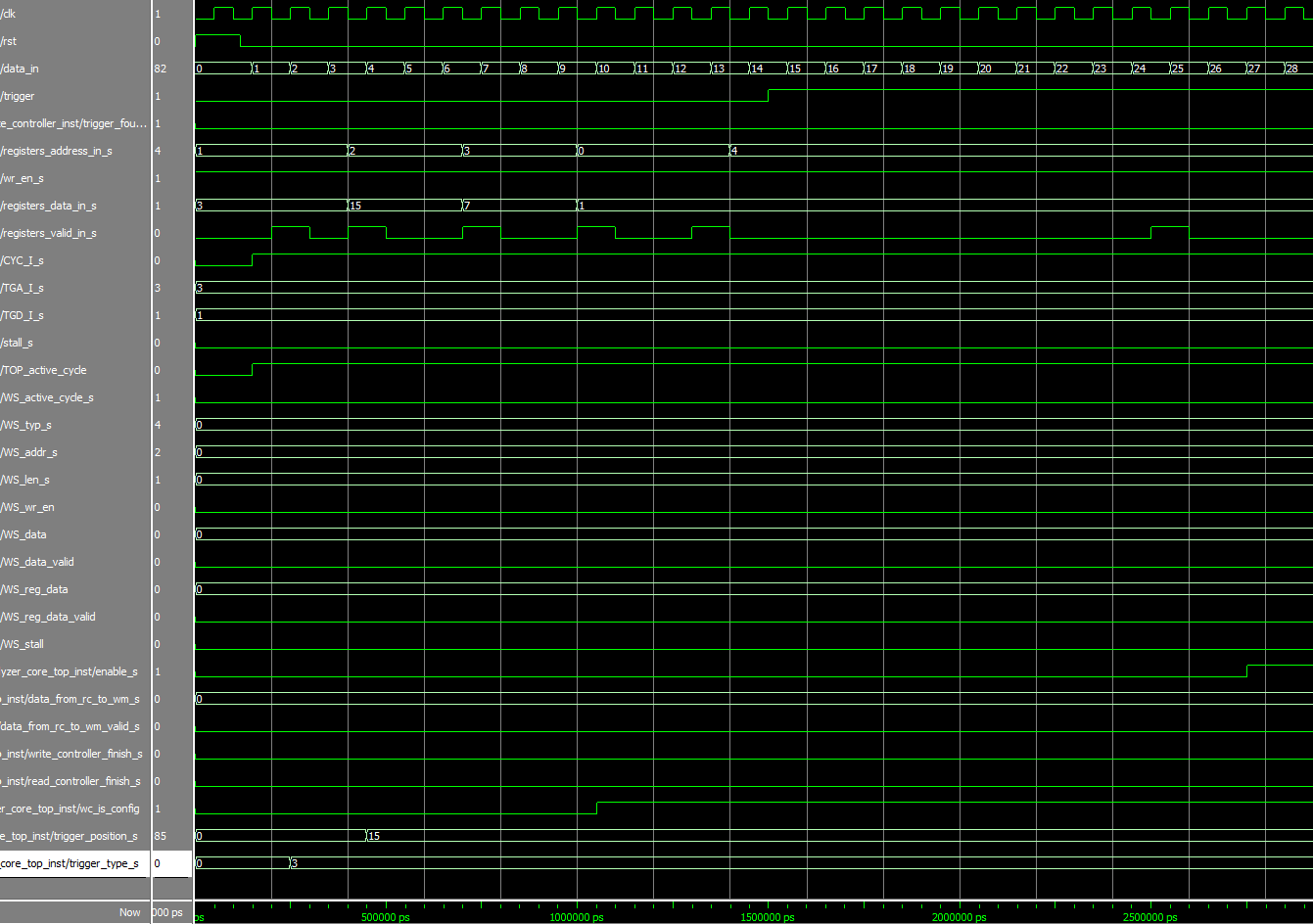
Explanation:

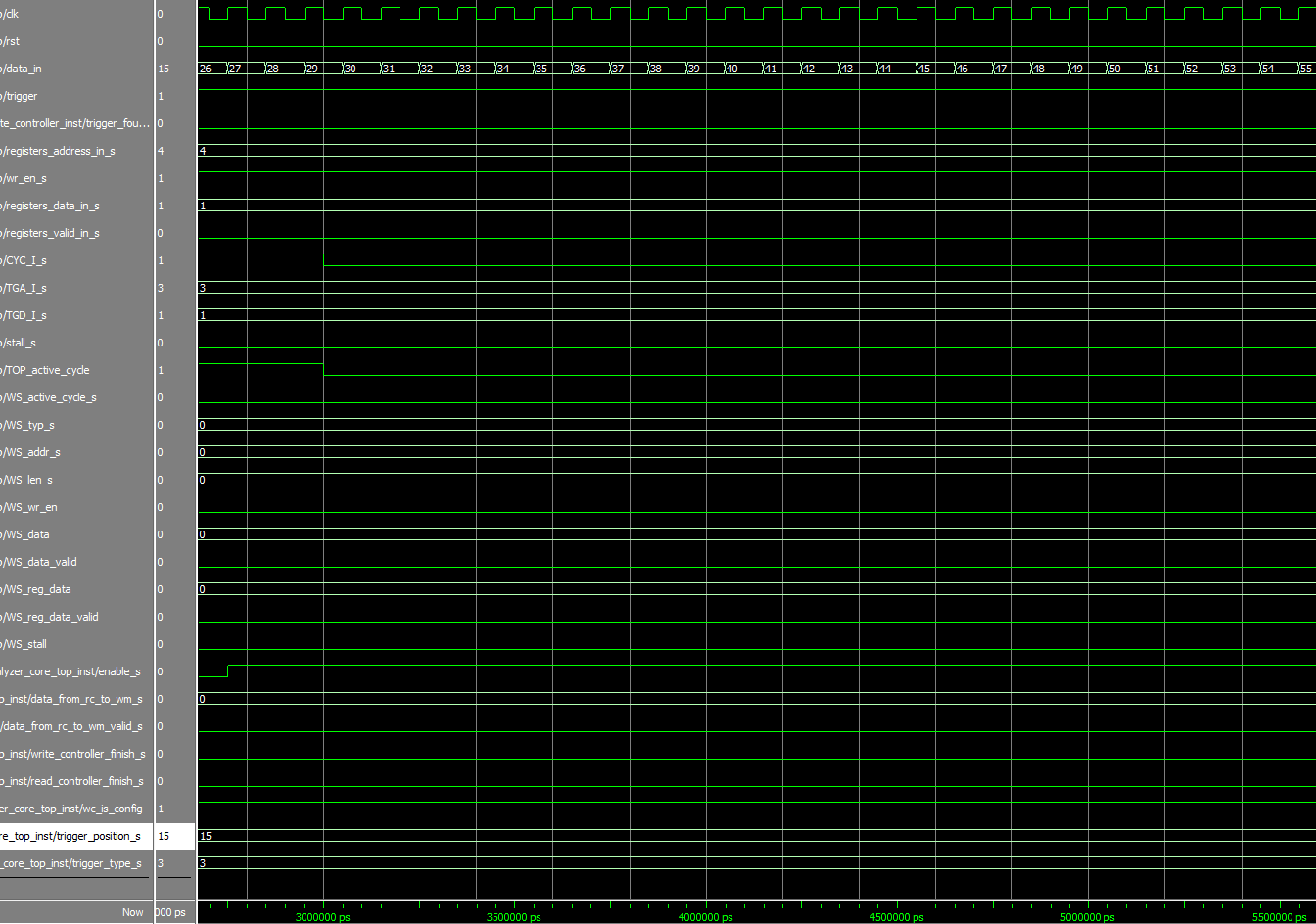
In this simulation we check two trigger positions that are not complete, 15% and 85%.

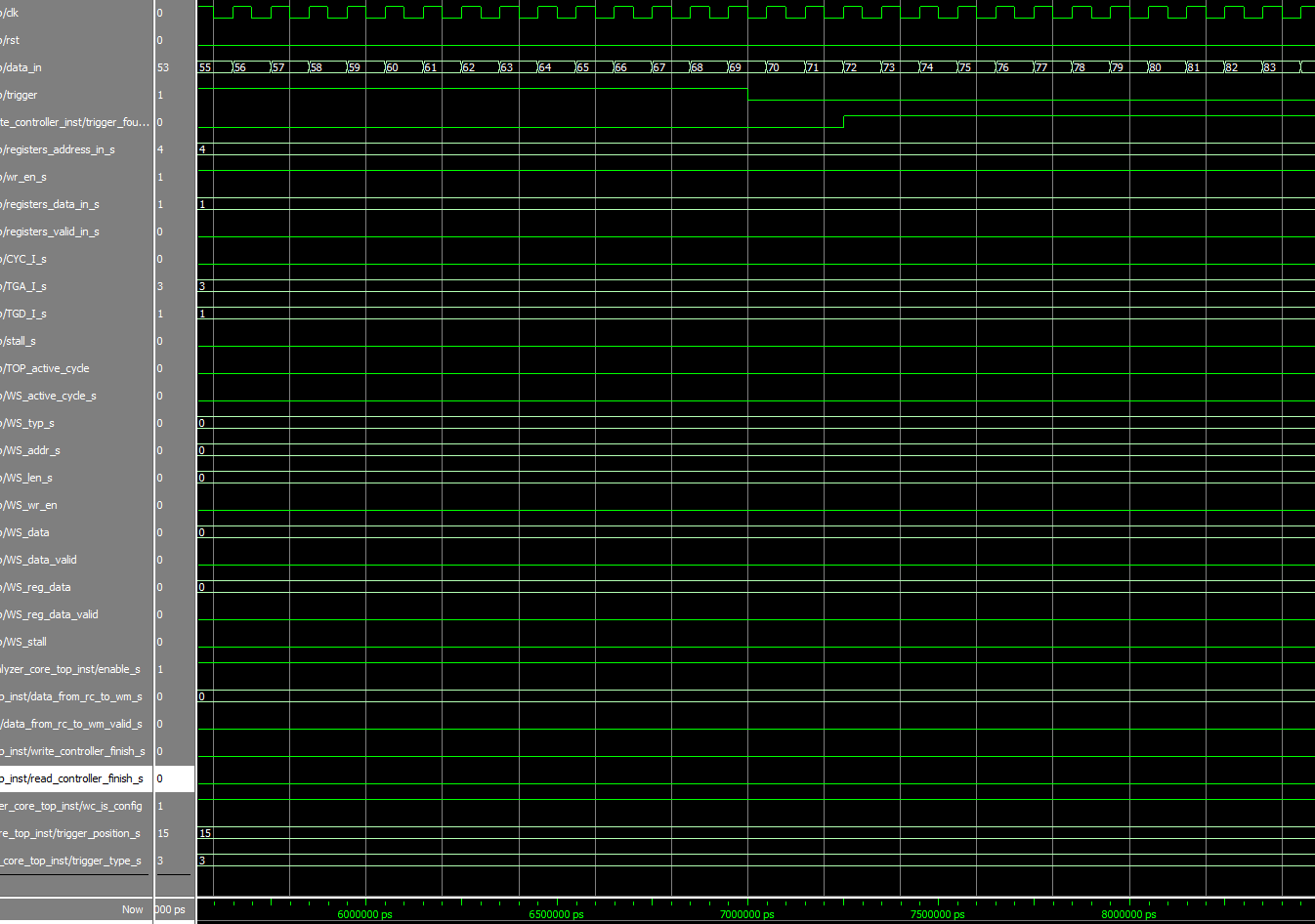
We also change the trigger type, from 'zeroes' (3 low in a row) in the first one to rise in the second.

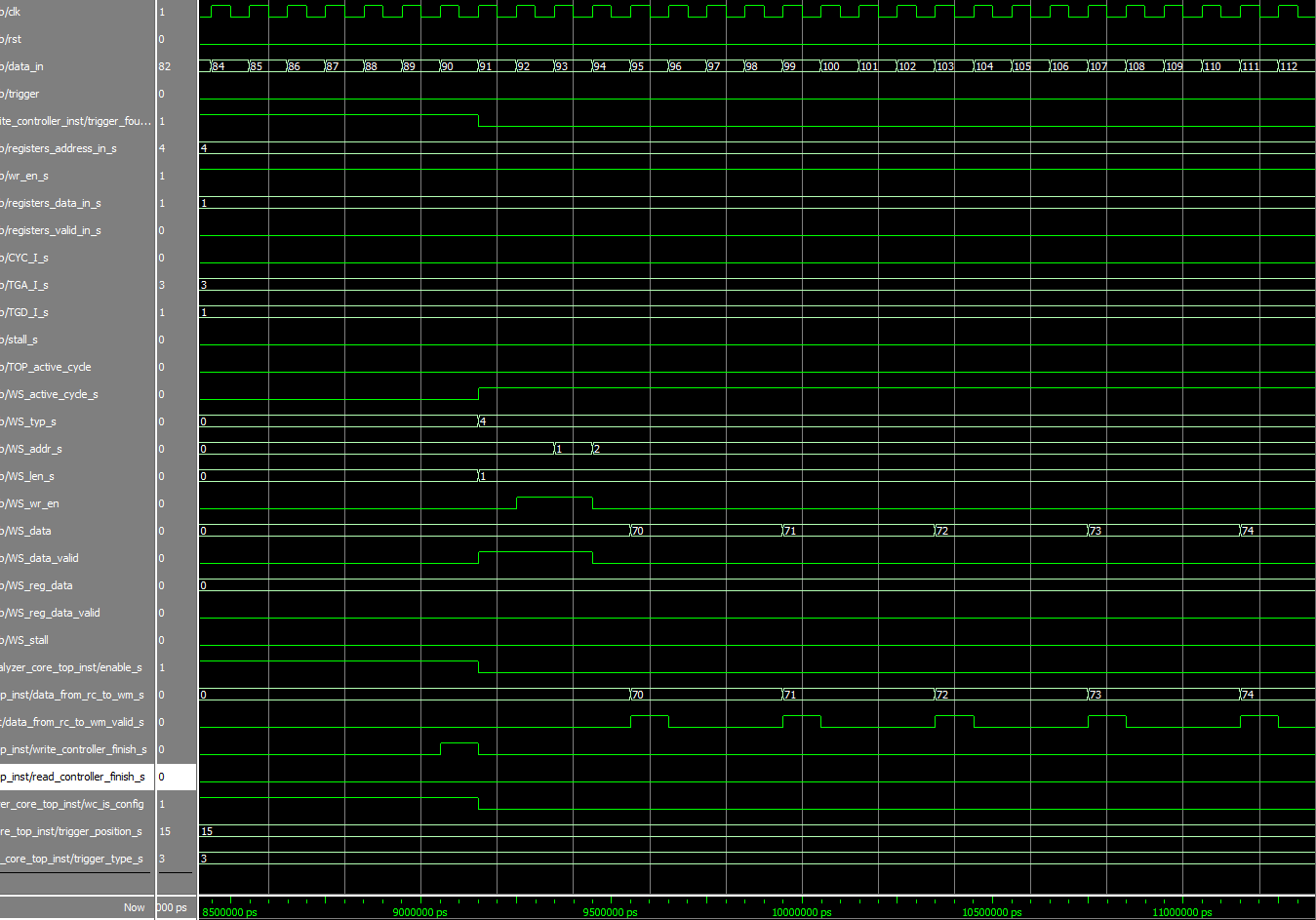
In the end we rise the RESET signal to show that the system is respond to that signal.

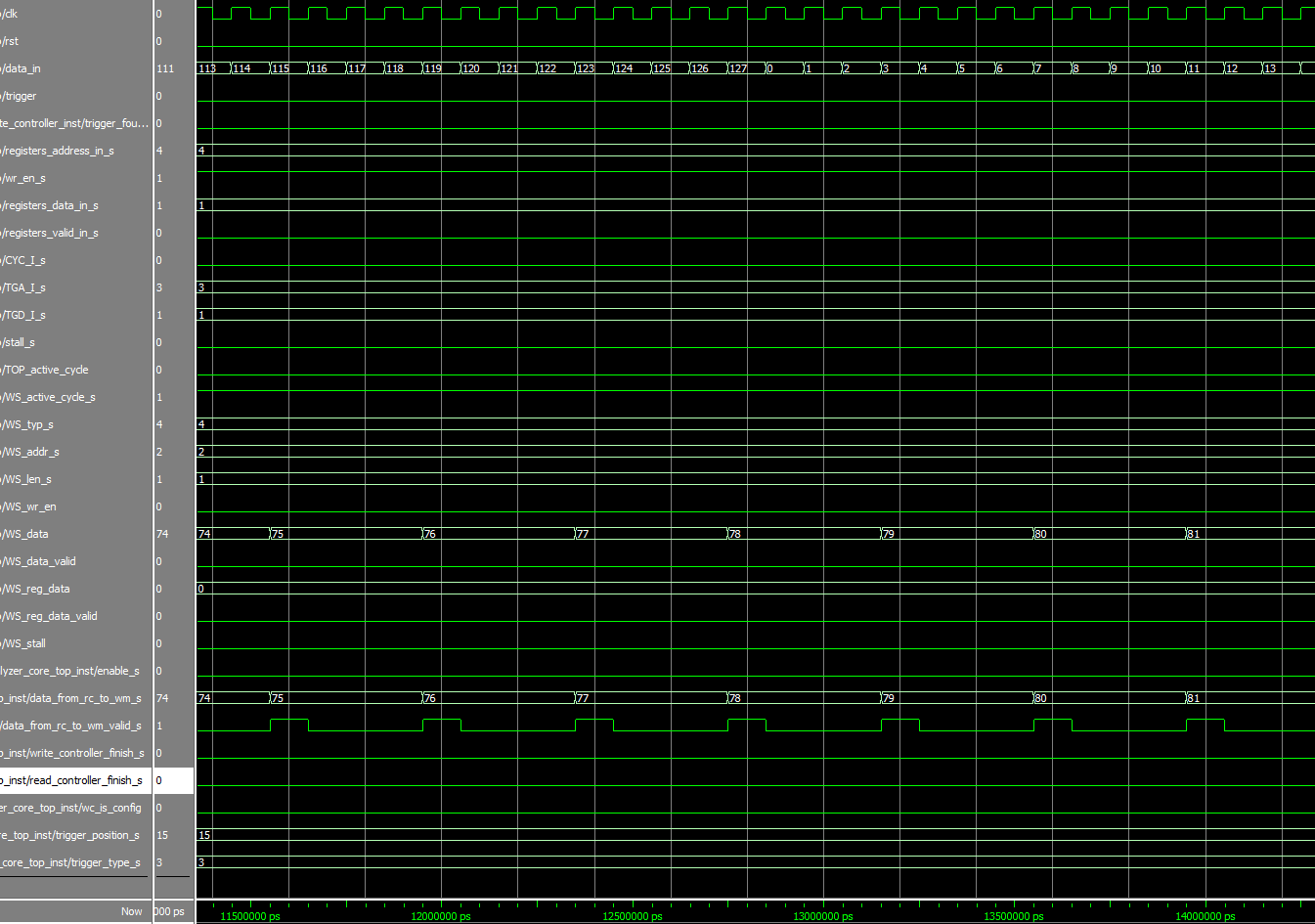
Simulation:

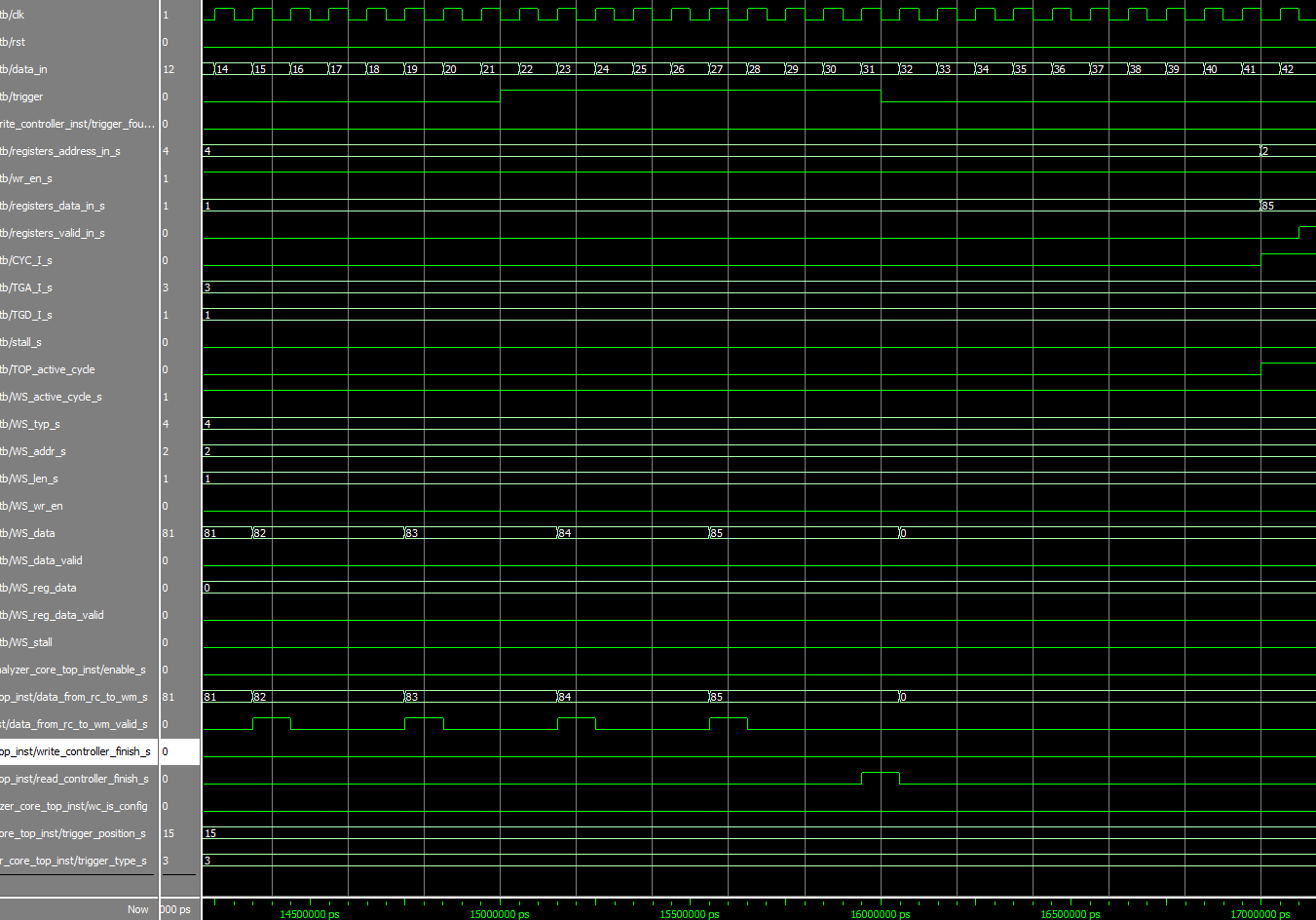


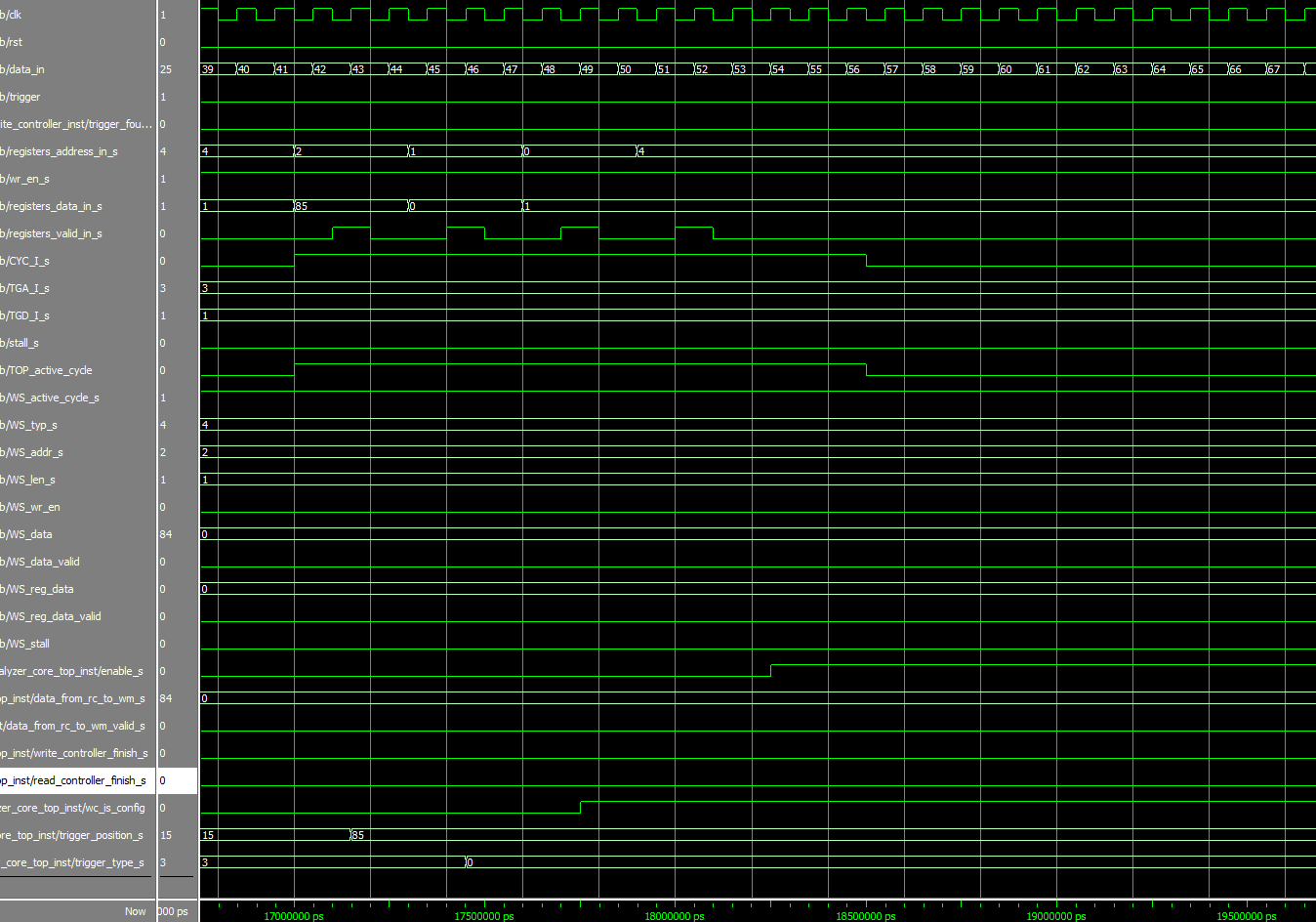


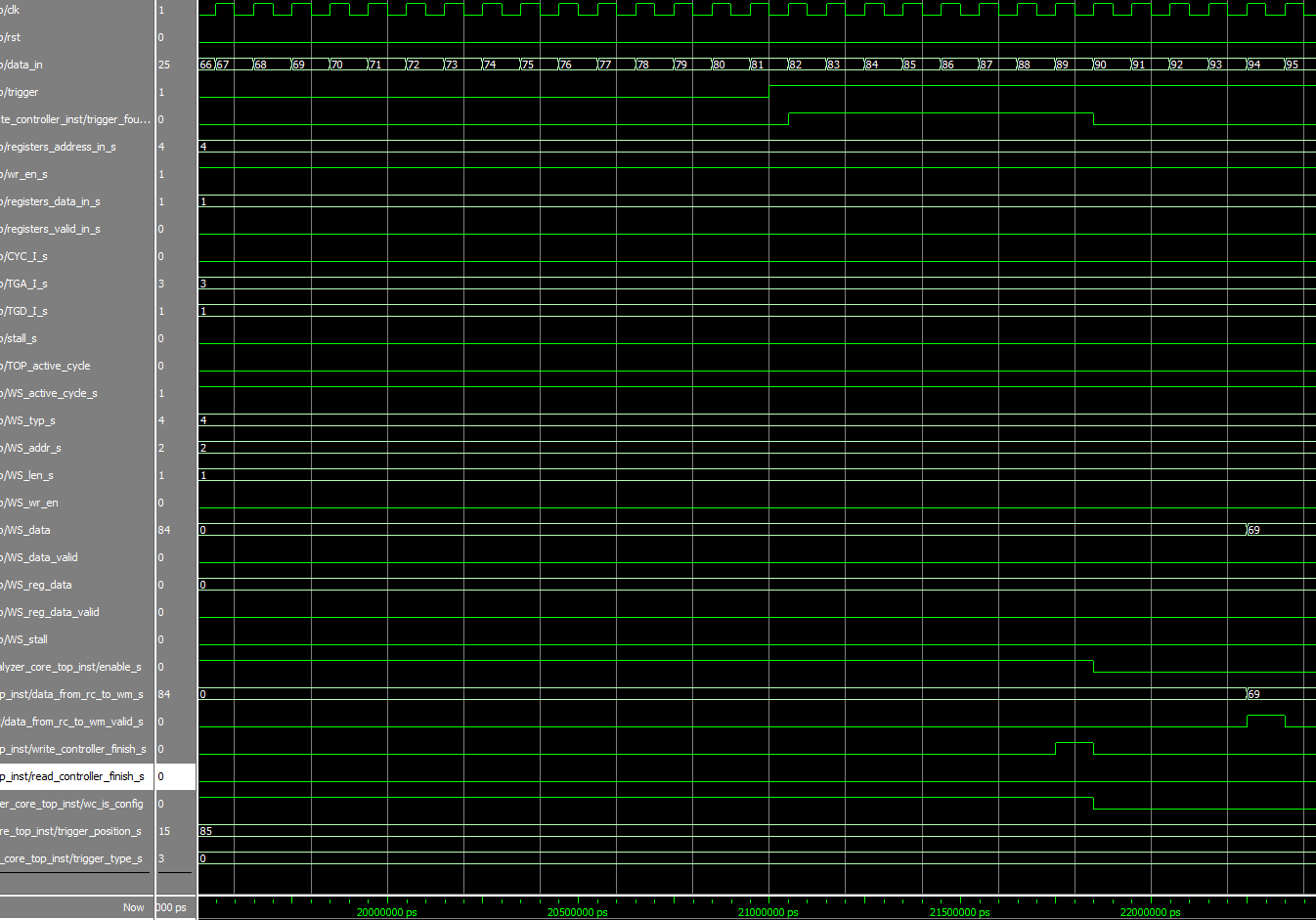


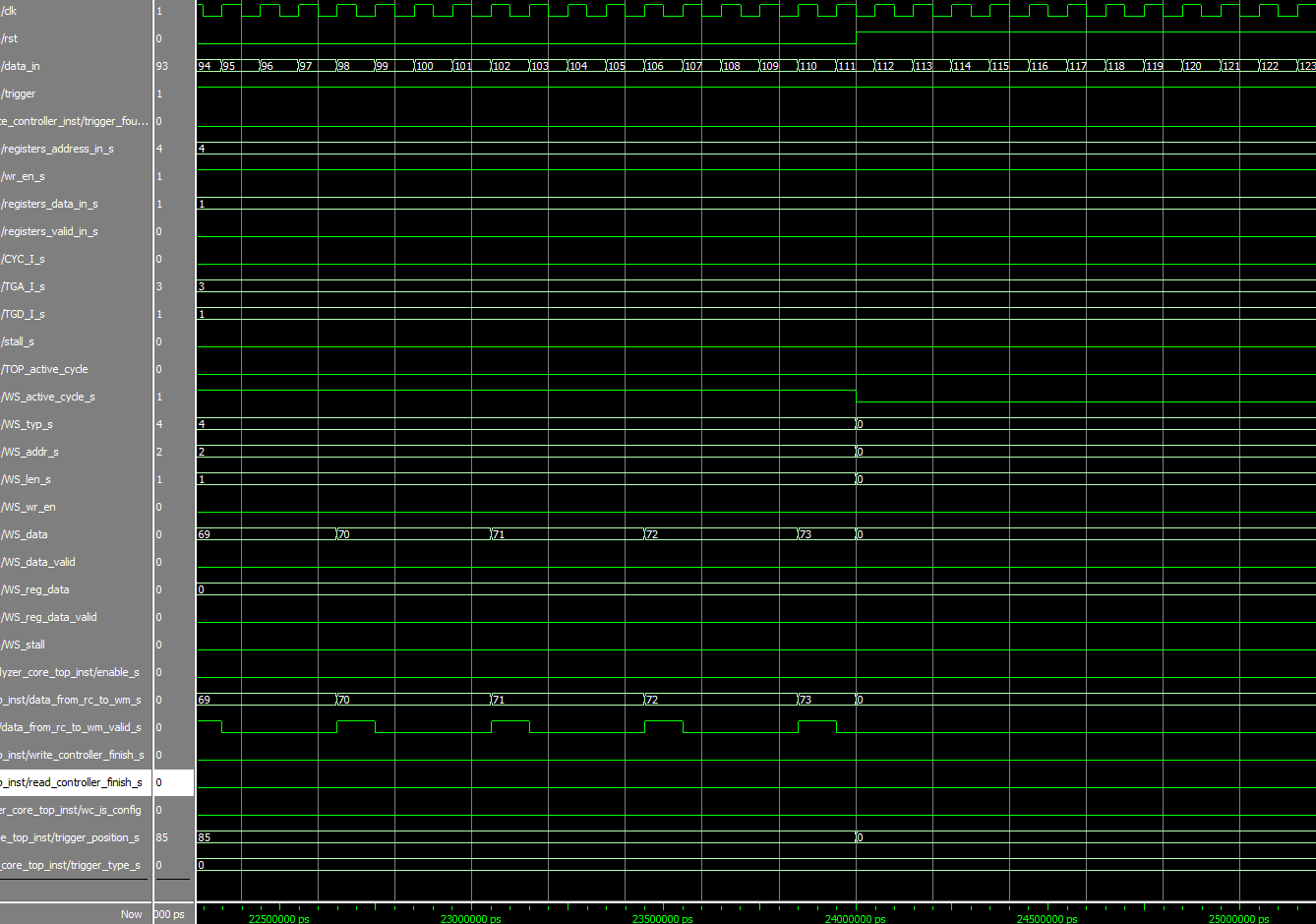












Analysis:

Like all the other simulations, we can see that the system is getting the initial configurations at the beginning. We record 7 signals and 2^4 = 16 samples from each one.

In the first cycle position is 'zeroes', and we seek for three low trigger at a row, we found that when the data is 72, and since the position is 15% we save 16\*(15/100) = 2.4 = 2 samples before the trigger ( 70 to 85).

In the second cycle we change the position to 85%, meaning that we save 16\* (85/100) = 13.6 = 13 samples before the trigger ( 69 to 84 ). We rise the RESET signal after 5 outputs to see that this signal is functional but the output at the start is according our expectation.

טסט מספר 5

ערכי ג'נריק:

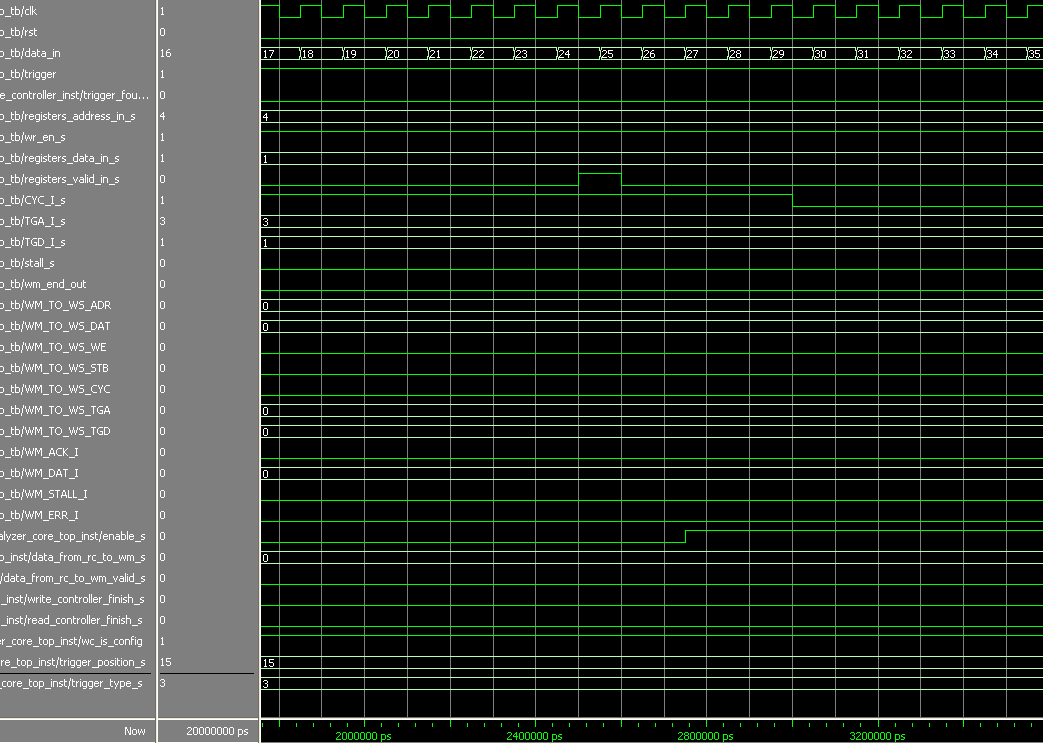
|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 3 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 6 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

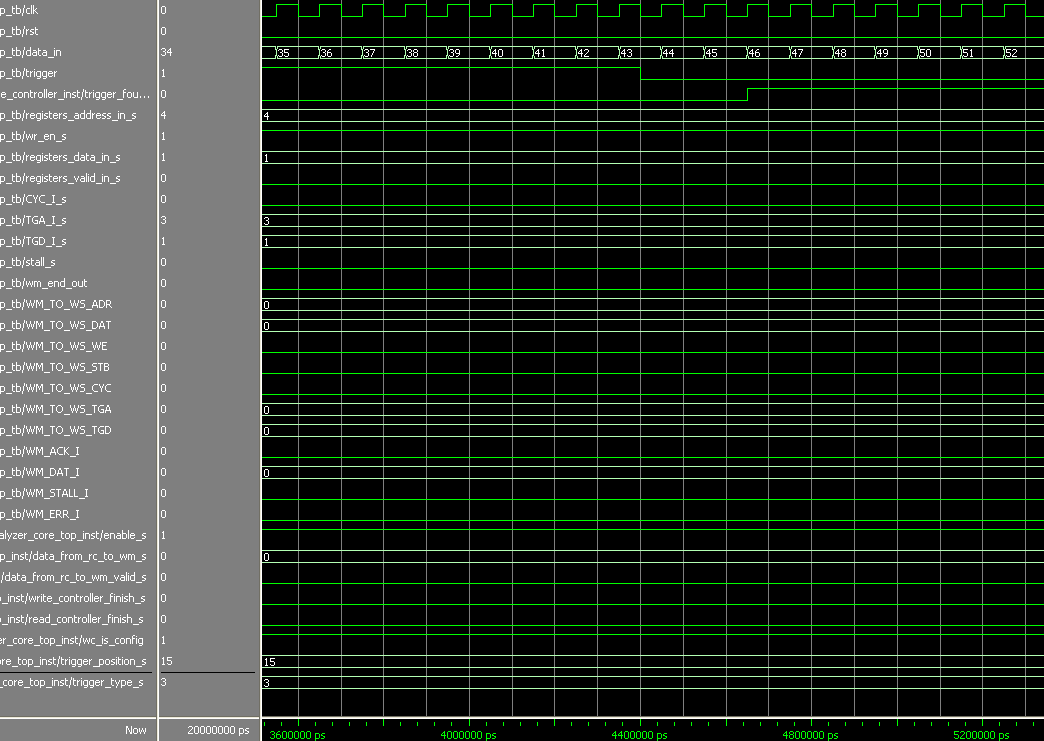
תיאור הבדיקה:

בדיקת אות RESET. בתחילת הבדיקה אנו מקנפגים את המערכת לקונפיגורציה ראשונה בה מיקום הטריגר הוא 15% וסוג הטריגר הוא נמוך, באמצע הוצאת המידע אנו מעלים את אות ה RESET ומקנפגים את המערכת לקונפיגורציה שונה בה המיקום הוא 85% והסוג הוא עלייה ולאחר מכן מוודאים שהמערכת מזהה שוב את עליית הטריגר.

תוצאות הסימולציה:

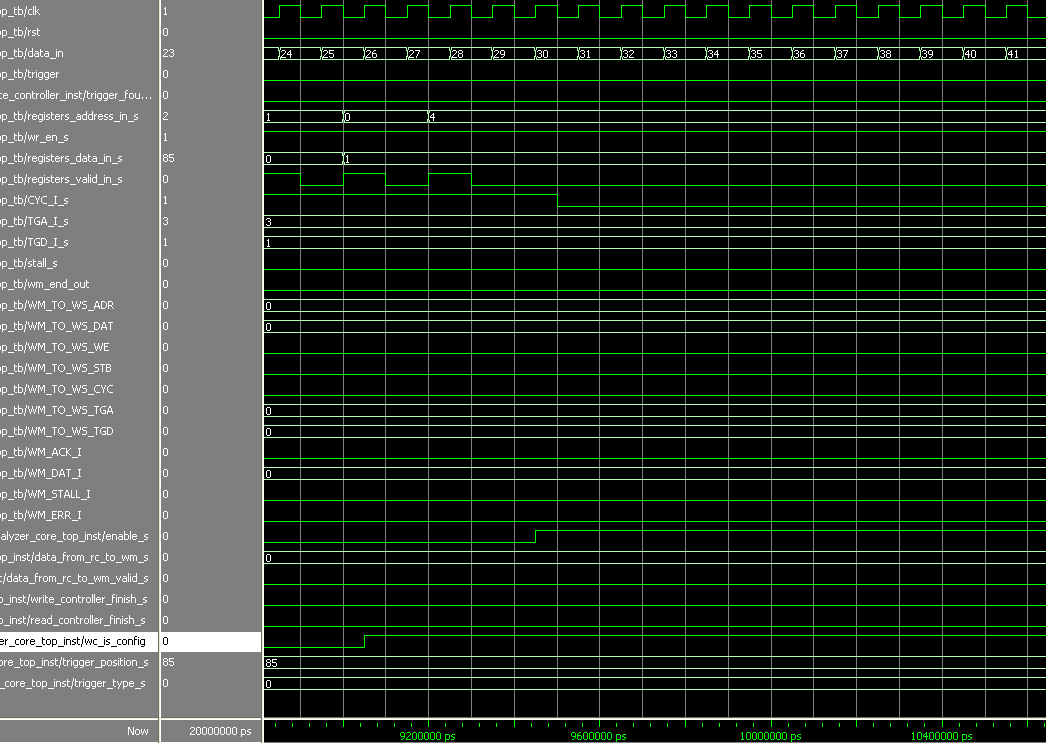




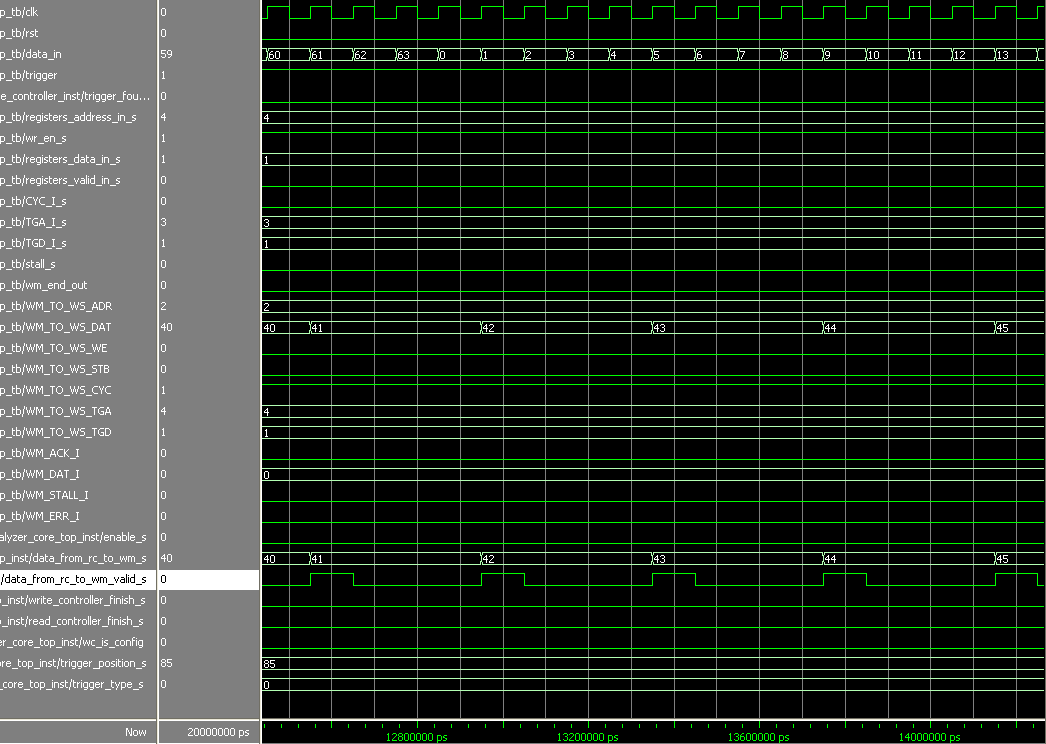














ניתוח:

בתחילה מתבצע קינפוג למערכת כפי שמוסבר, לאחר מכן אות טריגר יורד ומזוהה לאחר שלושה מחזורי שעון, כנדרש. לאחר שהמידע מתחיל לצאת אנו מעלים את אות RESET והמערכת חוזרת למצב התחלתי (ניתן לראות כי מיקום וסוג טריגר מקבלים ערך אפס) ומתבצע קינפוג שונה של המערכת כאשר כעת מיקום הטריגר הוא 85% וסוג הטריגר הוא עלייה, כנדרש לאחר עליית הטריגר המערכת מזהה זאת, מסיימת להקליט את הסיגנלים הנדרשים ומוציאה אותם כנדרש ולבסוף מסיימת את מעגל ההקלטה (אות READ CONTROLLER FINISH עולה למשך מחזור שעון אחד) ומכאן המערכת מוכנה לקליטת ערכים חדשים.

טסט מספר 6

ערכי ג'נריק:

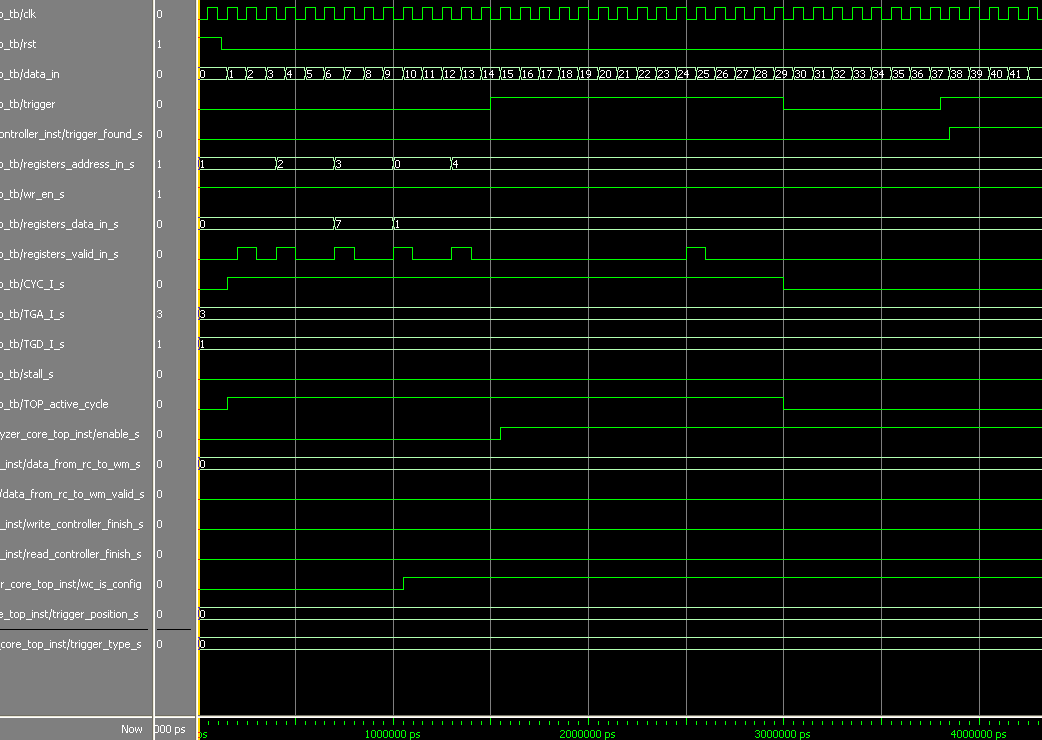
|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

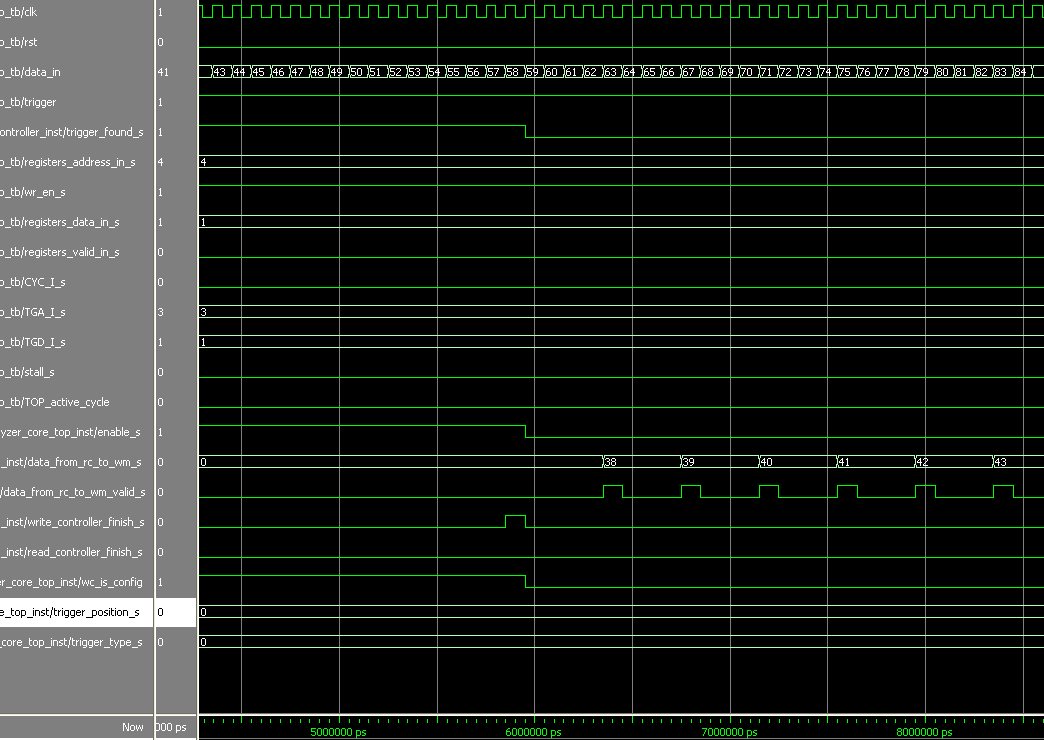
תיאור הבדיקה:

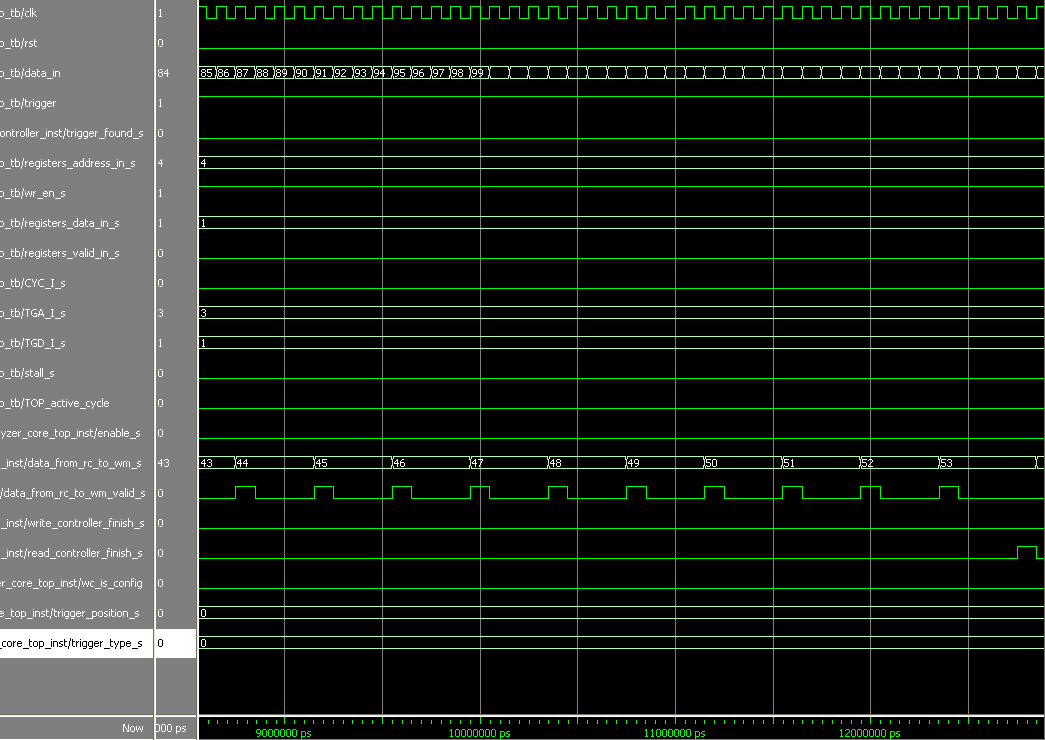
בדיקת סוגים שונים של מיקום הטריגר וסוג הטריגר.

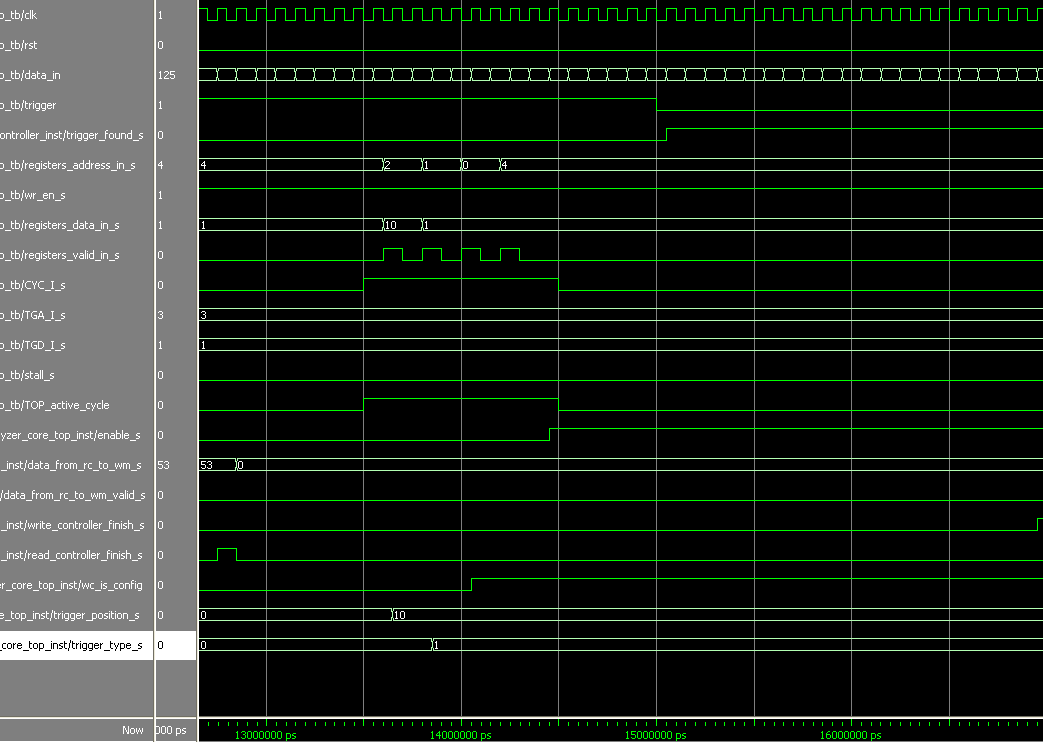
מהלך ראשון- מיקום 0 סוג עלייה, מהלך שני- מיקום 10 סוג ירידה, מהלך שלישי- מיקום 30 מהלך גבוה.

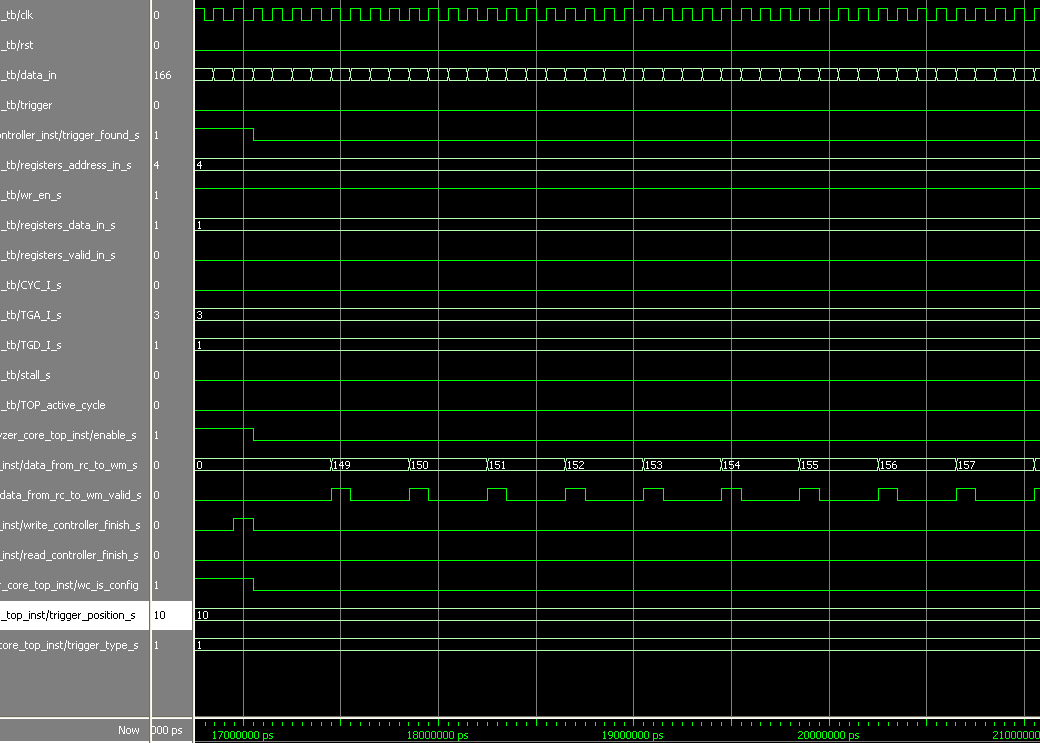
תוצאות הסימולציה:

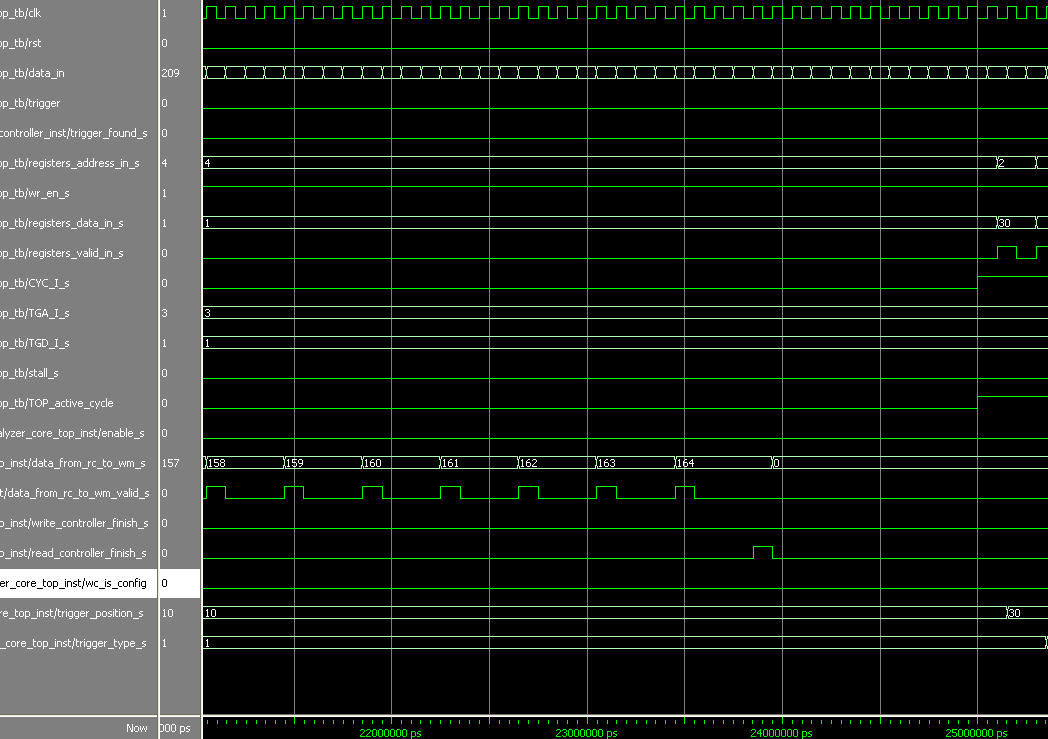


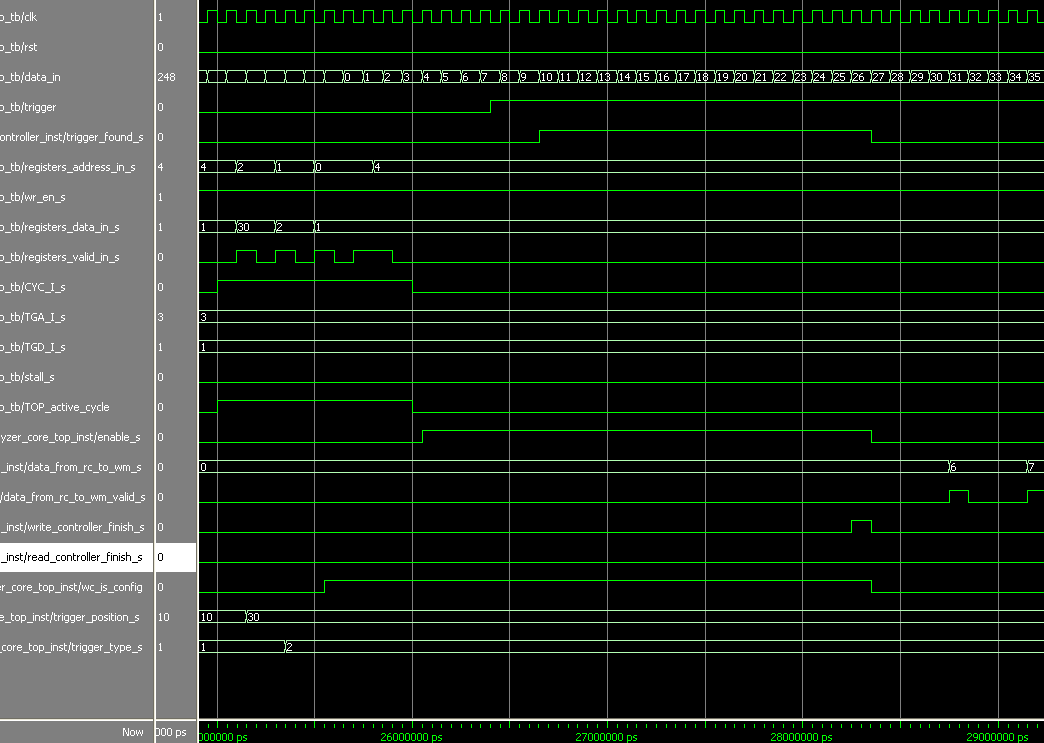


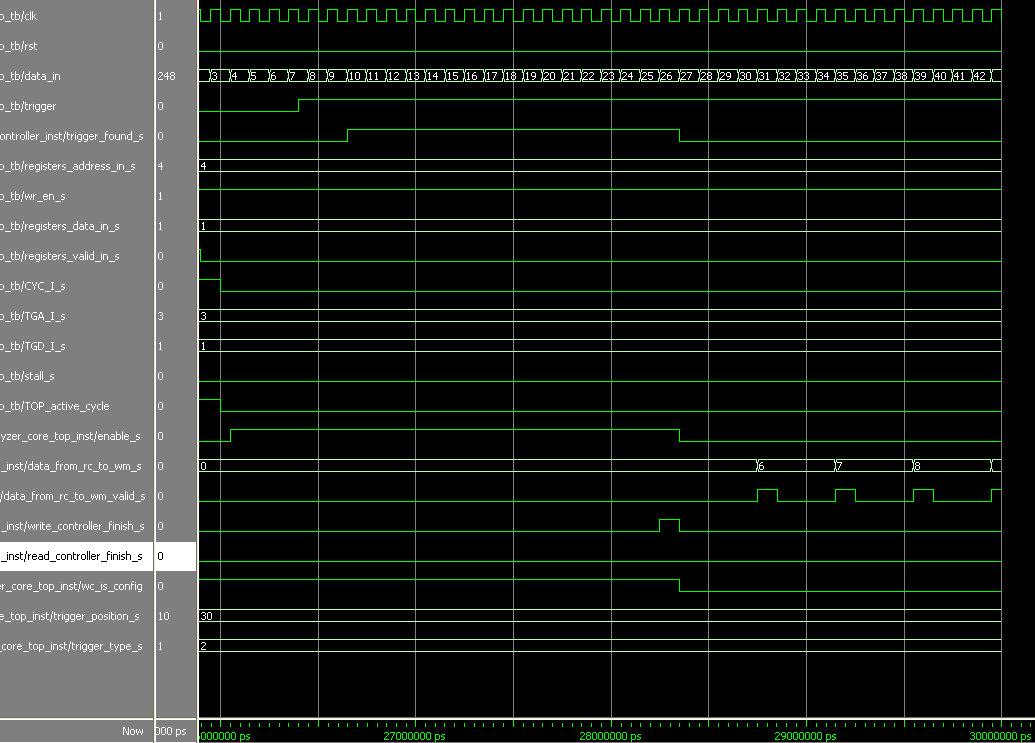












ניתוח:

מטרת סימולציה זו היא לבדוק את סוגי המיקום וסוגי הטריגר השונים. בקונפיגורציה הראשונה מיקום הטריגר הוא 0, כלומר כל המידע מוקלט לאחר עליית הטריגר, בנוסף סוג הטריגר הוא עלייה. כפי שניתן לראות הטריגר עולה כאשר המידע הוא 38 ולכן המידע היוצא לאחר מכן הוא 38 ועד 53 (record\_depth\_g הוא 4 ולכן אנו מקליטים 16 דגימות).

בקונפיגורציה הבאה מיקום הטריגר הוא 10 (כלומר ) וסוג הטריגר הוא ירידה, ניתן לראות כי אות הטריגר יורד כאשר המידע הוא 150 (לא נראה היטב בתמונה שהוצאנו אך זה ערכו בעת "מציאת" הטריגר) ולכן מוקלטת דגימה אחת לפני הטריגר ושאר הדגימות מוקלטות לאחריו.

בקונפיגורציה השלישית מיקום הטריגר הוא 30 וסוג הטריגר הוא גבוה (שלוש דגימות גבוה רצוף), הטריגר עולה כאשר המידע הוא 10 ולכן המידע היוצא מתחיל מ 4 דגימות לפניו- כלומר מדגימה שערכה 6. (את הדגימה הזו אנו קוטעים לפני שהיא מסתיימת כיוון שהראנו כי המידע היוצא הוא מדוייק והטריגר מזוהה כנדרש.

טסט מספר 7

ערכי ג'נריק:

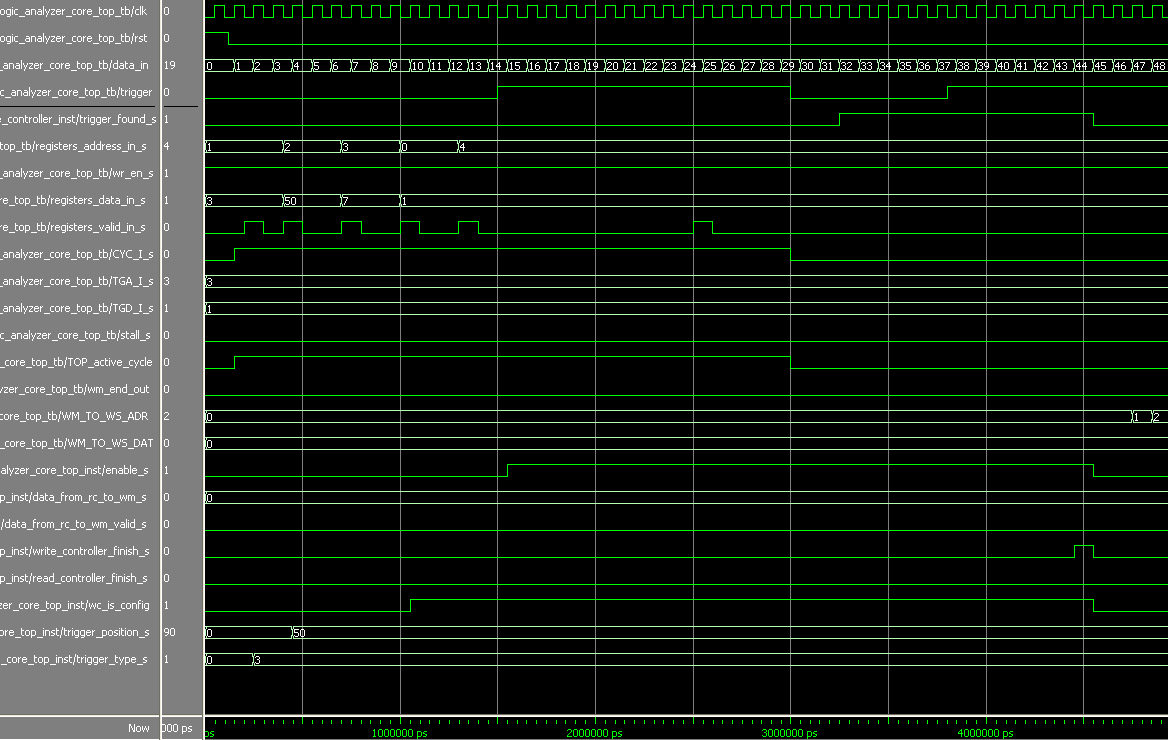
|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

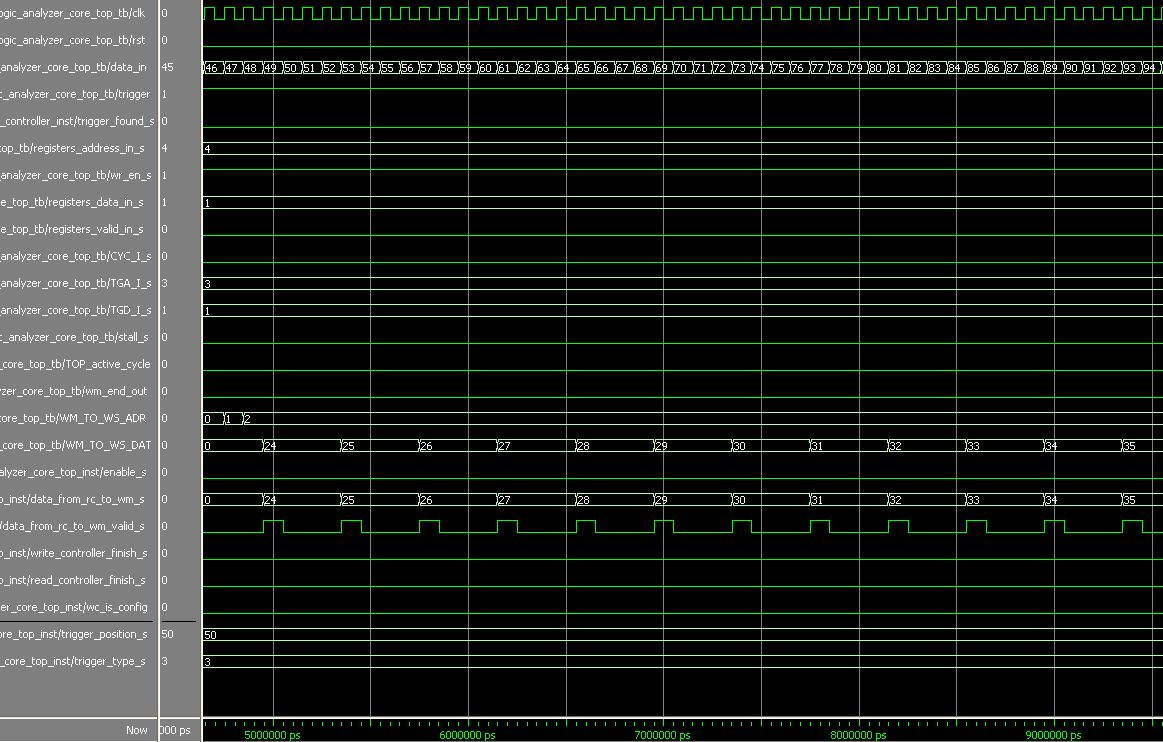
תיאור הבדיקה:

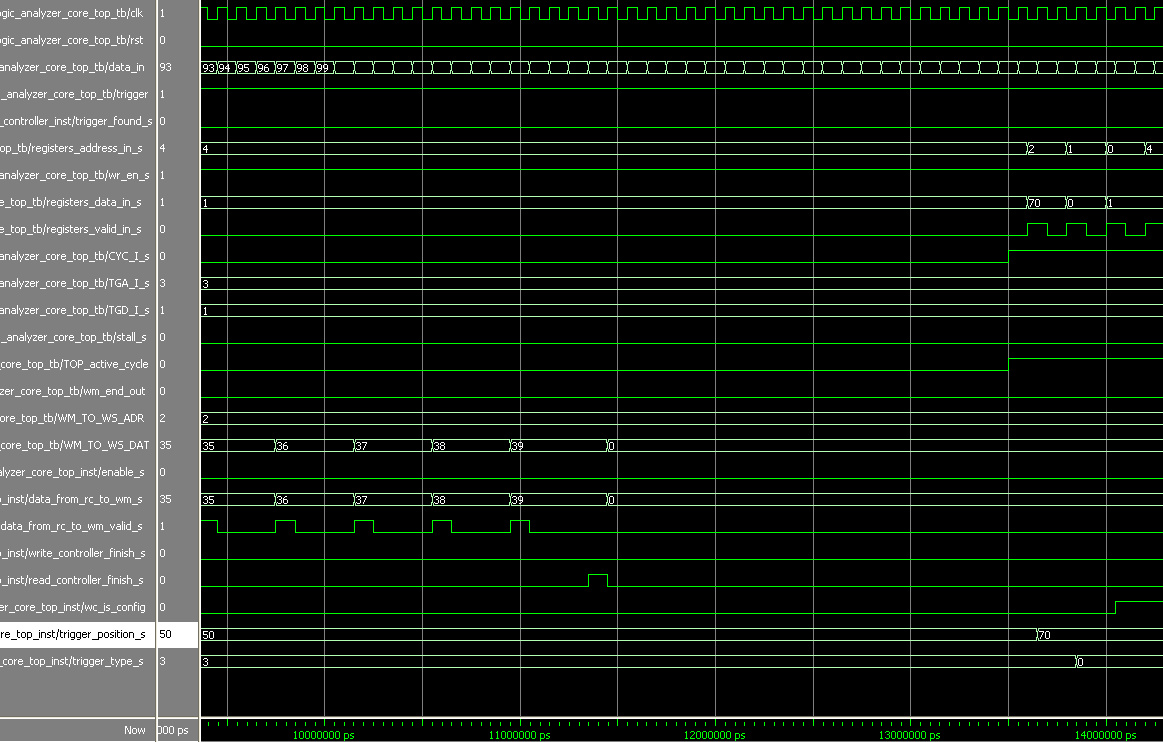
המשך בדיקת סוגים שונים של מיקום הטריגר וסוג הטריגר.

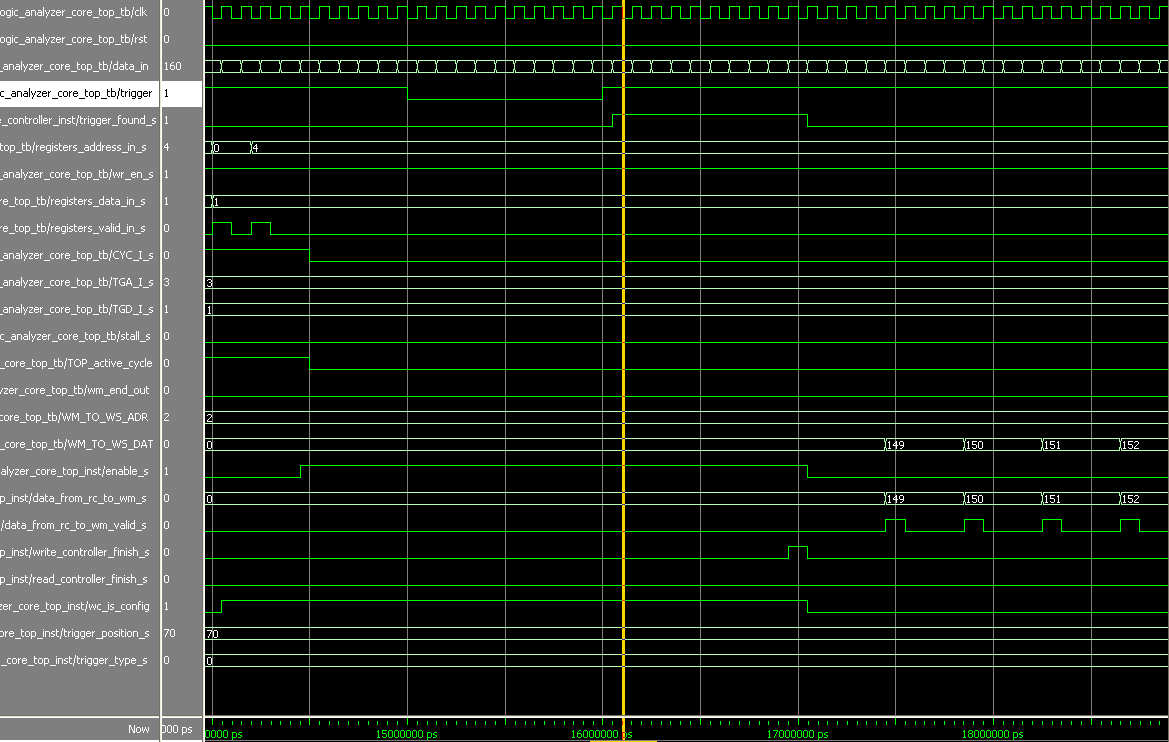
מהלך ראשון- מיקום 50 סוג נמוך, מהלך שני- מיקום 70 סוג עליה, מהלך שלישי- מיקום 90 סוג ירידה.

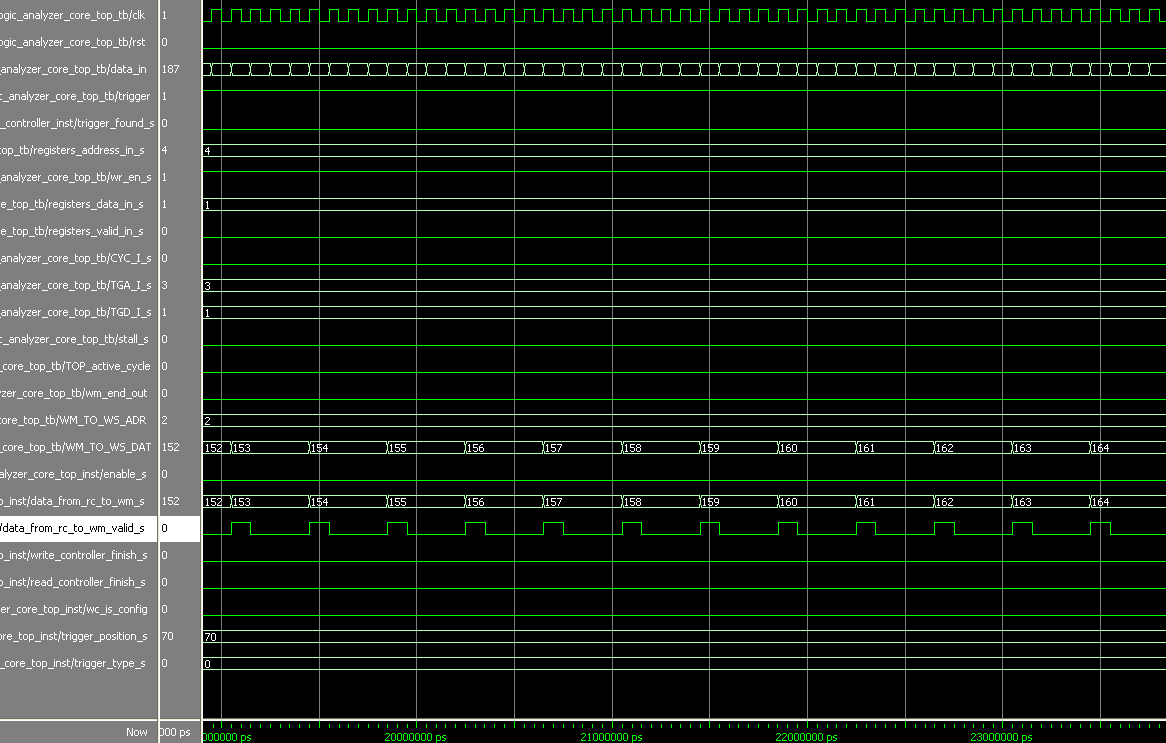
תוצאות הסימולציה:

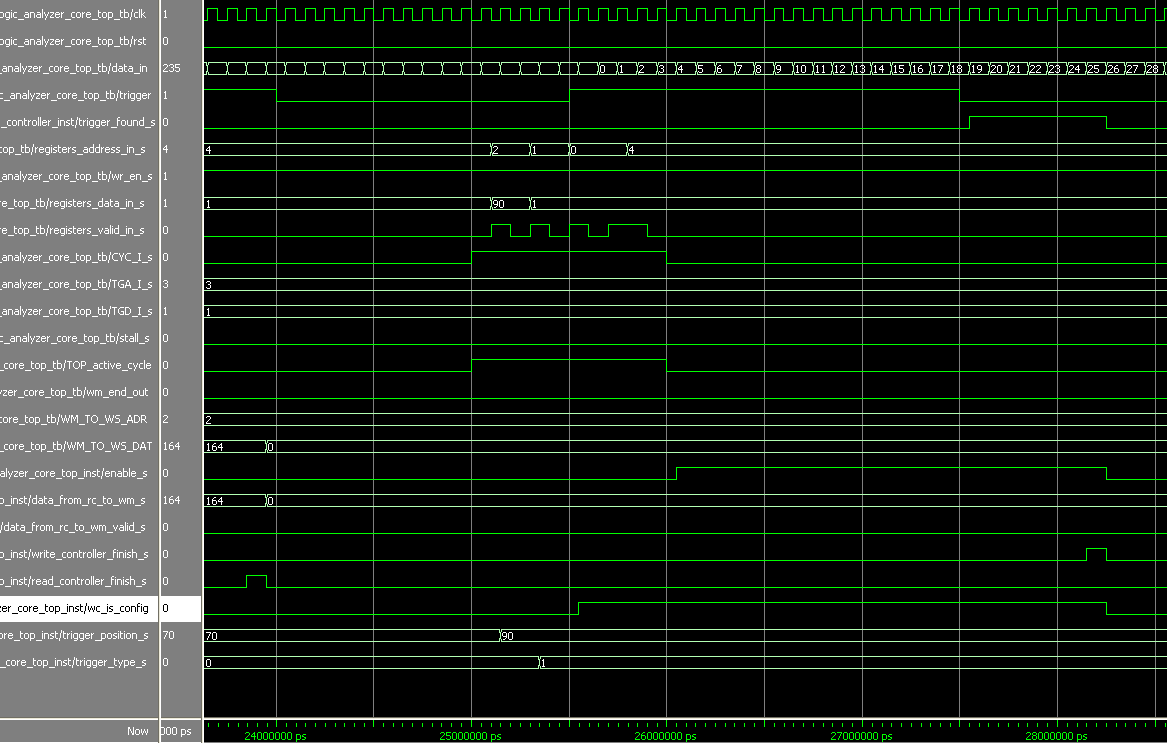


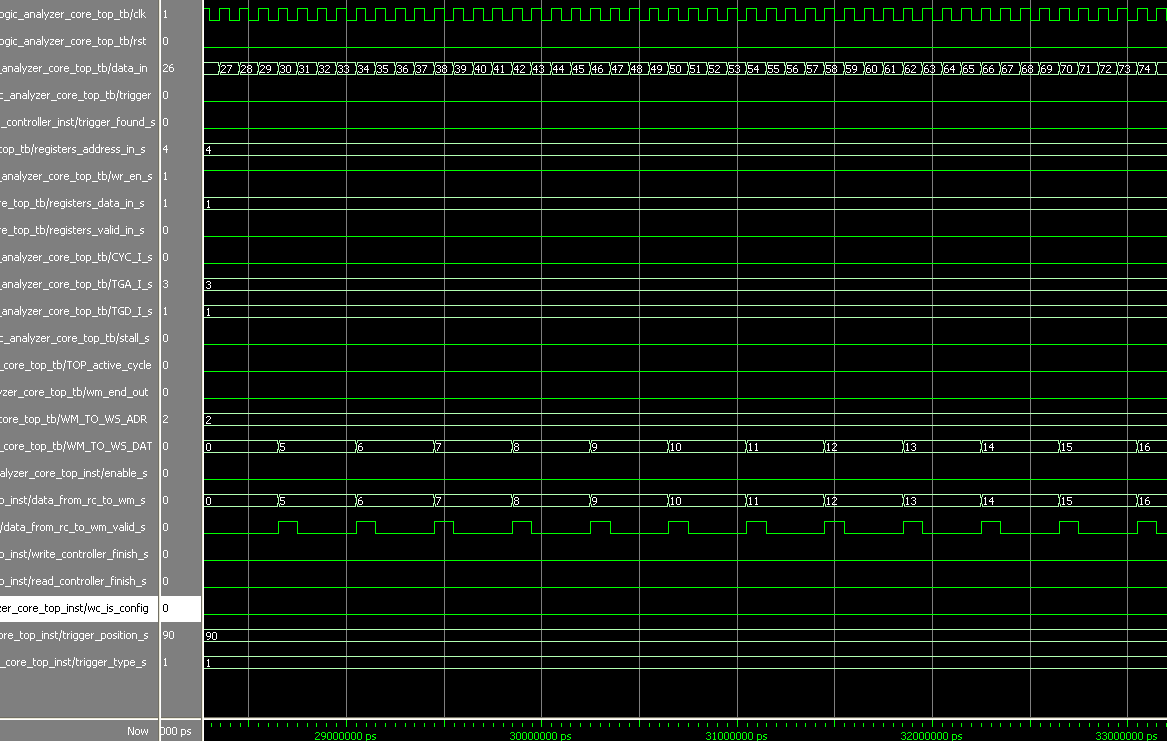


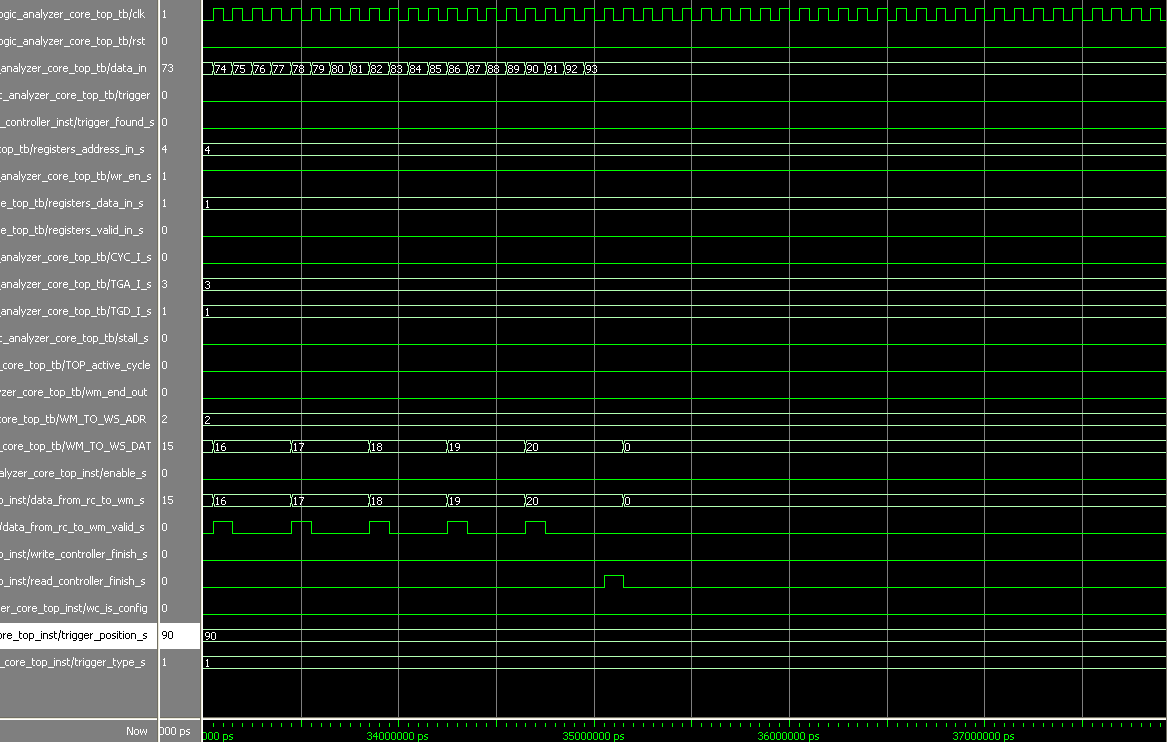












ניתוח:

טסט זה הוא המשכו של הטסט הקודם. ניתן לראות כי עליית הטריגר והוצאת המידע מתרחשות בהתאם מלא לציפיות:

מהלך ראשון- המיקום הוא 50 והסוג הוא נמוך, ואכן אחרי 3 מחזורי שעון בהם הטריגר נמוך אנו מקבלים עליית טריגר במיקום 32 והמידע היוצא הוא בהתאם 8 דגימות (חצי מסך כל הדגימות ) ולכן המידע מתחיל לצאת במיקום 24.

במהלך השני סוג הטריגר הוא עלייה והמיקום הוא 70- כלומר  11 דגימות ישמרו לפני עליית הטריגר. הטריגר עולה במידע 160 ולכן המידע היוצא מתחיל מ 149.

במהלך השלישי סוג הטריגר הוא ירידה והמיקום הוא 90- וכאמור הטריגר נמצא במידע מספר 19 ולכן המידע היוצא מתחיל מתא מספר 5.

טסט מספר 8

ערכי ג'נריק:

|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 5 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

תיאור הבדיקה:

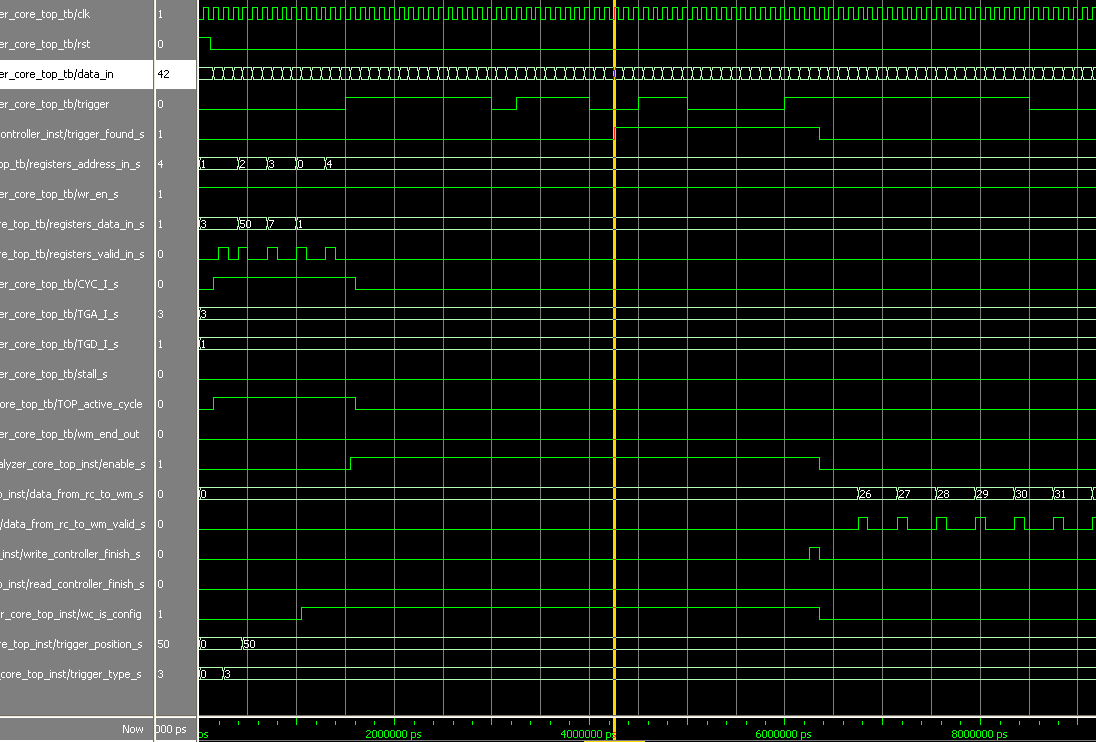
בבדיקה זו אנו מתמקדים בשינוי הקונפיגורציה לאחר שכבר עלה הטריגר, ובכך שעליית טריגר נוספת, לאחר שכבר מצאנו עליית טריגר או לאחר שכל המידע כבר יצא אך לא בוצע קינפוג נוסף, אינה תשפיע על המערכת.

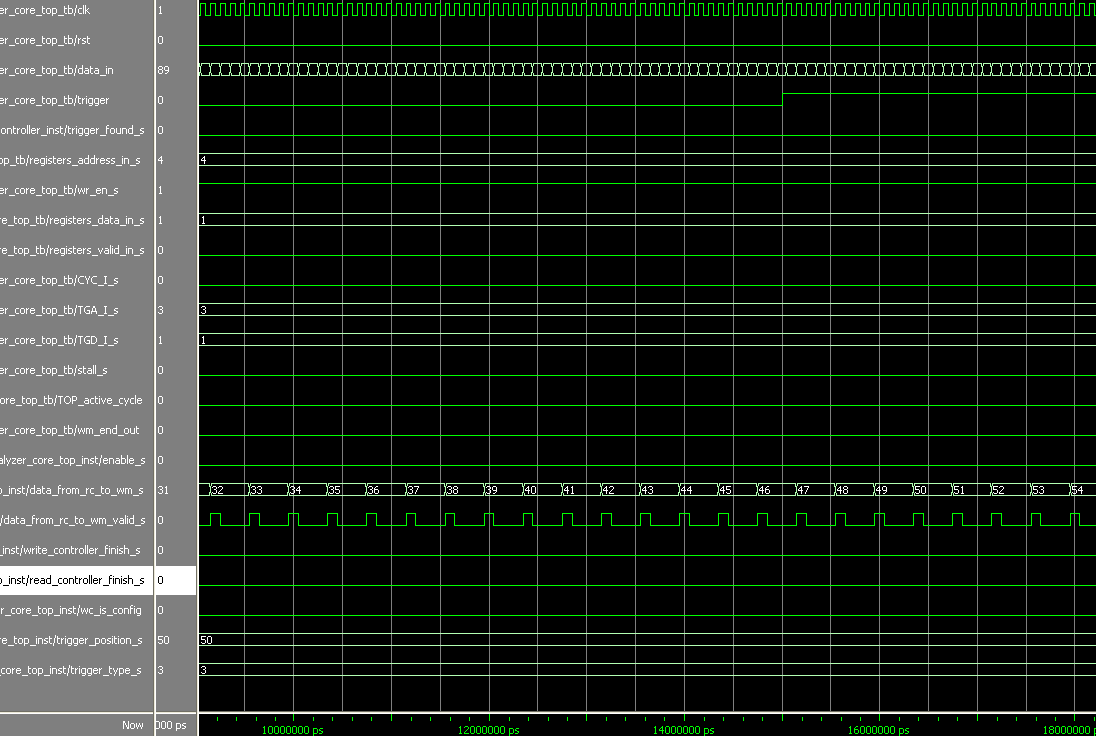
אנו בתחילה מבצעים קונפיגורציה מסוימת, ולאחר עליית טריגר והוצאת כל המידע, אנו רואים כי עליית טריגר נוספת לא גורמת ליציאת מידע לפני שמתבצע קינפוג נוסף.

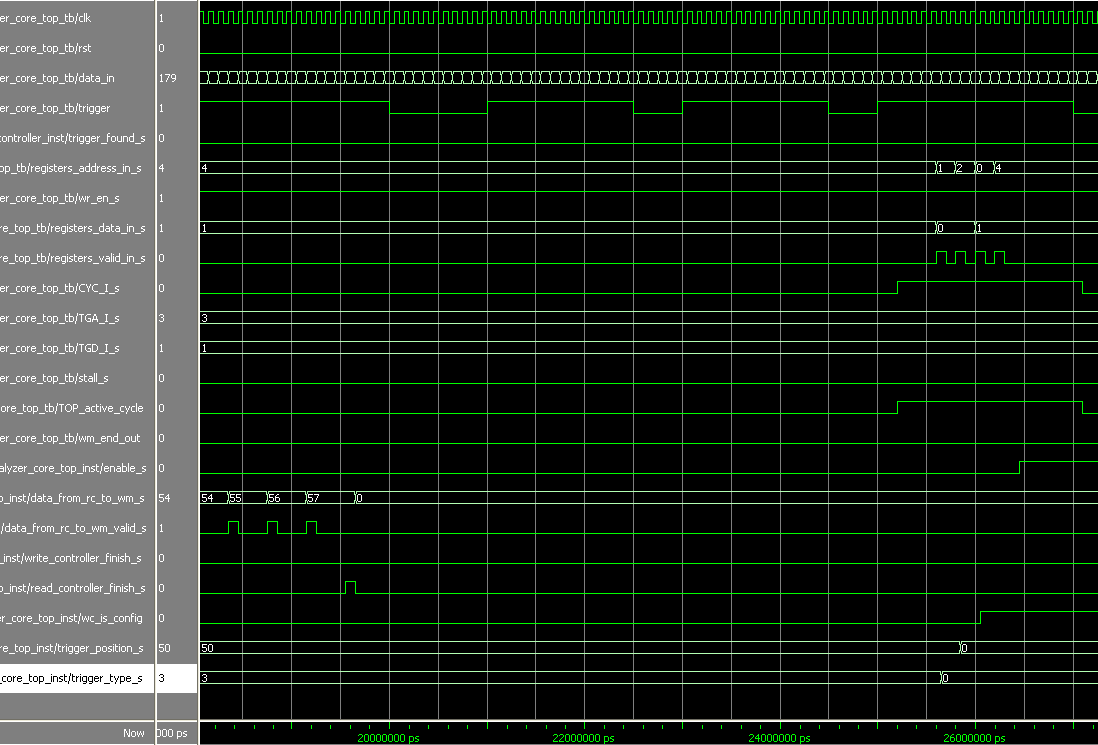
בנוסף אנו מקליטים 32 דגימות ומוודאים כי המערכת מגיבה היטב לערך זה (5 = record\_depth\_g).

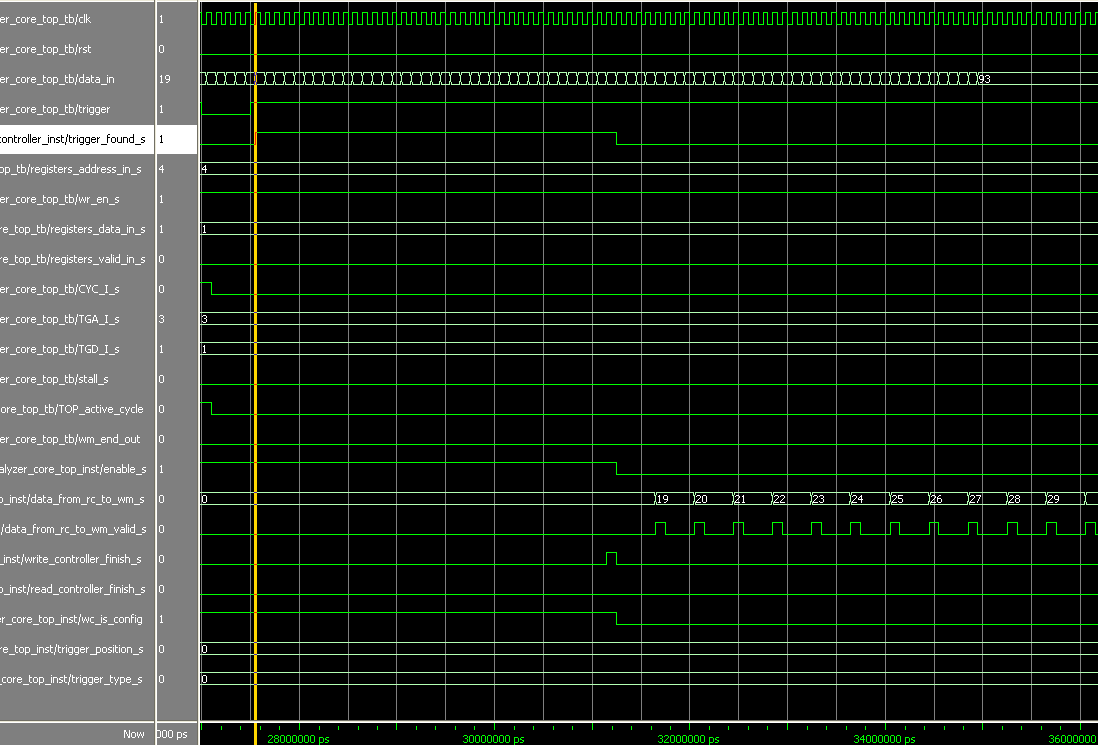
תוצאות הסימולציה:

(הקטנו את הרזולוציה כיוון שאנו מקליטים מספר רב של דגימות)











ניתוח:

לאחר הכנסת קונפיגורציה ראשונה שבה מיקום הטריגר הוא 50% וסוג הטריגר הוא נמוך, אנו רואים כי רק לאחר שאות הטריגר נמצא במצב נמוך 3 מחזורי שעון נמצאת עליית טריגר. לאחר מכן המערכת ממשיכה להקליט את 16 הדגימות הנותרות ורק אחר כך מתחילה להוציא את המידע. תוך כדי הוצאת המידע אנו משנים את אות הטריגר ורואים כי לשינוי זה אין השפעה על תפקוד המערכת, כנדרש.

לאחר שהמערכת מסיימת להוציא את כל המידע אנו שוב משנים את אות הטריגר שמקבל את כל הערכים האפשריים (עלייה, ירידה, גבוה ונמוך למשך שלושה מחזורים) ושוב רואים כי לפני שמתבצע קינפוג נוסף, המערכת אינה מוציאה שוב מידע.

רק לאחר ביצוע קינפוג נוסף (בו המיקום הוא 0% וסוג הטריגר הוא עליה) אנו רואים כי המערכת מוצאת את עליית הטריגר ולאחר שהיא מקליטה את כל המידע הנדרש, היא מוציאה את המידע הרלוונטי.

אנו רואים בסימולציה זו כי רק לאחר קינפוג של המערכת מתחיל להתבצע חיפוש של עליית טריגר בהתאם לקונפיגורציה שנבחרה. כדאי לציין כי קינפוג נוסף של המערכת יכול בהחלט להתרחש במקביל להוצאת המידע מהמערכת, אך עליית הטריגר אינה יכולה להתרחש לפני שכל המידע כולו יצא בחזרה למשתמש. (לאחר שאות read\_controller\_finish עולה בסיום הוצאת המידע, הרגיסטר בו נמצא אות ENABLE מתאפס, על מנת לזהות עליית טריגר דרושה כתיבה של ערך VALID לרגיסטר זה לאחר מכן)

טסט מספר 9

ערכי ג'נריק:

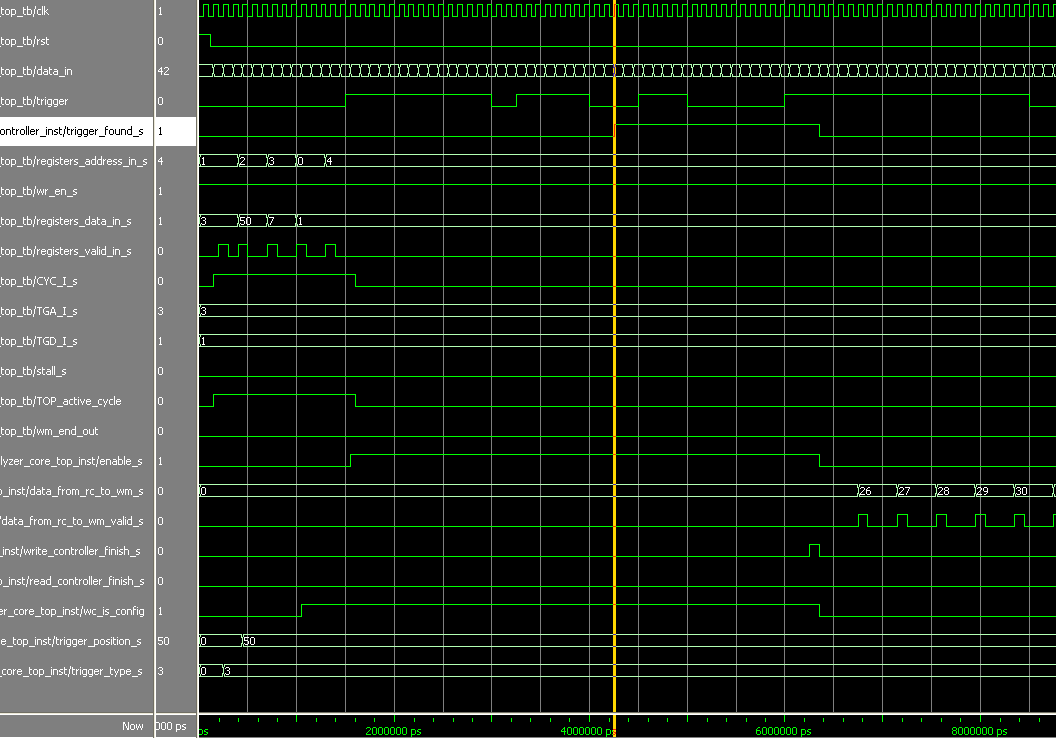
|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 5 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

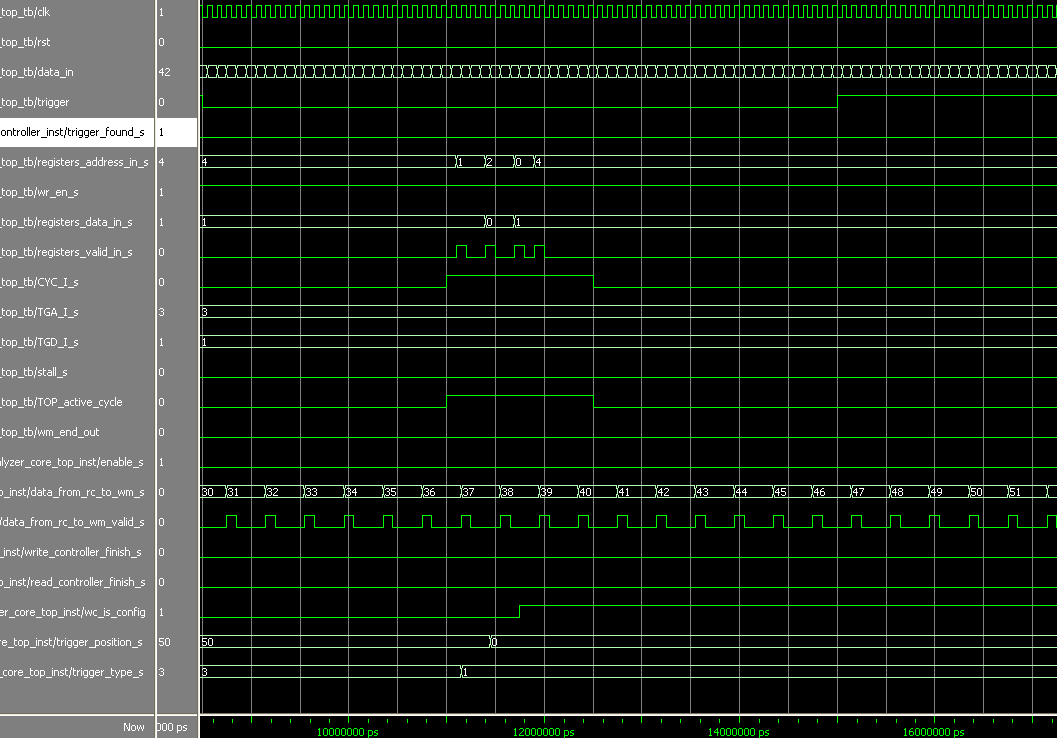
תיאור הבדיקה:

בחינה של מיקום הקונפיגורציה. בטסט זה ניתן לראות כי קינפוג המערכת יכול לבוא אפילו לפני שכל המידע הוצא בחזרה למשתמש, אך כמובן שה STREAM השני וזיהוי הטריגר יכולים להתרחש רק לאחר סיום ה STREAMהראשון. ( לאחר עליית אות READ CONTROLLER FINISH)

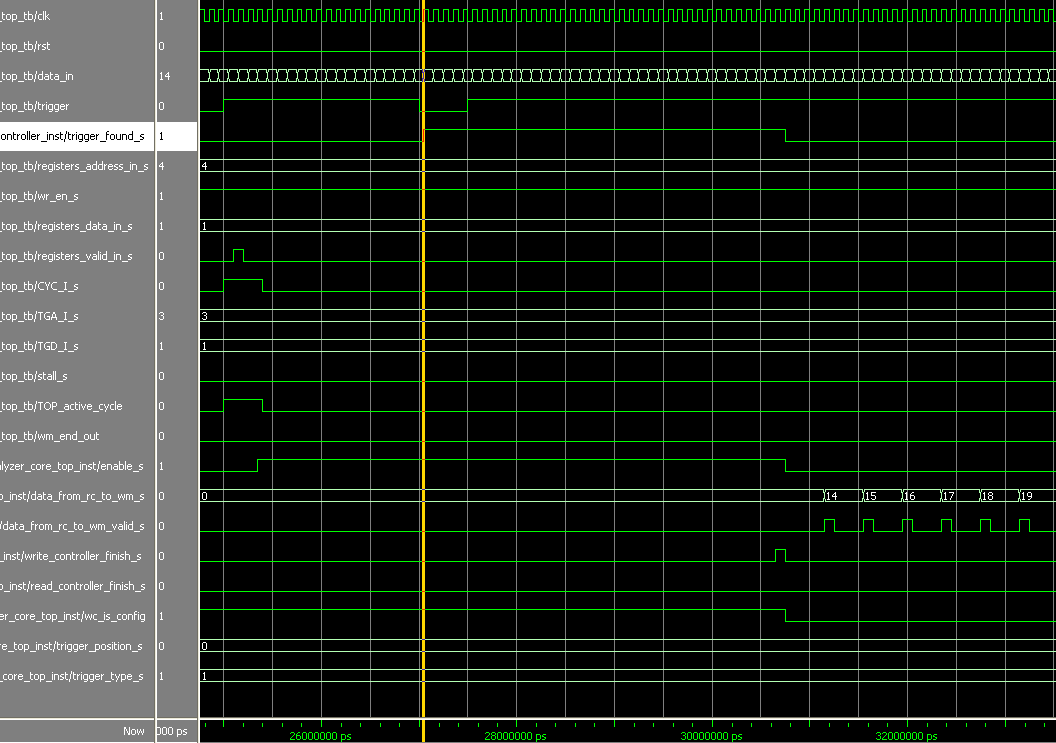
בבדיקה זו אנו מקנפגים את המערכת למצב ראשון ובזמן שהמידע עדיין יוצא בחזרה למשתמש אנו מקנפגים כבר למצב שני. לאחר שכל המידע יוצא נותר לנו רק לכתוב אות VALID לתוך הרגיסטר ENABLE והמערכת מתחילה לעבוד עם הקונפיגורציה השנייה.

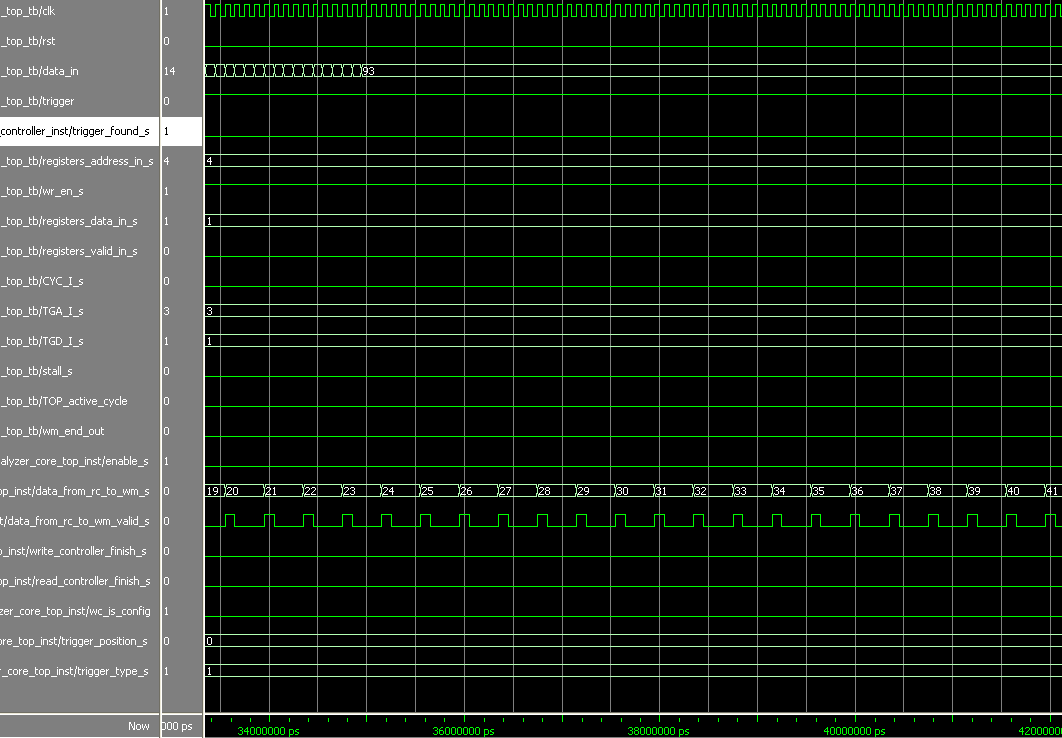
תוצאות הסימולציה:

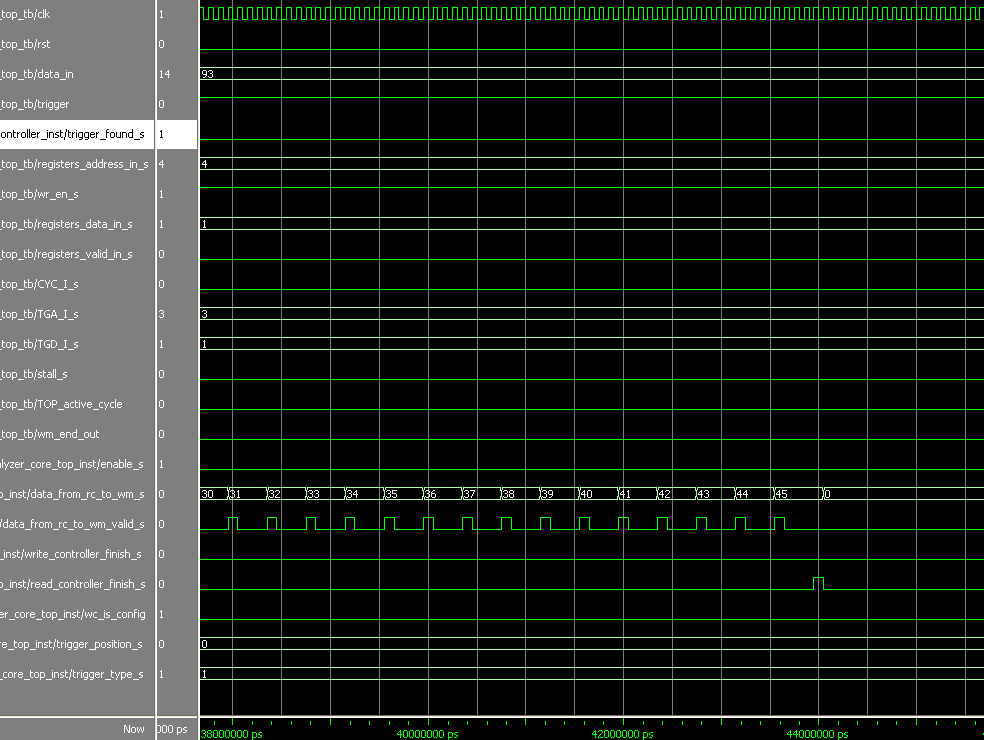












ניתוח:

סימולציה זו דומה לסימולציה הקודמת, אך השינוי הוא במיקום הקונפיגורציה השנייה. ניתן לראות כי את הקינפוג השני אנו מבצעים עוד לפני סוף הוצאת המידע, וכי המערכת פועלת בהתאם לקונפיגורציה השנייה שהוגדרה לאחר הוצאת כל המידע, וכי כדי להתחיל את ה STREAM השני אנו מתחילים ישר לאחר כתיבת אות VALID לרגיסטר ENABLE, ללא צורך בפעולות נוספות.

בשאר הסימולציה ניתן לראות כי המידע יוצא כנדרש, וכי זיהוי הטריגר מתקיים בצורה התואמת לקונפיגורציה שהוגדרה.

טסט מספר 10

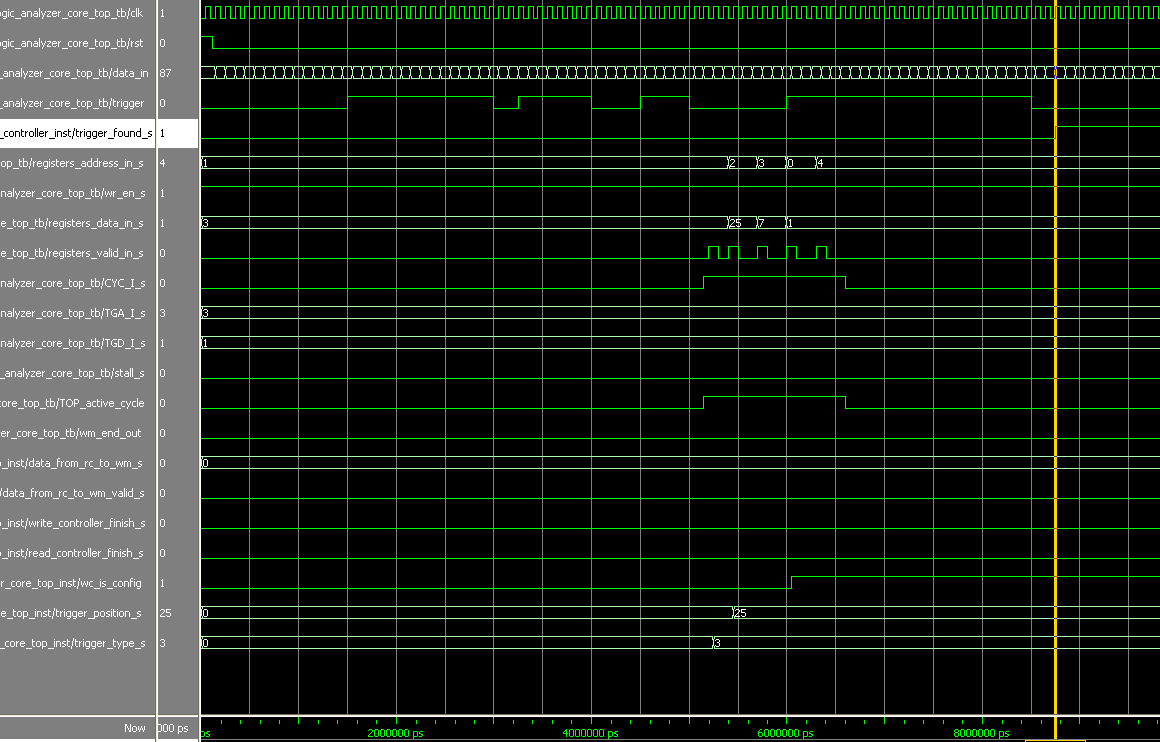
ערכי ג'נריק:

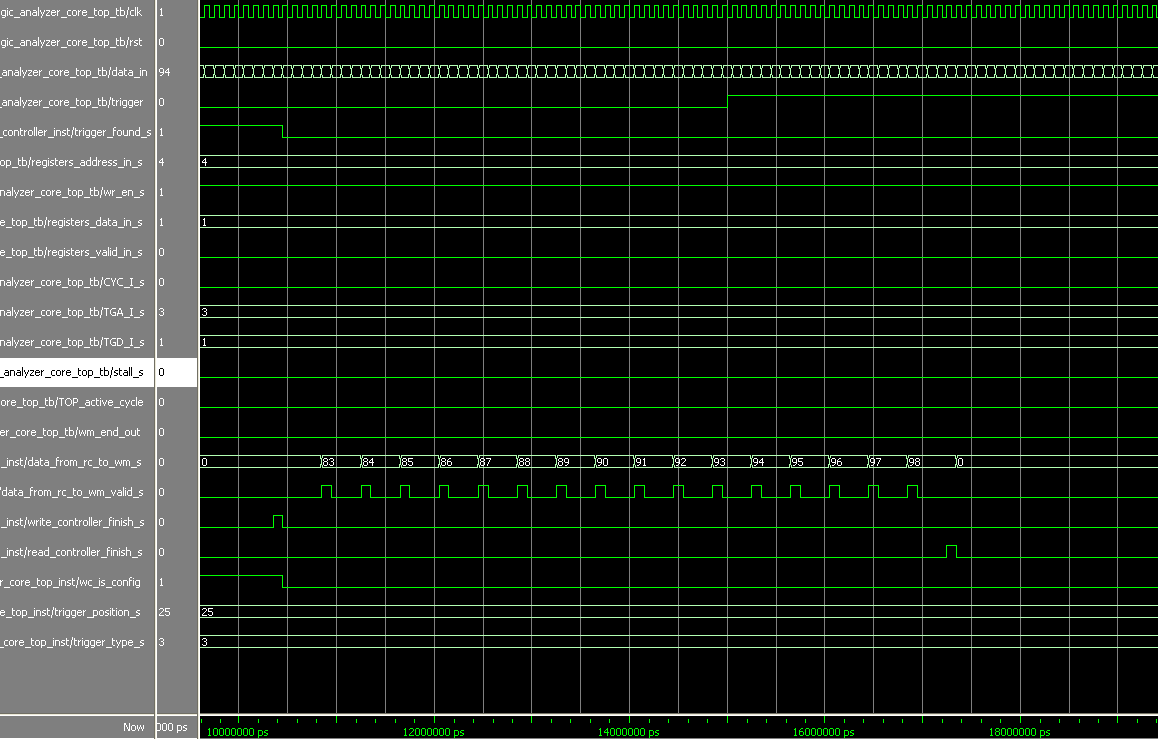
|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

תיאור הבדיקה:

בדיקה נוספת של מיקום קונפיגורציה. אנו משנים את אות הטריגר כך שיקבל את כל סוגי הטריגר האפשריים, אך מקנפגים את המערכת רק לאחר מכן ובודקים האם אות טריגר עלה כאשר המערכת עדיין לא מקונפגת.

תוצאות הסימולציה:





ניתוח:

אות הטריגר מקבל בתחילה ערכים שונים וניתן לראות כי המערכת אינה מזהה עליית טריגר. רק לאחר קינפוג המערכת (מיקום 25% סוג נמוך) וירידת הטריגר למשך שלושה מחוזרי שעון רצופים נמצאה עליית טריגר (מידע מספר 87) מיקום הטריגר הוא 25 ומספר הדגימות הנשמרות הוא 16 ולכן  המערכת מתחילה להוציא את המידע 4 דגימות לפני עליית הטריגר, כלומר מתא מספר 83 למשך 16 דגימות.

טסט מספר 11

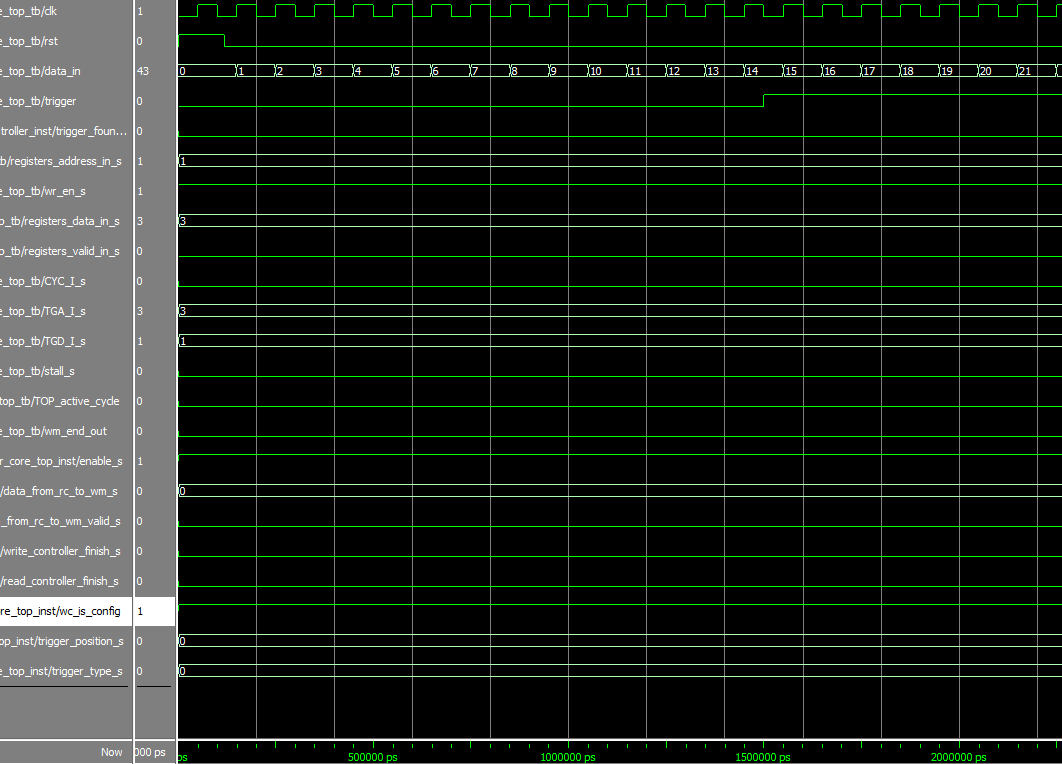
ערכי ג'נריק:

|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 0 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

תיאור הבדיקה:

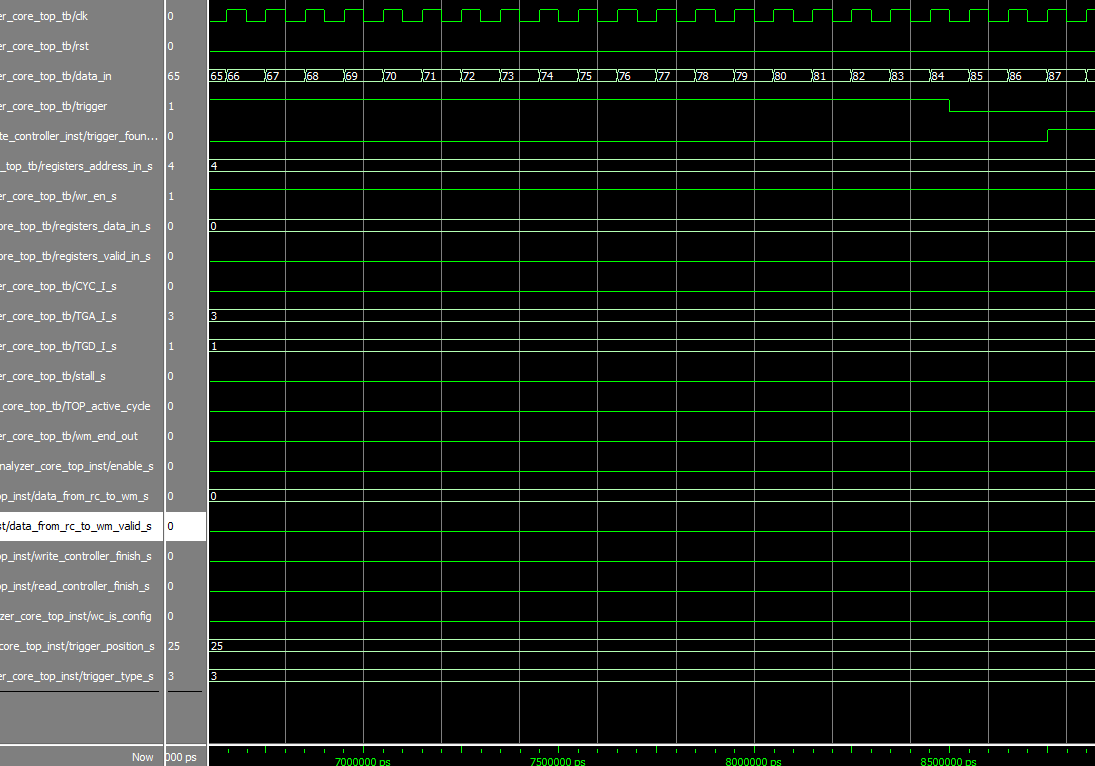
אנו בודקים את השינוי בסיגנל ENABLE שכעת פעיל בנמוך. אנו משנים בתחילה את אות הטריגר כדי לראות כי הוא אינו עולה לפני קינפוג המערכת, וכאמור לאחר כתיבה של ערך מתאים לרגיסטר ENABLE (כעת ערך זה יהיה 0 במקום 1 לתחילת עבודה) המערכת תתחיל בעצם לשמור את המידע ותחפש עליית טריגר.

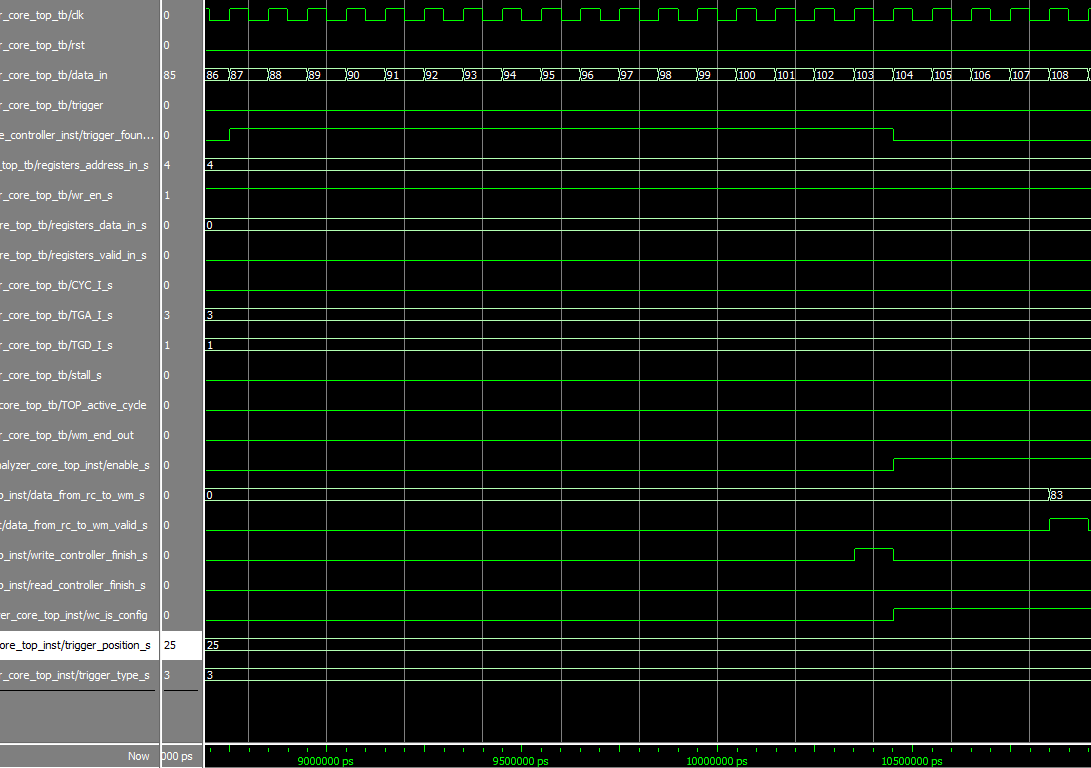
תוצאות הסימולציה:

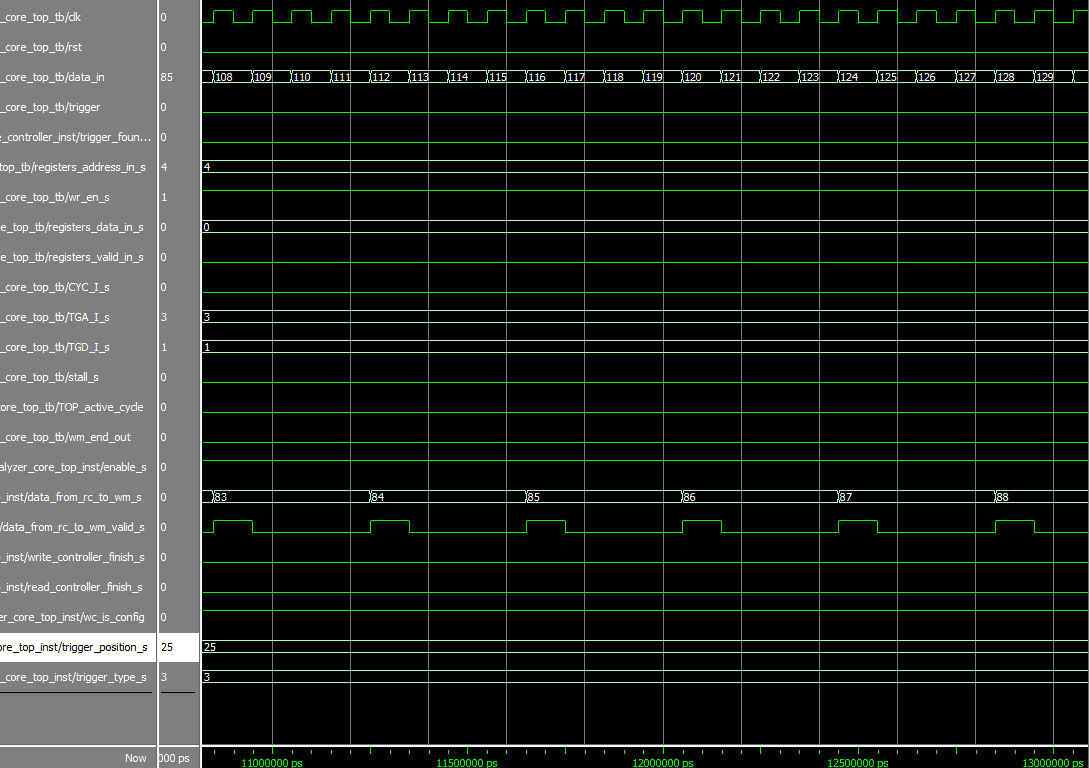


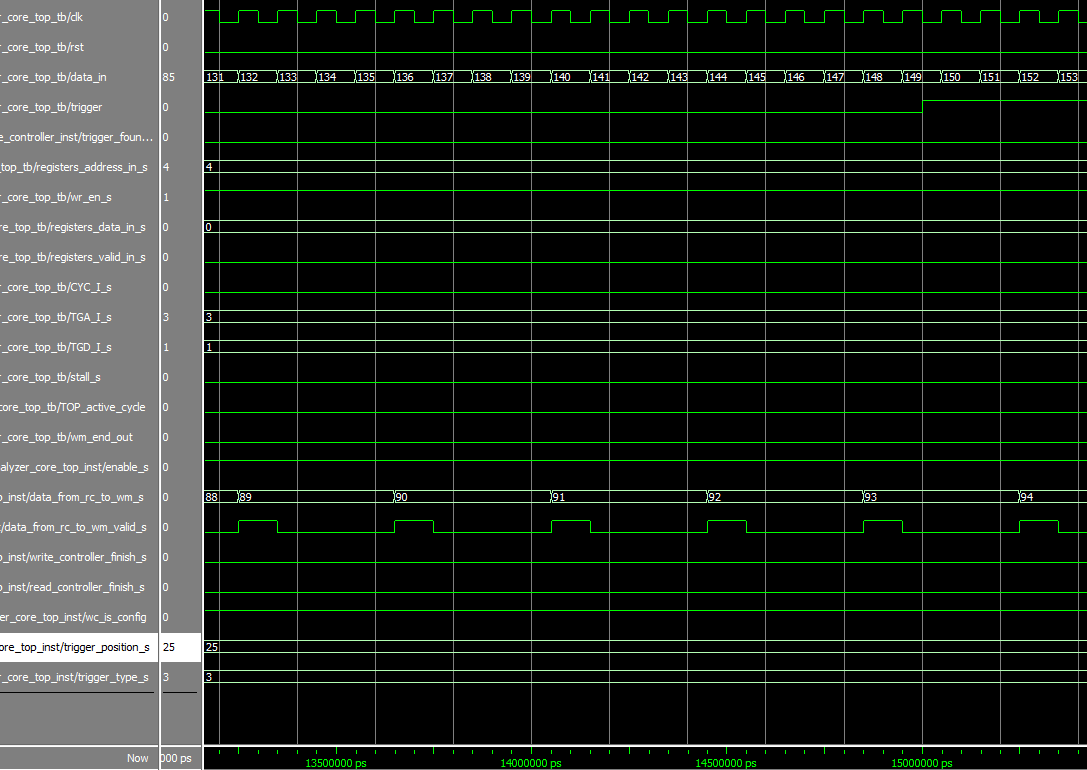


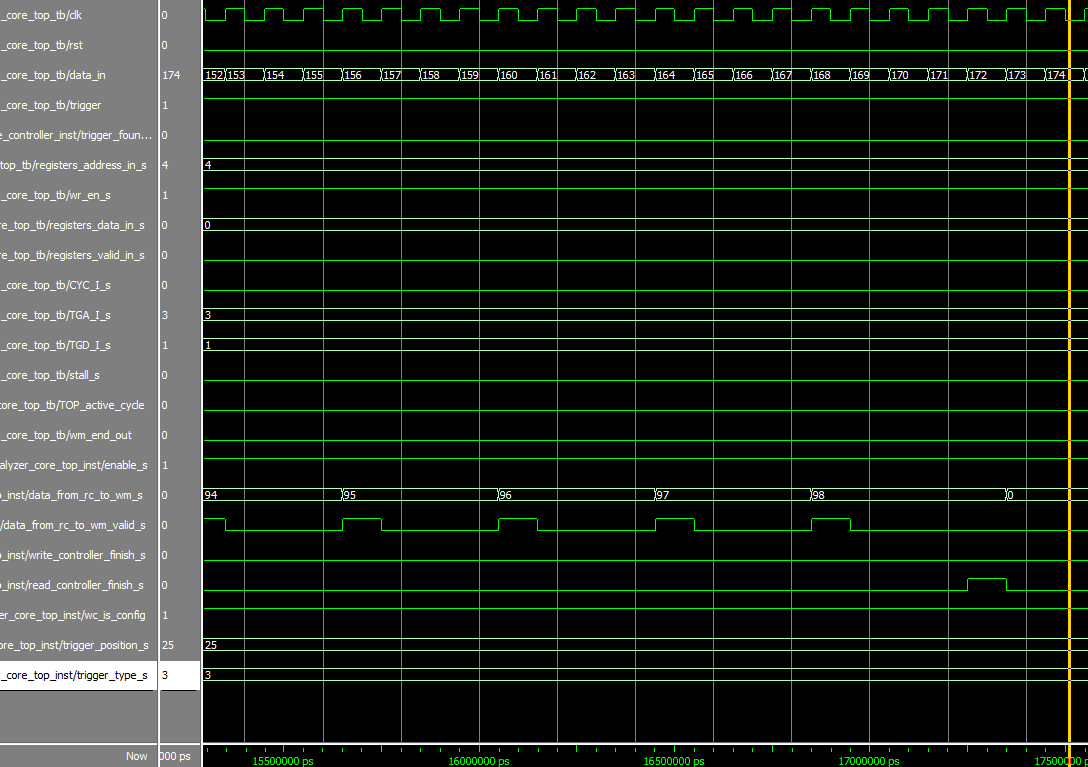












ניתוח:

המערכת כעת במצב שבו ENABLE פעיל בנמוך, כתוצאה מכך ניתן לראות כי שני האותות enable\_s, wc\_is\_config מקבלים בתחילה ערך גבוה, ולאחר ביצוע קינפוג של המערכת (תמונה שלישית) שבה נכתב ערך נמוך לרגיסטרים, אותות אלו משנים את ערכם מגבוה (לא פעיל) לנמוך (פעיל) ומרגע זה המערכת מתחילה להקליט את המידע ולחפש עליית טריגר.

לאחר שנמצאת עליית טריגר (מוגדר כנמוך למשך שלושה מחזורים), אות trigger\_found עולה, המערכת ממשיכה להקליט את המידע הנחוץ (מיקום הטריגר הוא 25% ולכן אנו נדרשים להמשיך ולהקליט עוד 75% מהמידע) ולבסוף יוצא המידע הרלוונטי. מיקום הטריגר הוא כאמור 25 ולכן רבע מהמידע המוקלט (4 דגימות) יוקלטו לפני הטריגר, טריגר עולה במידע 87 ולכן המידע מתחיל לצאת מתא מספר 83.

בנוסף ניתן לראות כי בסיום ההקלטה ערכי enable\_s, wc\_is\_config חוזרים להיות גבוהים (לא פעילים) והמערכת מחכה לקינפוג הבא.

טסט מספר 12

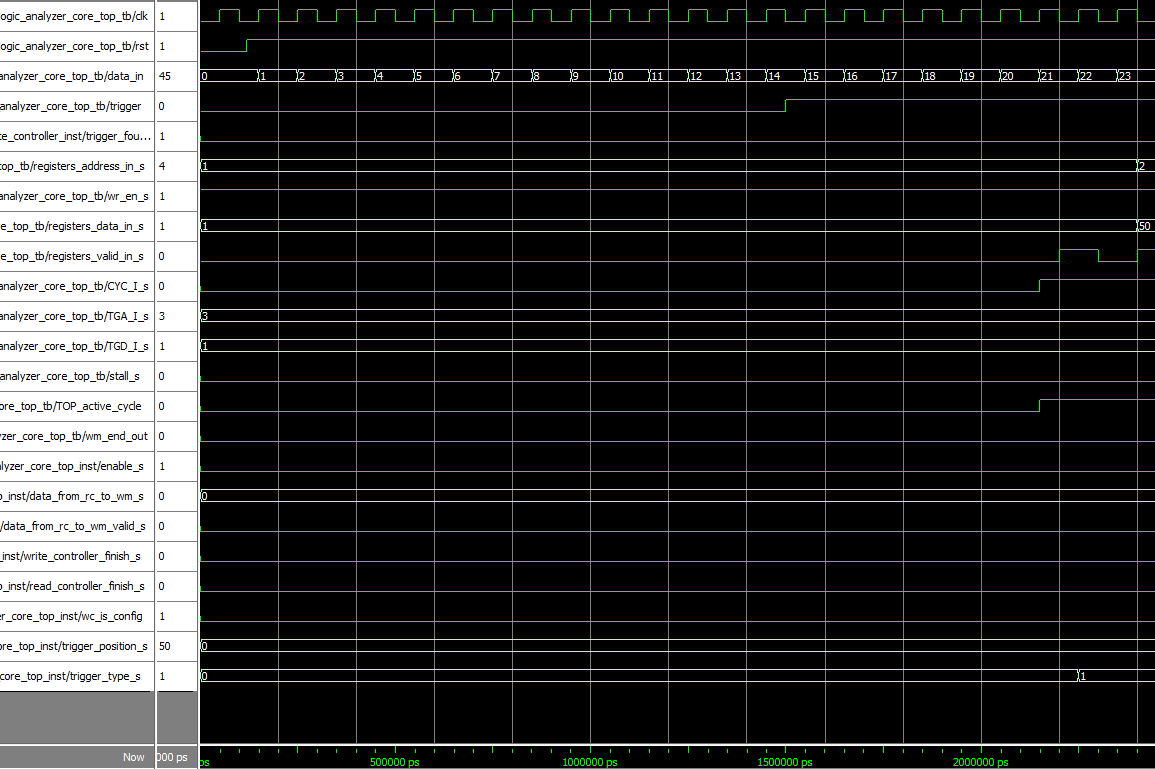
ערכי ג'נריק:

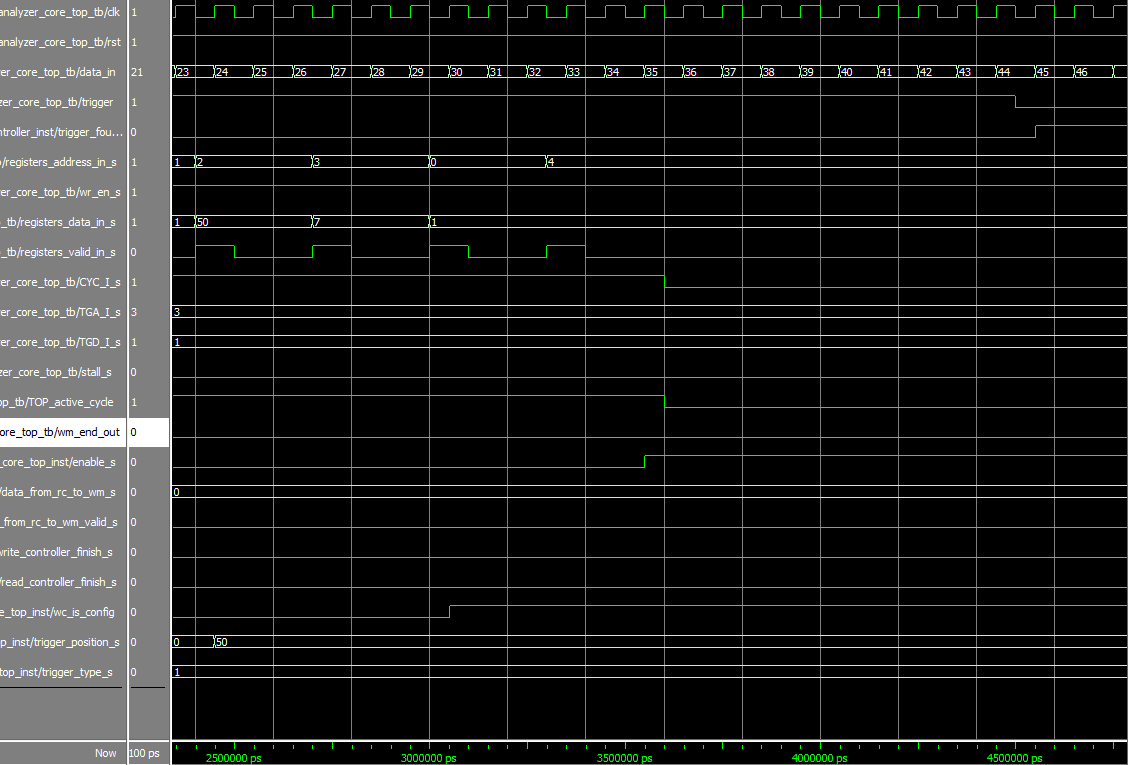
|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 0 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 3 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Word width in WB protocol | 8 | Add\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

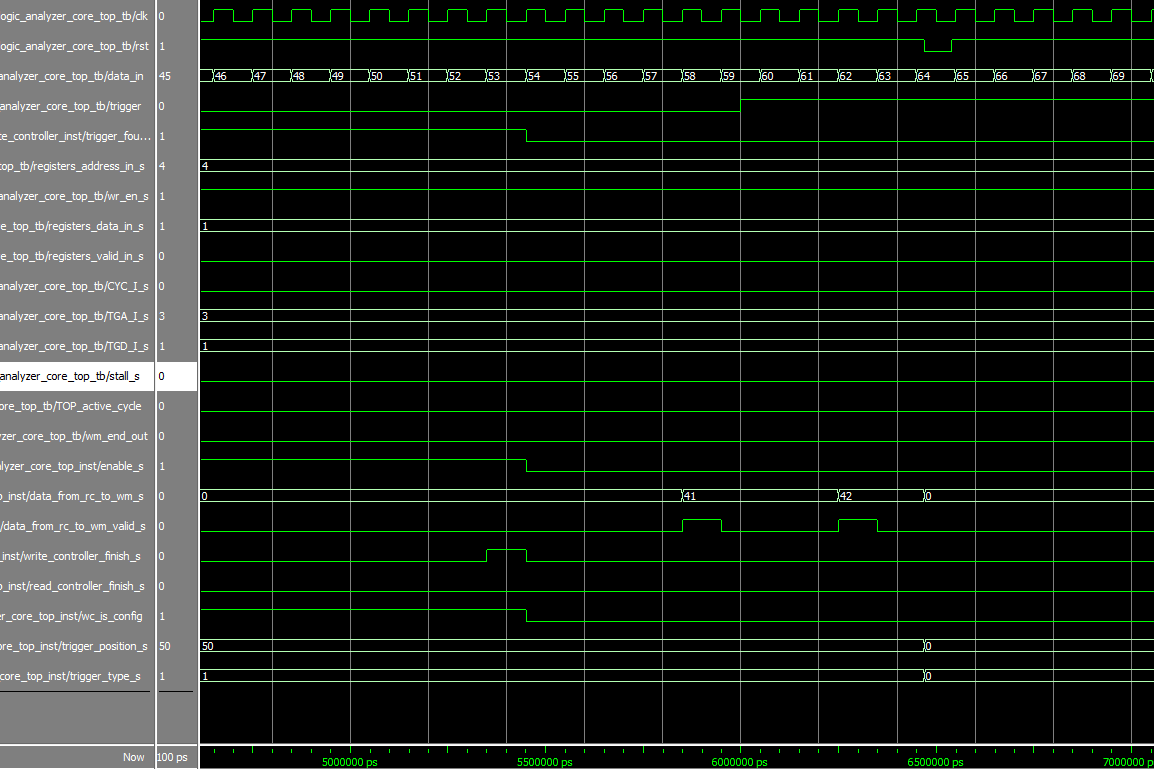
תיאור הבדיקה:

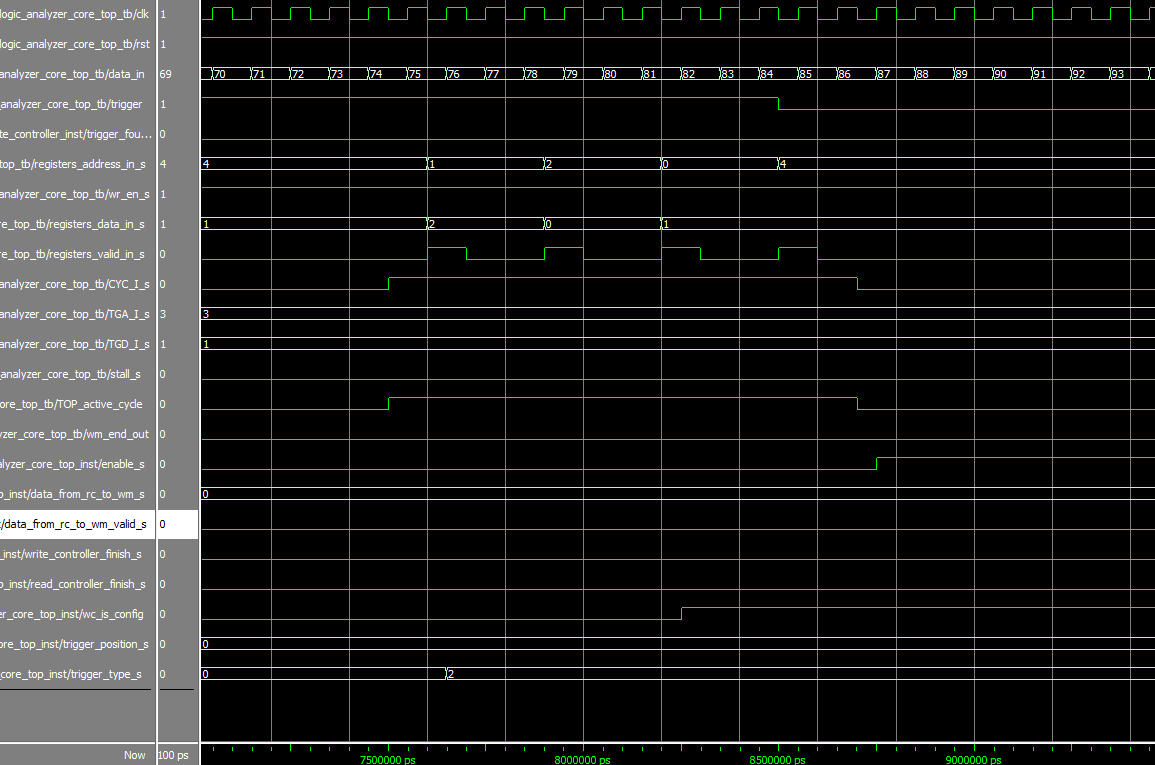
אנו בודקים את השינוי בסיגנל RESET שכעת פעיל בנמוך. אנו בתחילה מקנפגים את המערכת, מעלים את אות הטריגר ולאחר שמספר דגימות יוצאות אנו מורידים אות RESET ומקנפגים את המערכת מחדש לערכים שונים. אנו באים לבדוק את תגובת המערכת לשינוי אות RESET (פעיל בנמוך במקום בגבוה), כיצד המערכת מתאוששת לאחר הורדת אות RESET ובודקים את המשך העבודה עם קונפיגורציה שונה של ערכים.

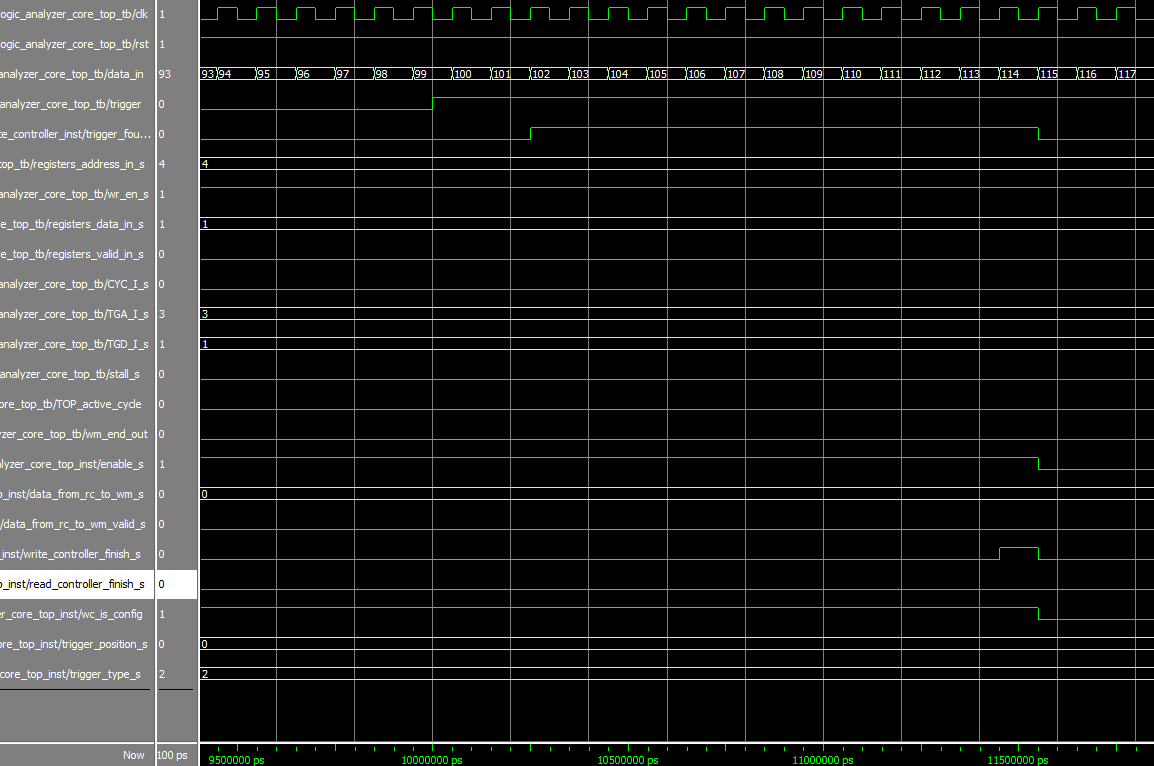
תוצאות הסימולציה:

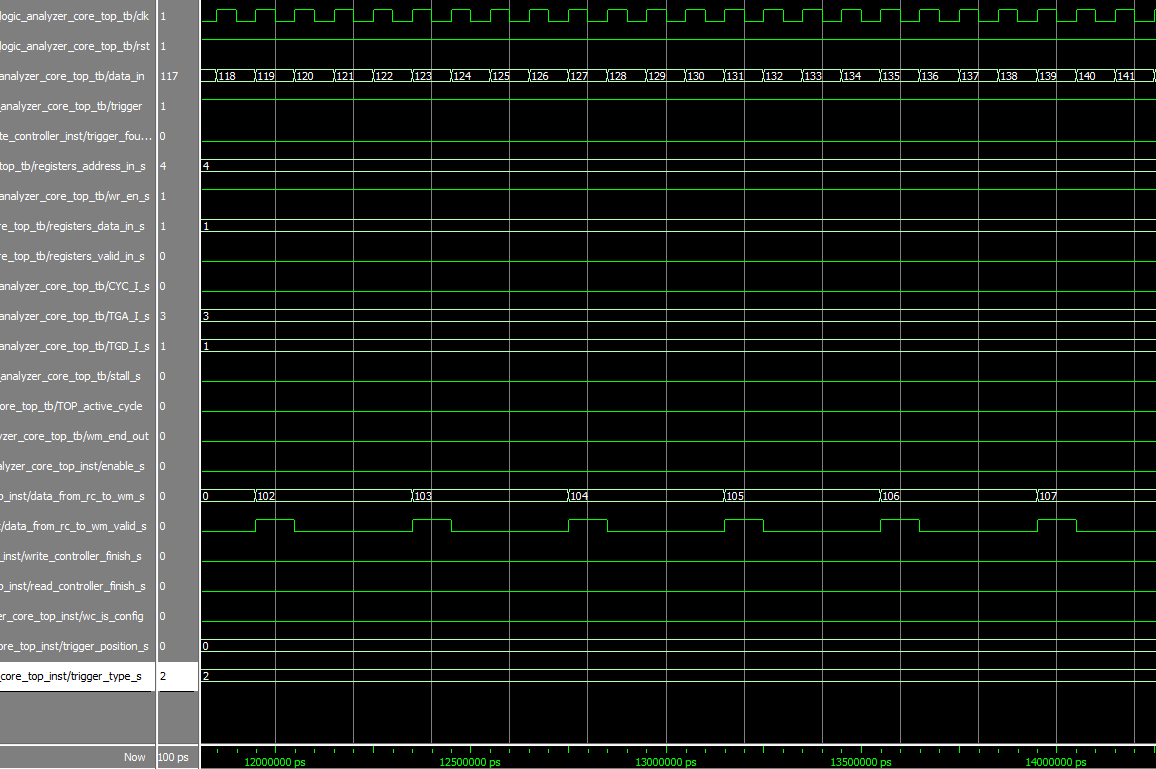


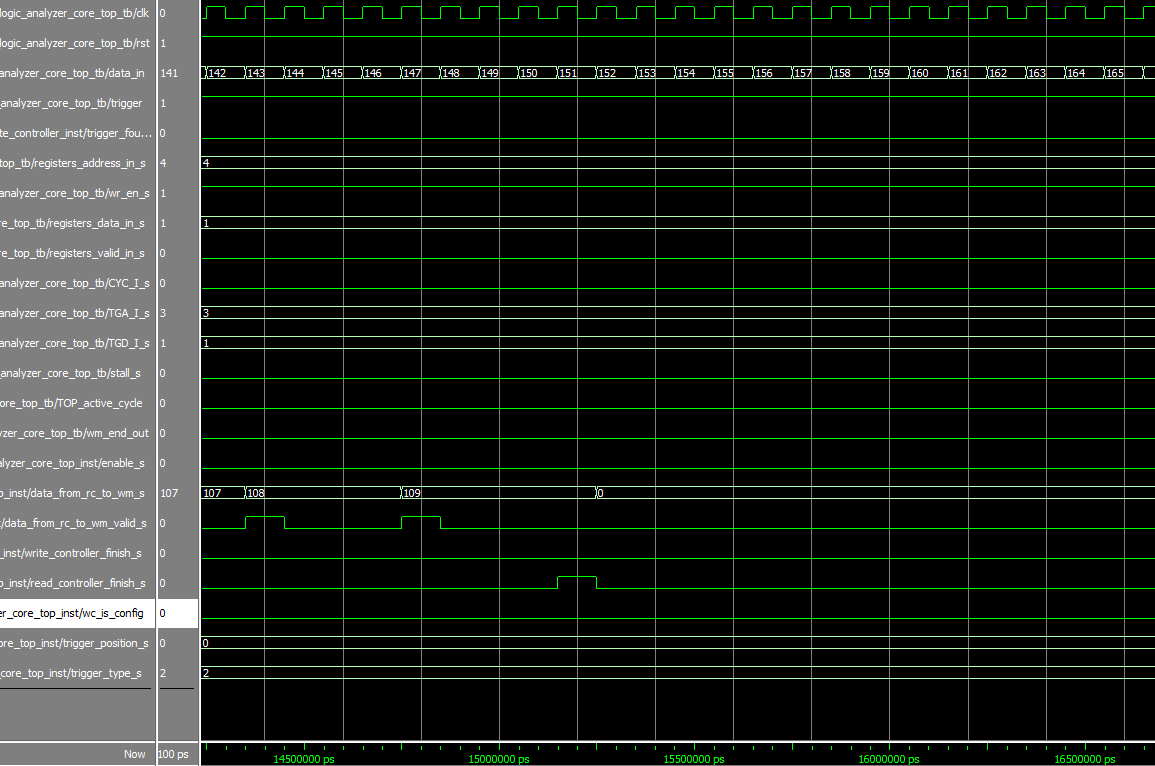












ניתוח:

אנו מקנפגים בתחילה את המערכת לערכים הבאים: מיקום טריגר-50%, סוג טריגר- עלייה. כאשר אנו מקליטים 8 דגימות (record\_depth\_g = 3).

לאחר הקינפוג המערכת מזהה עליית טריגר בתא מספר 45, כאשר המיקום הוא 50% ולכן אנו מתחילים להוציא את המידע המוקלט 4 דגימות לפני עליה זו- כלומר מתא 41. לאחר הוצאת שתי דגימות אנו מורידים את אות RESET )ניתן כאמור לראות כי אות זה פעיל בנמוך כעת) ולאחר מכן מקנפגים מחדש את המערכת כאשר סוג הטריגר הוא גבוהה והמיקום הוא 0, כלומר מרגע מציאת טריגר המערכת מוציאה 8 דגימות.

ניתן לראות כי המערכת אכן מוצאת עליה זו בתא מספר 102 וכאמור המידע היוצא מתחיל בתא זה למשך 8 דגימות.

טסט מספר 13

ערכי ג'נריק:

|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 7 | data\_width\_g |
| ~~Word width in WB protocol~~ | ~~8~~ | ~~Add\_width\_g~~ |
| Number of signals we record in each iteration | 6 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

תיאור הבדיקה:

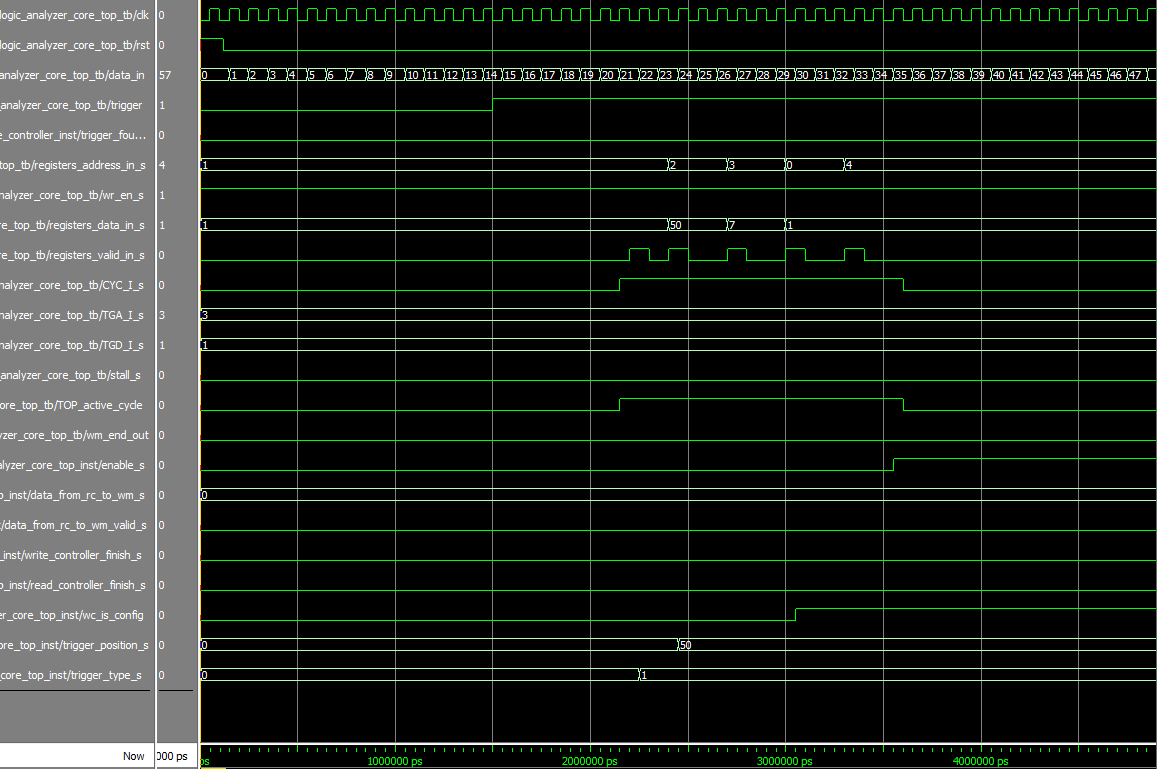
מסימולציה זו ואילך אנו מורידים את הג'נריק של רוחב הכתובת Add\_width\_g, ובמקומו מאילך אנו מציבים את הג'נריק record\_depth\_g.

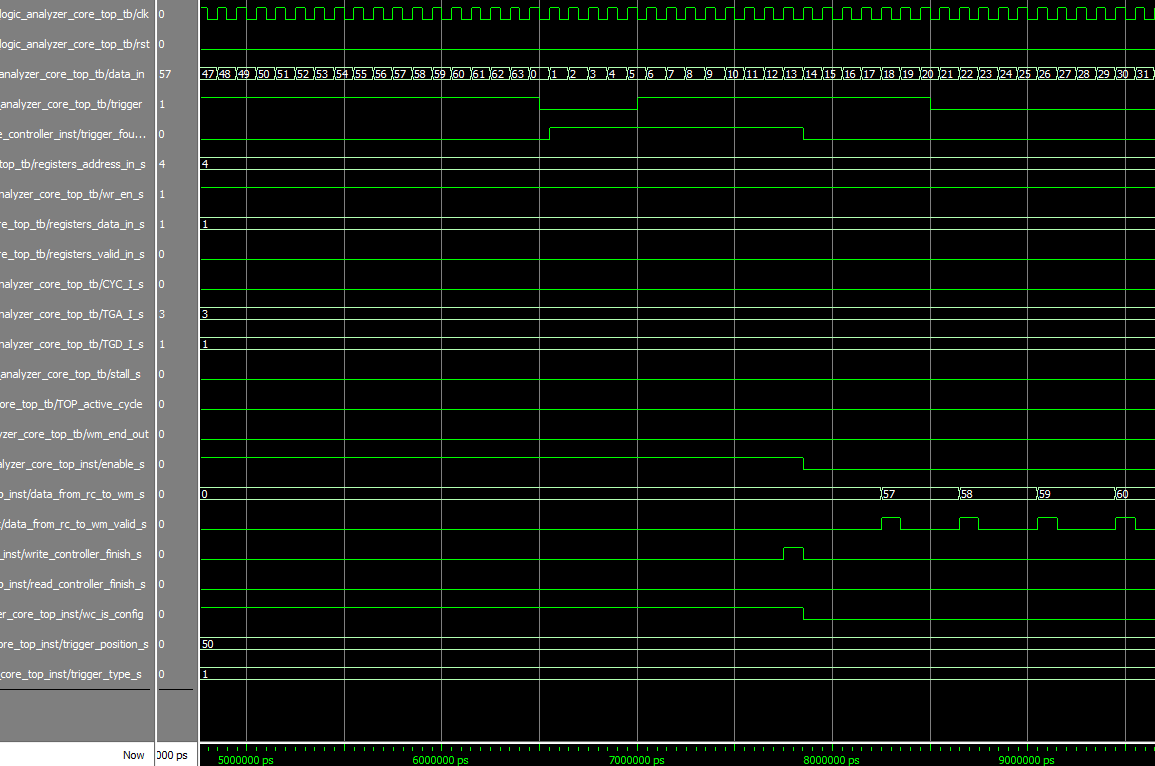
אנו בודקים בסימולציה זו את המצב בו רוחב RAM בודד רחב יותר מרוחב הסיגנלים הנשמרים, כלומר מצב בו signal\_ram\_width\_g> num\_of\_signals\_g.

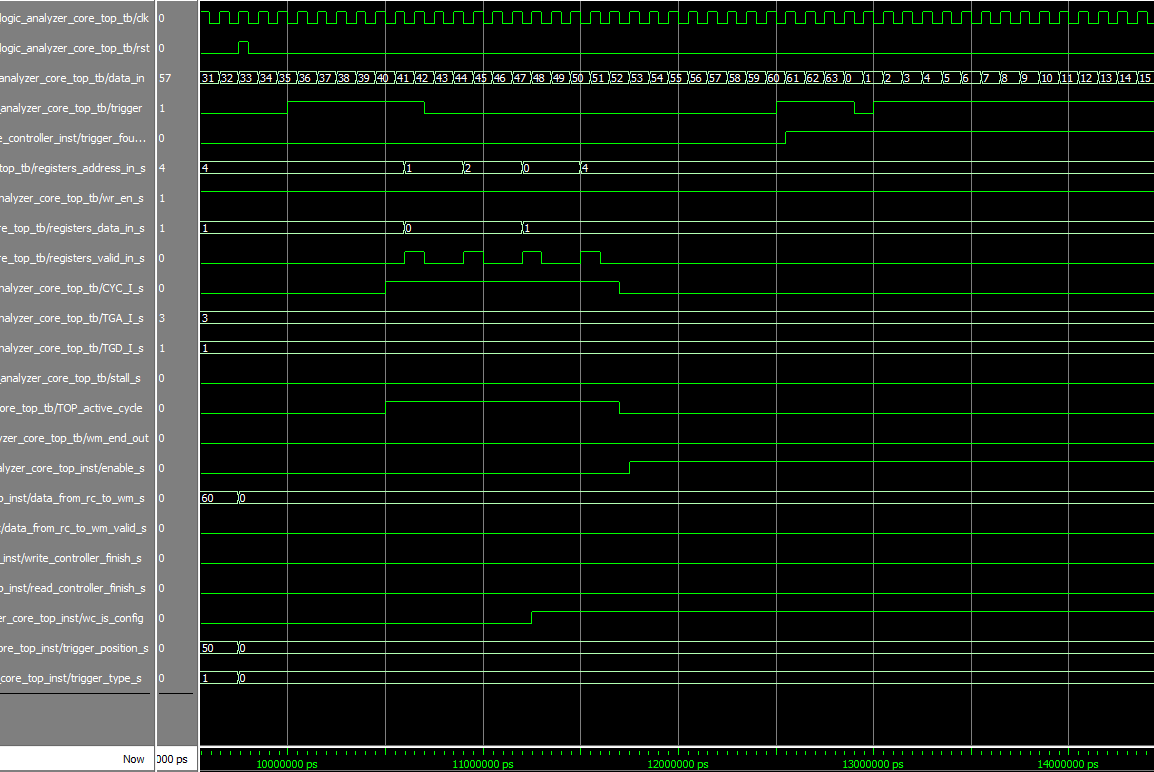
מצב זה אינו בעייתי (כמו למשל המקרה בו עומק הRAM גדול יותר מעומק המידע הנשמר, מצב שהוא בעייתי כיוון שגודל הRAM משמש אותנו לקביעת כתובת תחילת המידע).

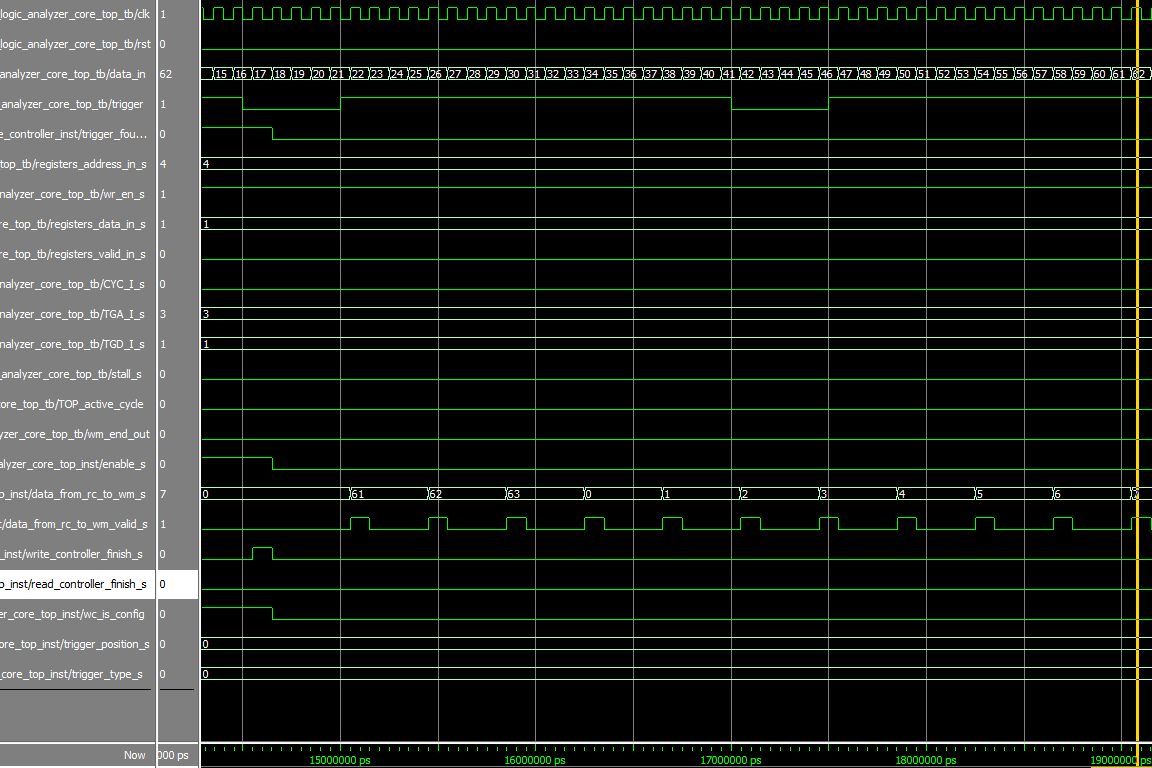
אנו מבצעים שני קינפוגים שונים, כאשר לאחר שאנו רואים שהמידע יוצא כמצופה מהתצורה הראשונה, אנו מעלים את אות ה RESET ולאחר מכן מבצעים את הקונפיגורציה השניה.

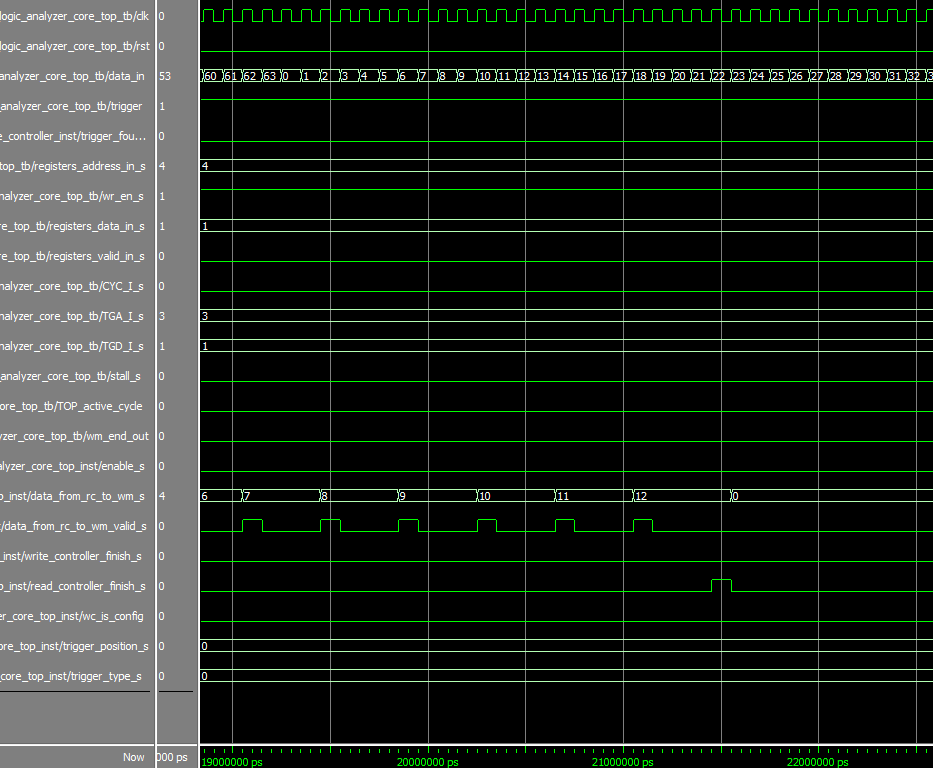
תוצאות הסימולציה:











ניתוח:

כפי שתיארנו בהתחלה, מבוצעת בתחילה קונפיגורציה בה מיקום הטריגר הוא 50, וסוג הטריגר הוא ירידה. ירידת הטריגר מתרחשת בתא מספר 1 ולכן במרחק 8 דגימות לפניו אנו מוצאים את תחילת המידע, בתא מספר 57.

לאחר שאות הRESET עולה אנו מבצעים קונפיגורציה נוספת שבה סוג הטריגר הוא עלייה ומיקומו הוא 0, ולכן כשהטריגר אכן עולה בתא מספר 61 אנו יכולים לראות כי המידע היוצא הוא מתא זה ולמשך 16 דגימות אחריו עד לסיום הוצאת המידע והעלאת אות rc\_is\_finish.

טסט מספר 14

ערכי ג'נריק:

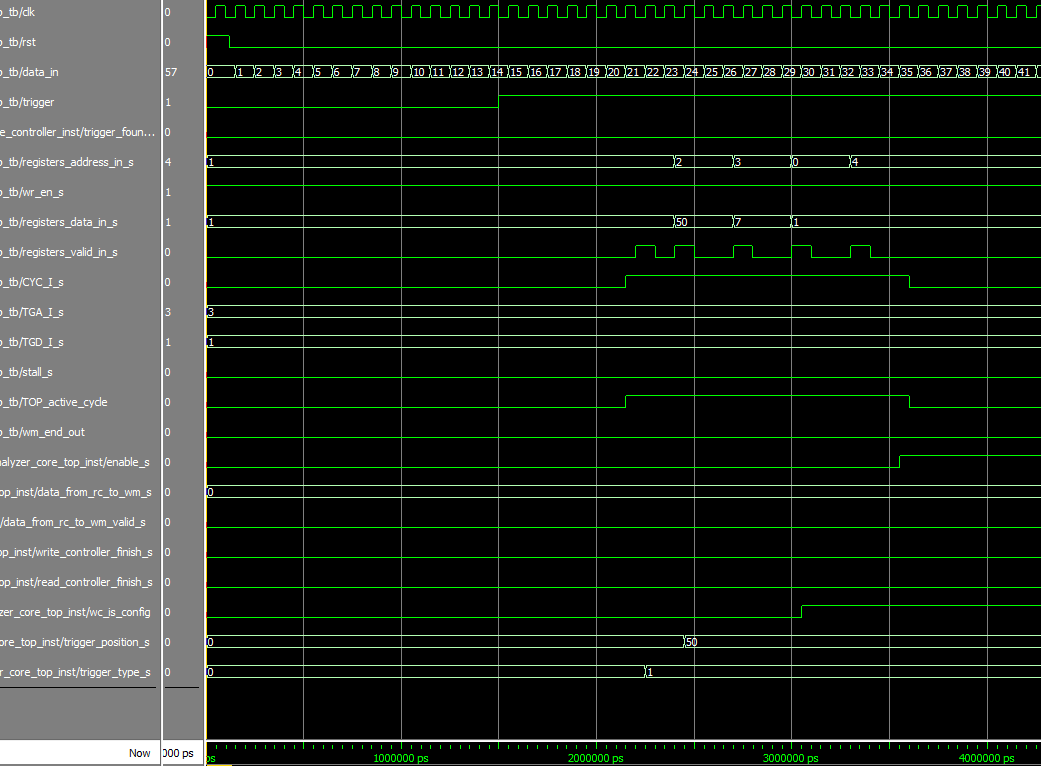
|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 6 | data\_width\_g |
| Number of signals we record in each iteration | 6 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

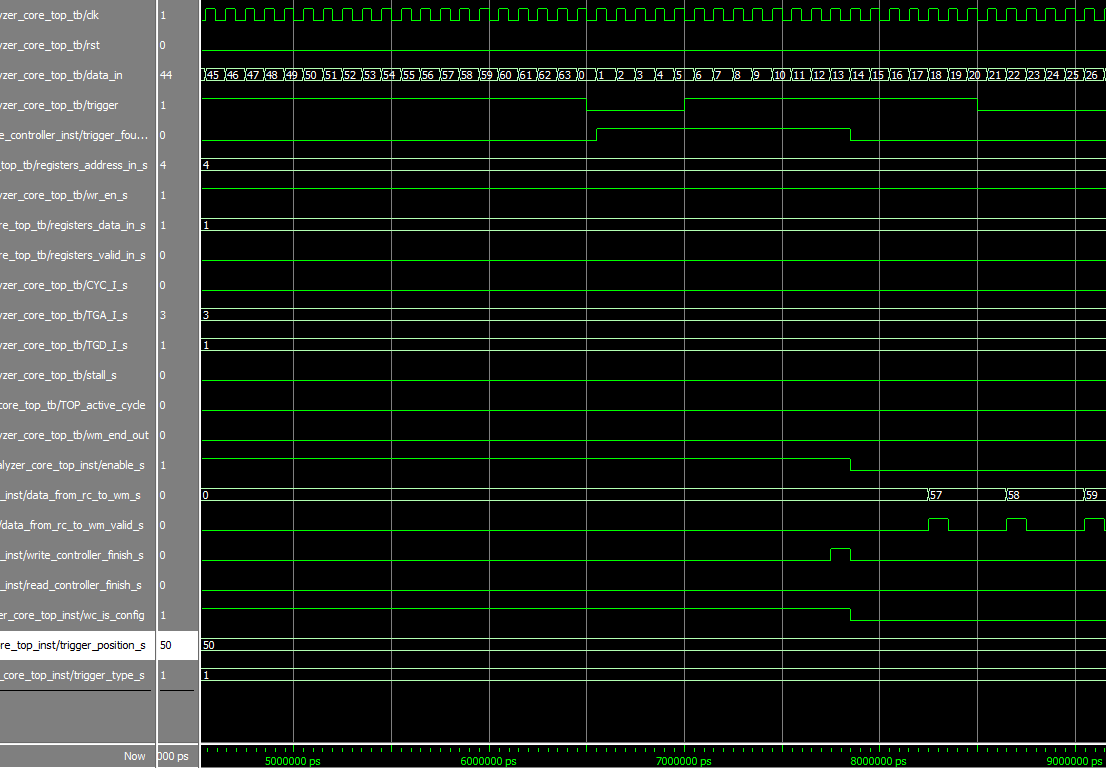
תיאור הבדיקה:

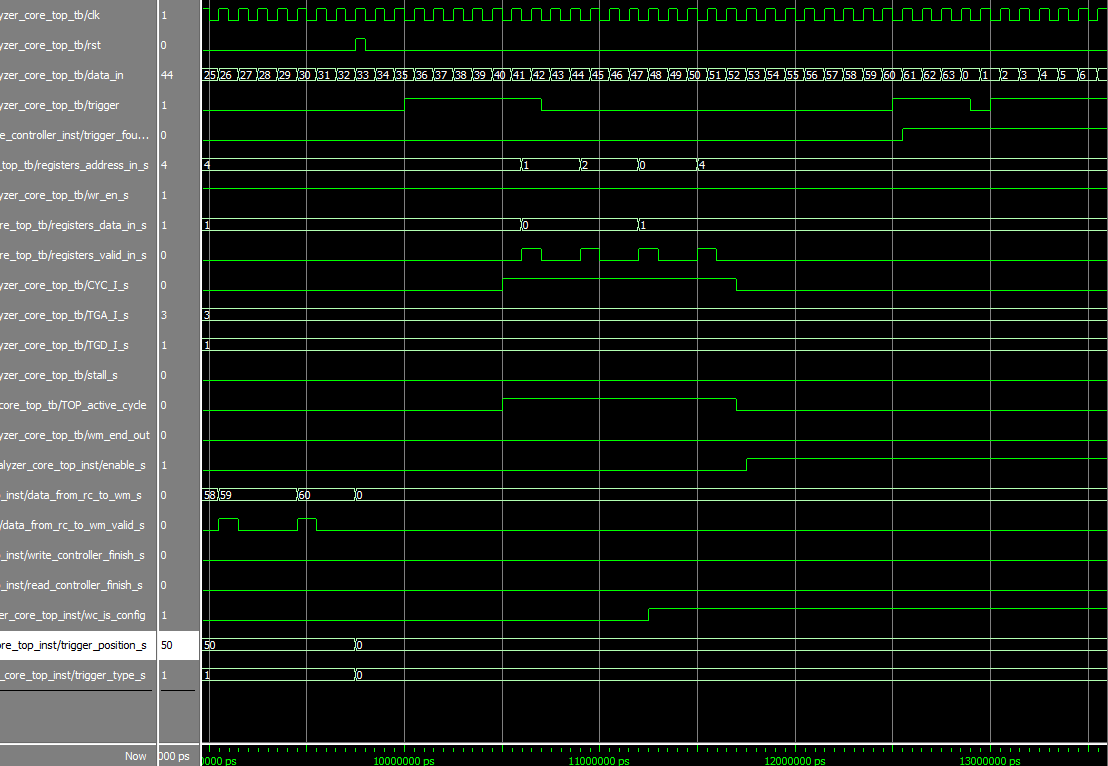
בסימולציה זו אנו משווים את מספר הסיגנלים הנכנסים לרוחב המידע של ה BUS, כלומר num\_of\_signals\_g = data\_width\_g.

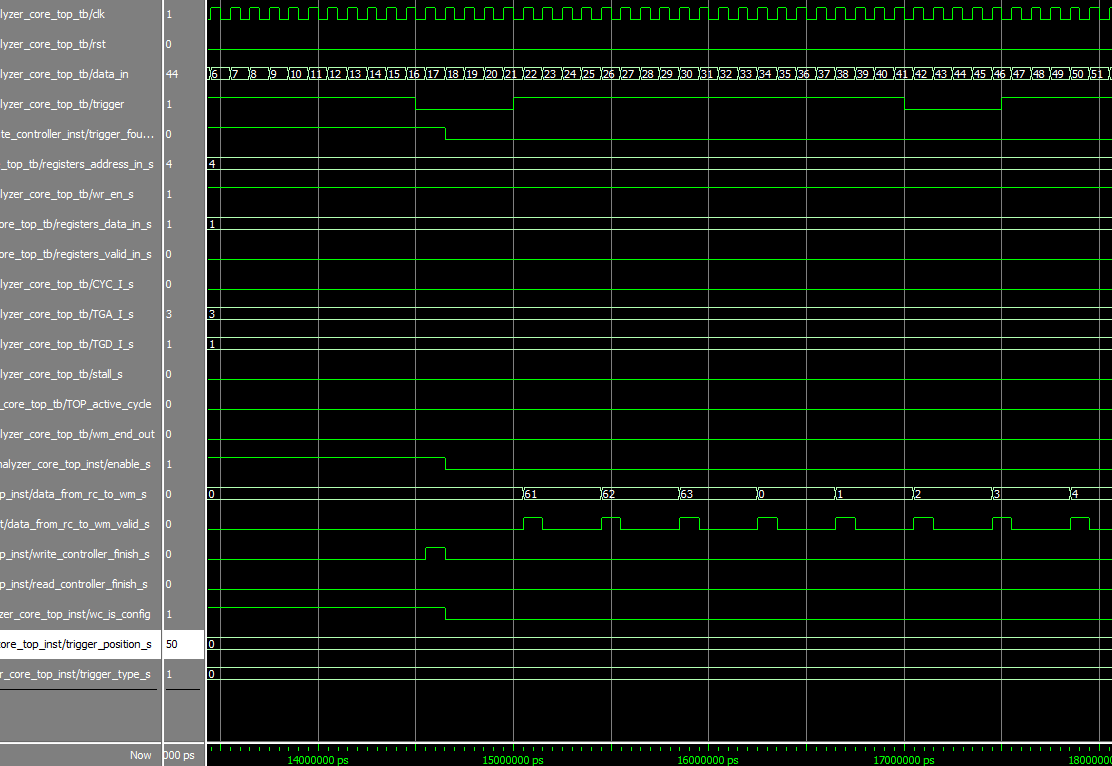
שאר חלקי הבדיקה דומים לטסט הקודם.

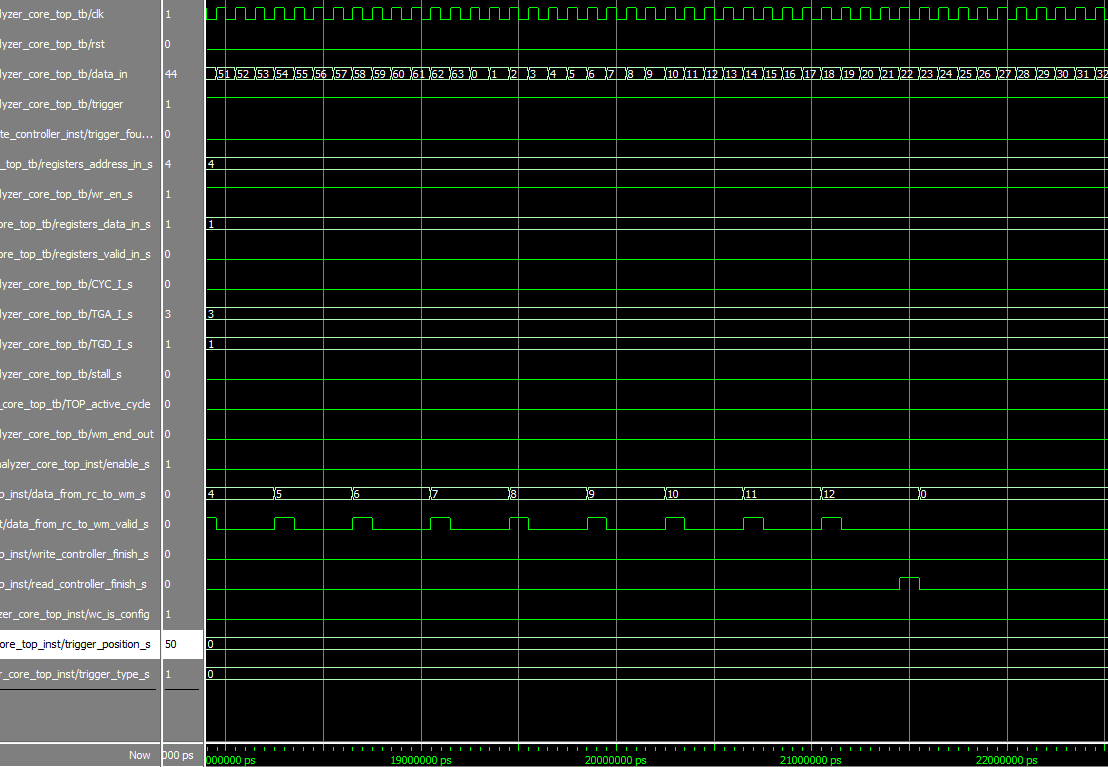
תוצאות הסימולציה:











ניתוח: תוצאות הסימולציה זהות לתוצאות הסימולציה הקודמת, ולכן לא נחזור עליהן שוב. ניתן לראות כי כאשר רוחב הסיגנלים המוקלטים הוא כרוחב ה BUS אין לנו בעיה כלל.

טסט מספר 15

ערכי ג'נריק:

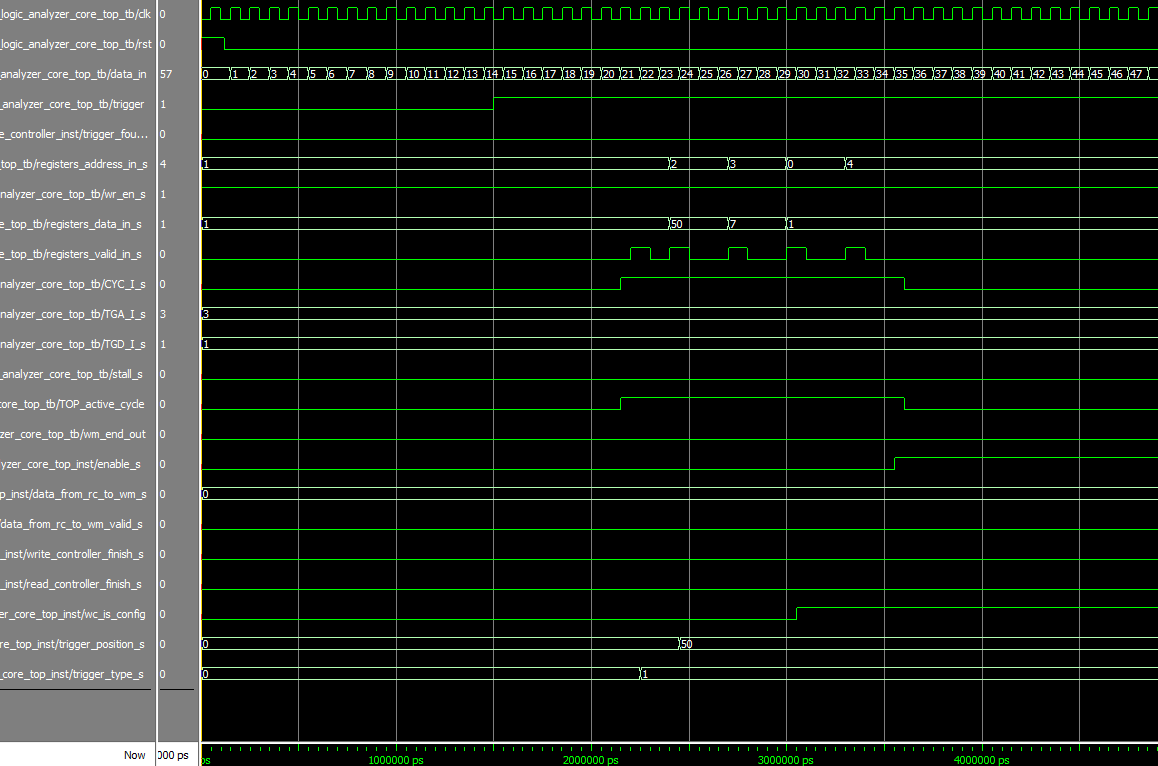
|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 8 | data\_width\_g |
| Number of signals we record in each iteration | 6 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

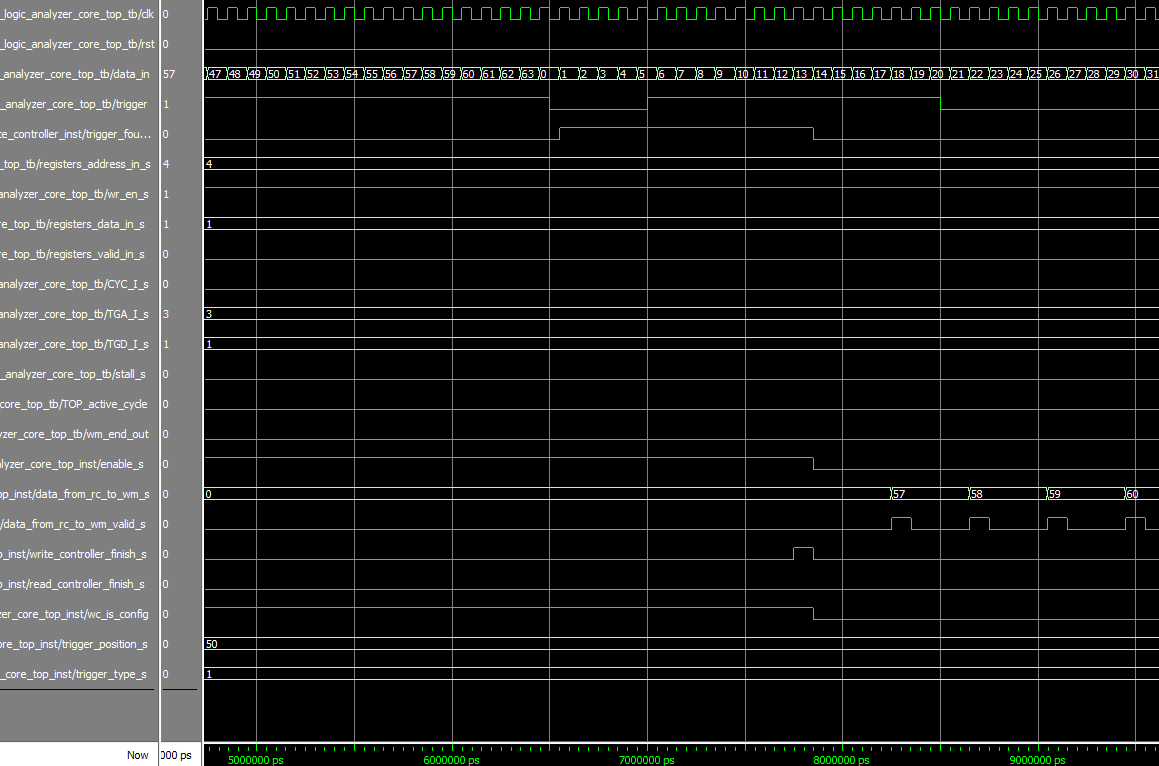
תיאור הבדיקה:

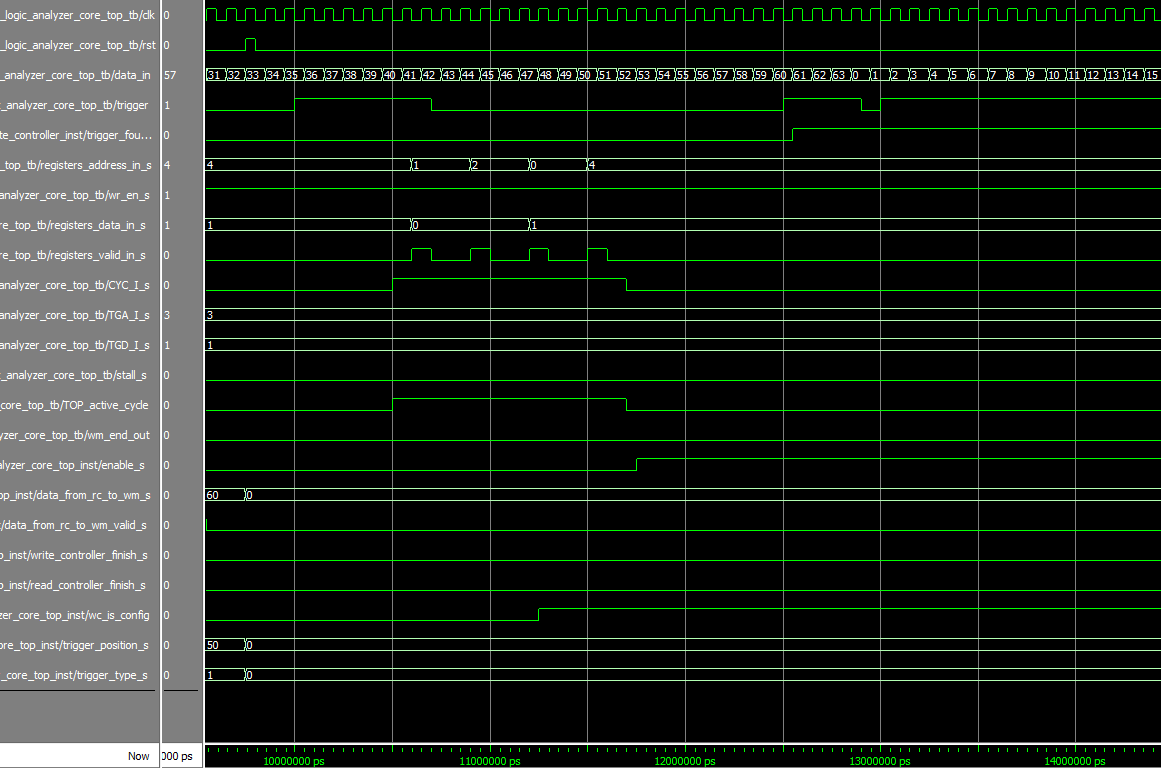
בסימולציה זו רוחב ה BUS גדול ממספר הסיגנלים הנדגמים, כלומר num\_of\_signals\_g < data\_width\_g.

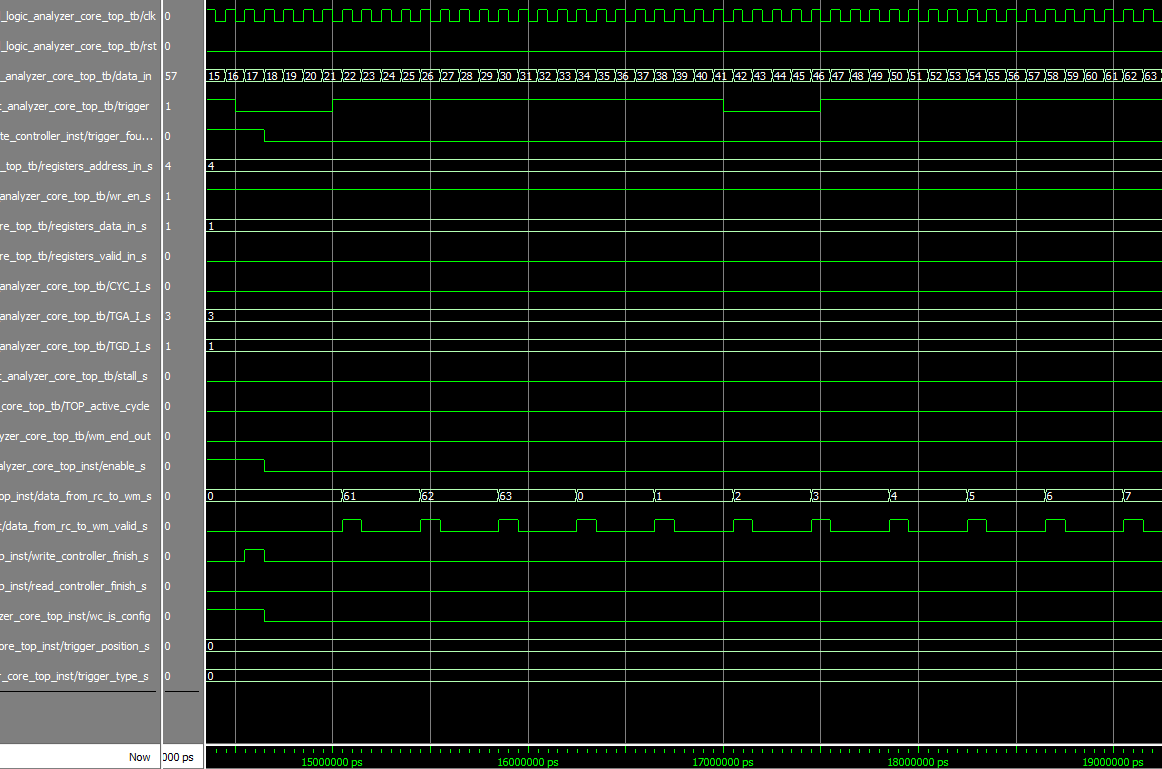
שאר חלקי הבדיקה דומים לטסט הקודם.

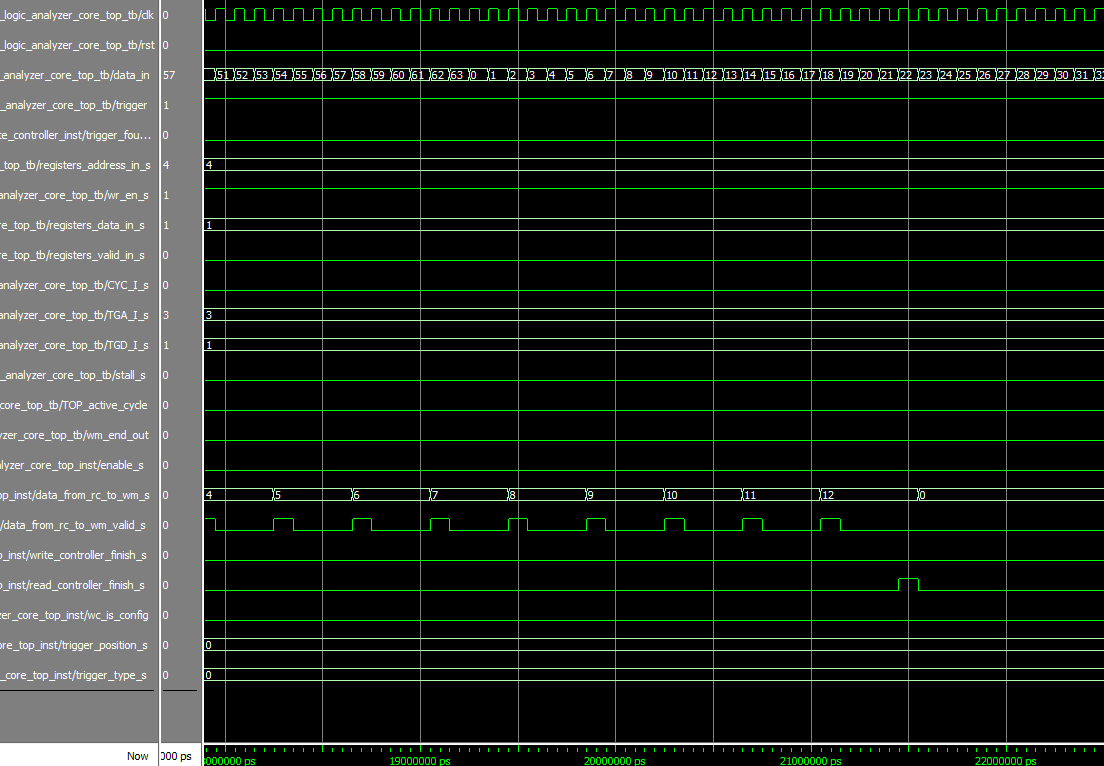
תוצאות הסימולציה:











ניתוח:

כפי שציפינו קיבלנו את אותה הדיאגרמה בדיוק, (ולכן גם הרשנו לעצמנו להקטין את הרזולוציה שלה). לכל דגימה פשוט התווספו שני אפסים בתחילתה, כך שערכה נשמר אך רוחבה כעת זהה לרוחב ה BUS.

טסט מספר 16

ערכי ג'נריק:

|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 3 | data\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

תיאור הבדיקה:

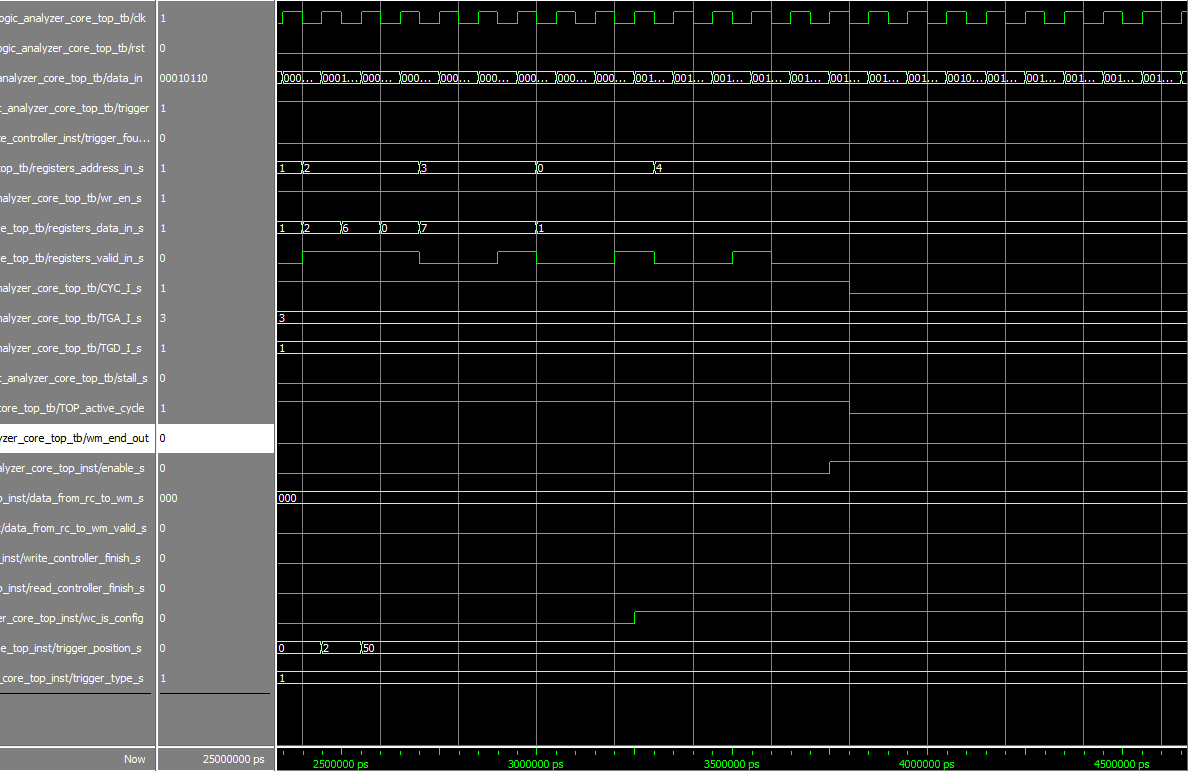
בסימולציה זו רוחב ה BUS קטן ממספר הסיגנלים הנדגמים, כלומר num\_of\_signals\_g > data\_width\_g. כתוצאה מכך מתבצעים לנו מספר שינויים: הראשון הוא שהמידע הנכנס לרגיסטרים ומגדיר את הקונפיגורציה (מיקום וסוג הטריגר) אינו יכול להיכנס במחזור שעון אחד, לכן אנו מכניסים את המידע במספר מחזורי שעון (אצלנו נדרשים 3 כיוון שרוחב הרגיסטרים הוא 7- על מנת להכיל מספר בתחום 0-100 בשפה בינארית דרושים לנו 7 ביטים), כלומר המידע שנכנס לכל אחד מהרגיסטרים האלו נדגם במשך שלושה מחזורי שעון על מנת שנקבל את המספר הרצוי. (אחריות זו היא על המידע המגיע מרכיב WBM).

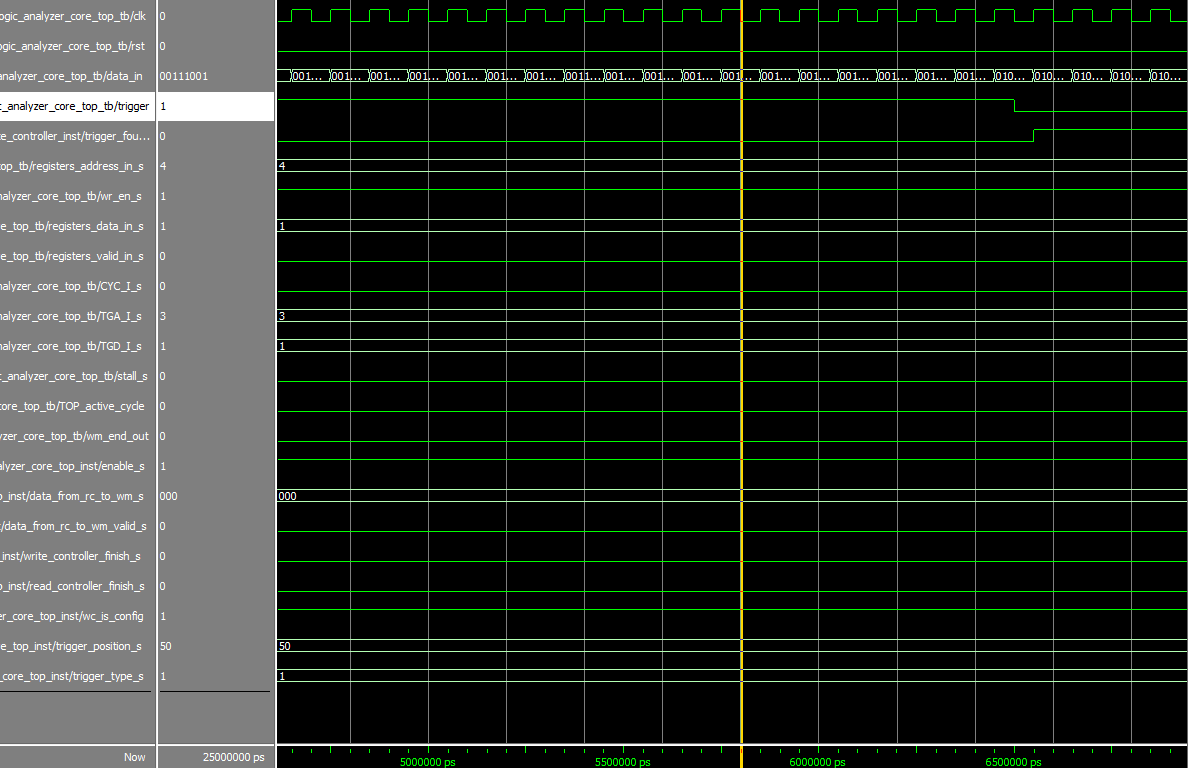
בנוסף אנו לא יכולים להוציא את המידע הנשמר במחזור שעון אחד (רוחב המידע הוא 8) ולכן גם לכך דרושים לנו שלושה מחזורי שעון, כשהוצאת המידע מתרחשת מה LSB לכיוון MSB ובמידת הצורך מתווספים אפסים לפני MSB (בתחילה בנינו את המערכת מה MSB לכיוון LSB אך הצורה כעת נראית לנו נכונה יותר, כיוון שגם בשירשור שני המספרים אנו שומרים על ערך המידע ההתחלתי שיוצא.

בסימולציה עצמה אנו מבצעים שני קינפוגים- בראשון מיקום הטריגר הוא 50% (מגיע במהלך שלושה מחזורי שעון) וסוג הטריגר הוא 1 (ירידה), בשני המיקום הוא 0 והסוג גם כן 0 (עליה).

תוצאות הסימולציה:

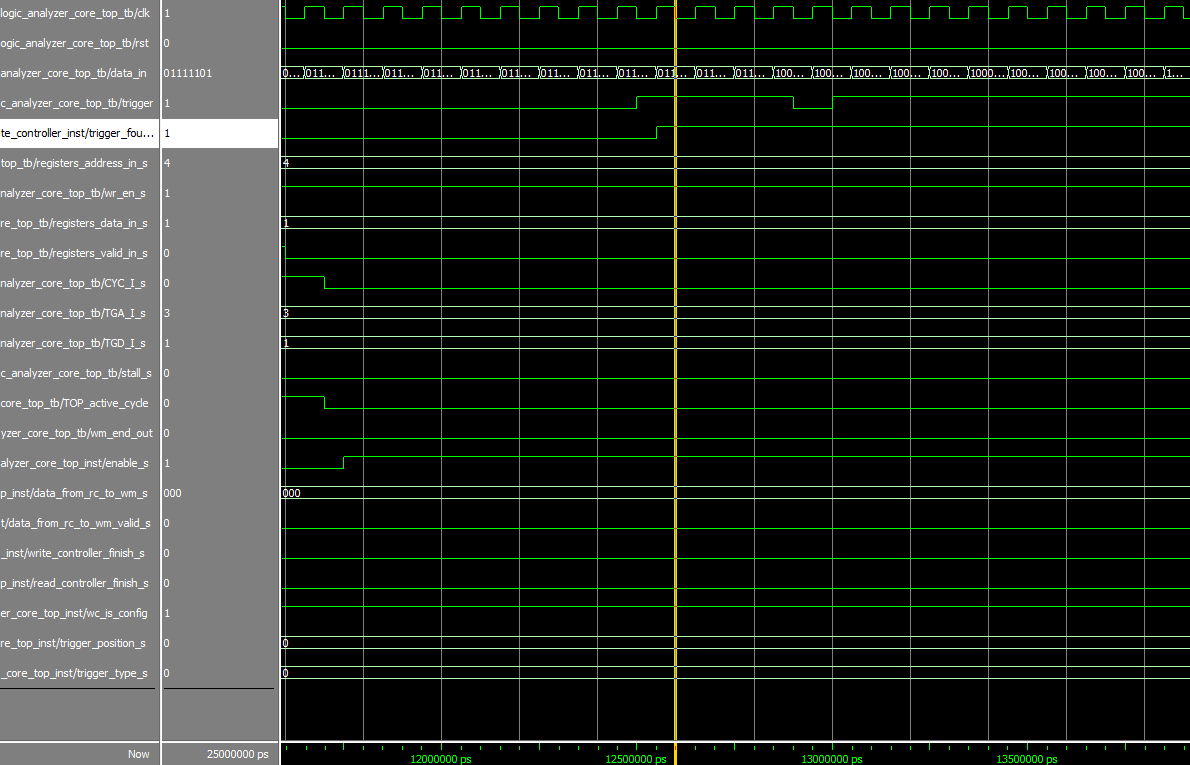


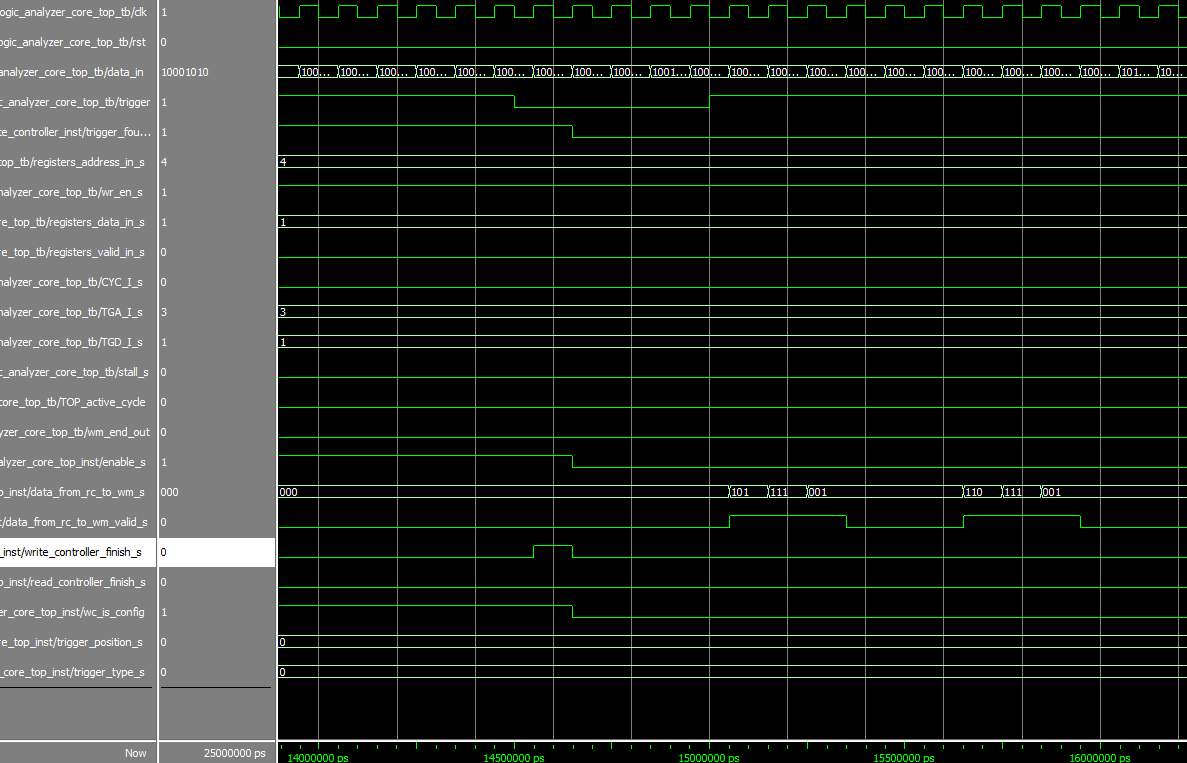


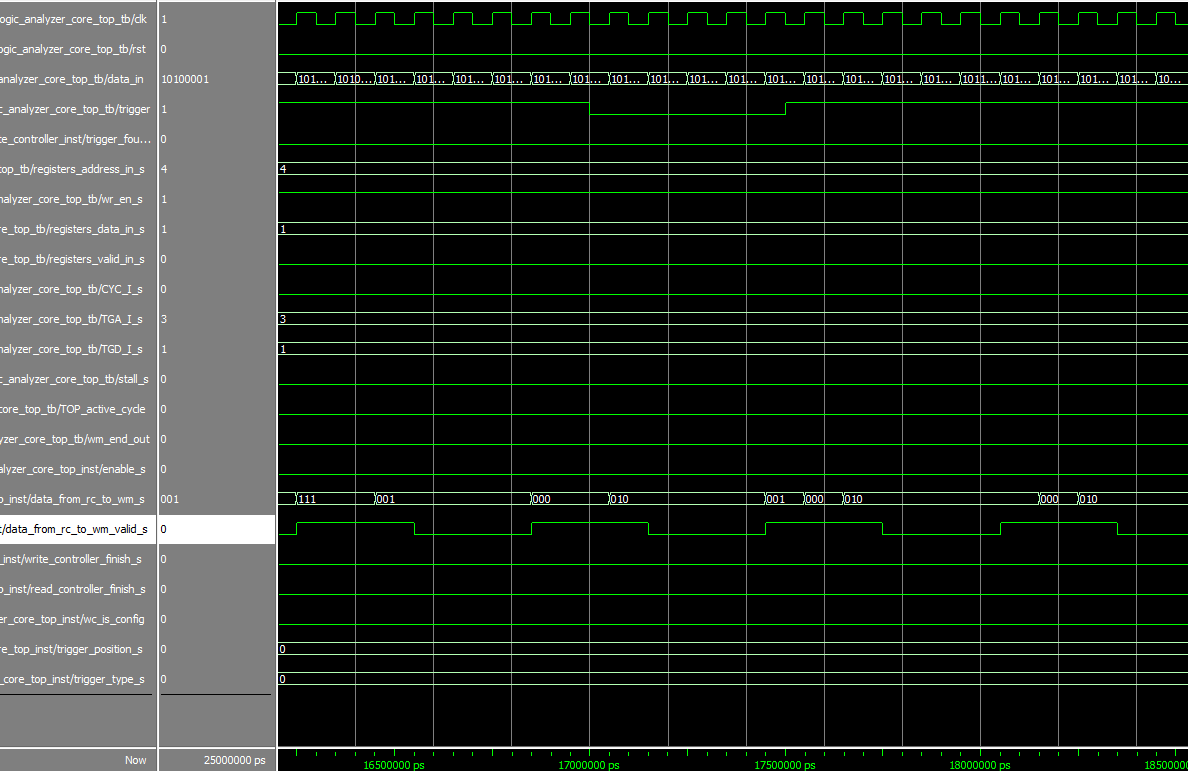


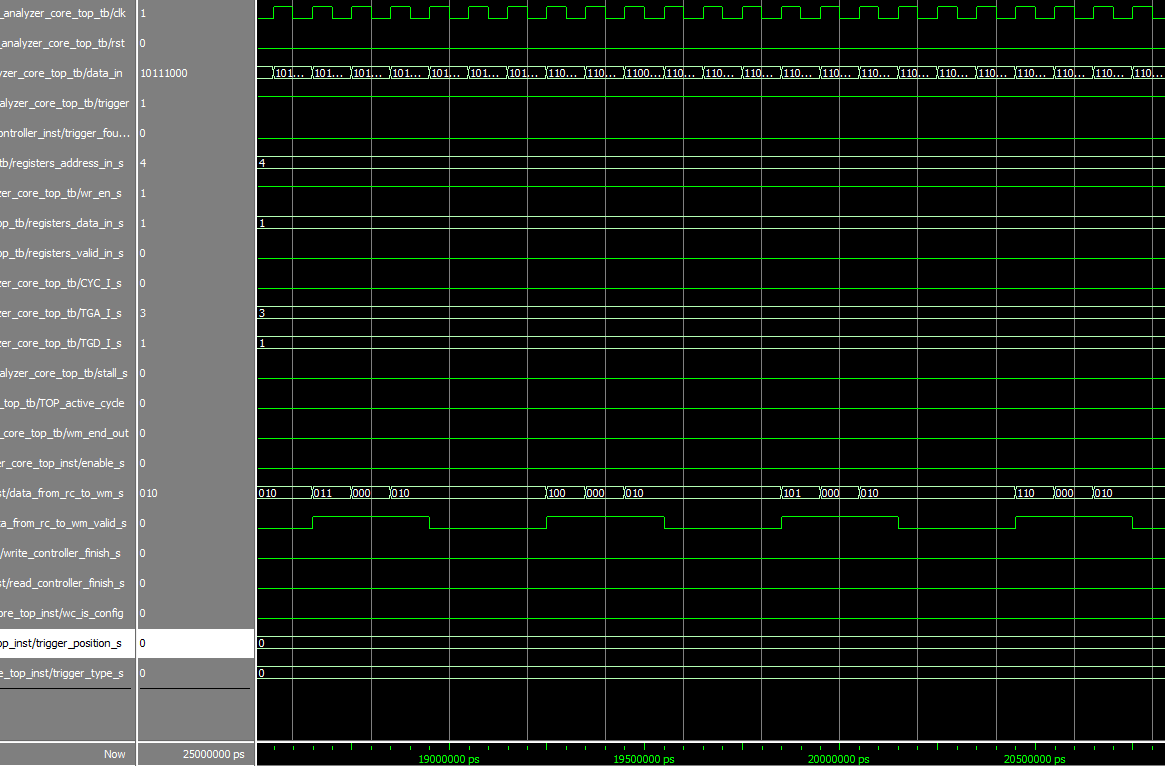


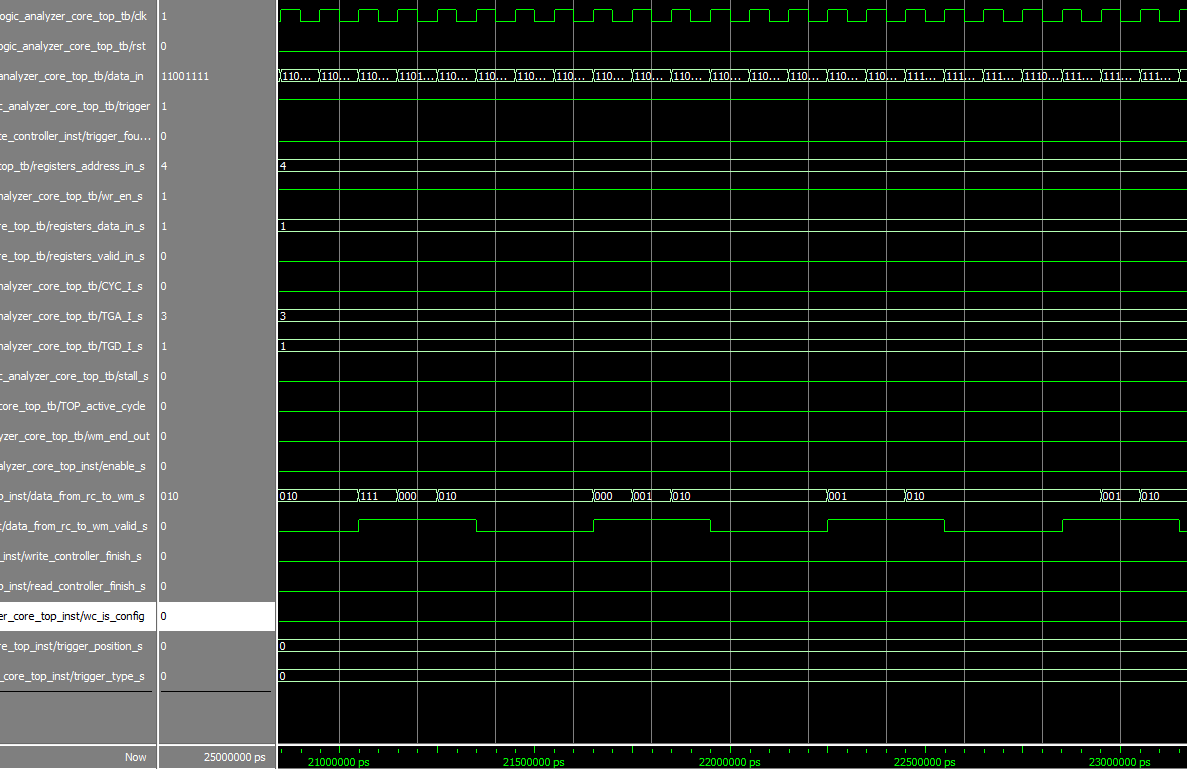


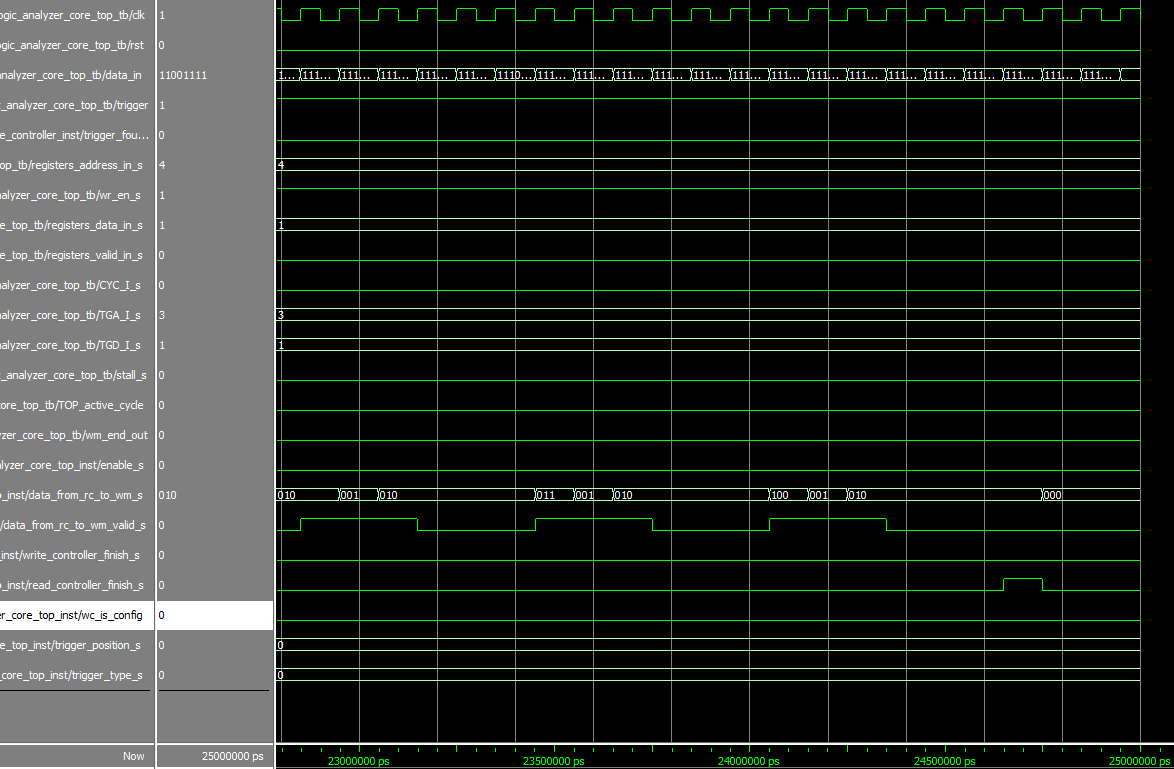












ניתוח:

ניתן בתחילה לראות כי אנו זקוקים לשלושה מחזורי שעון בכדי להכניס את הקונפיגורציה המתאימה לנו בעת קביעת התצורה. אפשר לראות כי הערך של הקונפיגורציה נקלט כנדרש בסיגנלים trigger\_position\_s, trigger\_type\_s שמקבלים בתחילה את הערכים 50,1 בהתאמה.

לאחר מכן המערכת נדרכת ועליית טריגר מזוהה כאשר מגיע מידע מספר 01000001 (ערכו 65) אך מכיוון שהמיקום הוא 50, תחילת המידע יוצא בתא מספר 00111001 (ערכו 57) שמרחקו 8 דגימות לפני עליית הטריגר. לאחר מכן המידע מתחיל לצאת מתא זה כאשר בכל יציאת מידע אנו מקבלים רוחב של 3 סיביות, ולכן תחילת המידע היוצא היא 001, 111, 000 וכפי שאמרנו אנו מתחילים את הוצאת המידע מכיוון LSB ובנוסף אנו מוסיפים 0 בתחילת ההוצאה השלישית בכדי להשלים את הסיגנל לרוחב הדרוש. בכל מחזור של הוצאת מידע מכאן ואילך אפשר לראות כי מתרחשת לנו מעין ספירה בינארית כאשר המידע היוצא ראשון (LSB) מתקדם באחד בכל פעם, ובזמן הנדרש יתקדם גם המידע הנמצא במקום השני וכן הלאה.

באופן דומה ניתן לראות כי בקונפיגורציה השניה מיקום הטריגר הוא 0 והסוג הוא עלייה, ולכן כאשר הטריגר עולה במידע מספר 01111101 המידע יוצא בצורת הסיגנלים 001,111,101 ומתקדם משם בכל פעם באחד בסיגנל הראשון, כנדרש.

טסט מספר 17

ערכי ג'נריק:

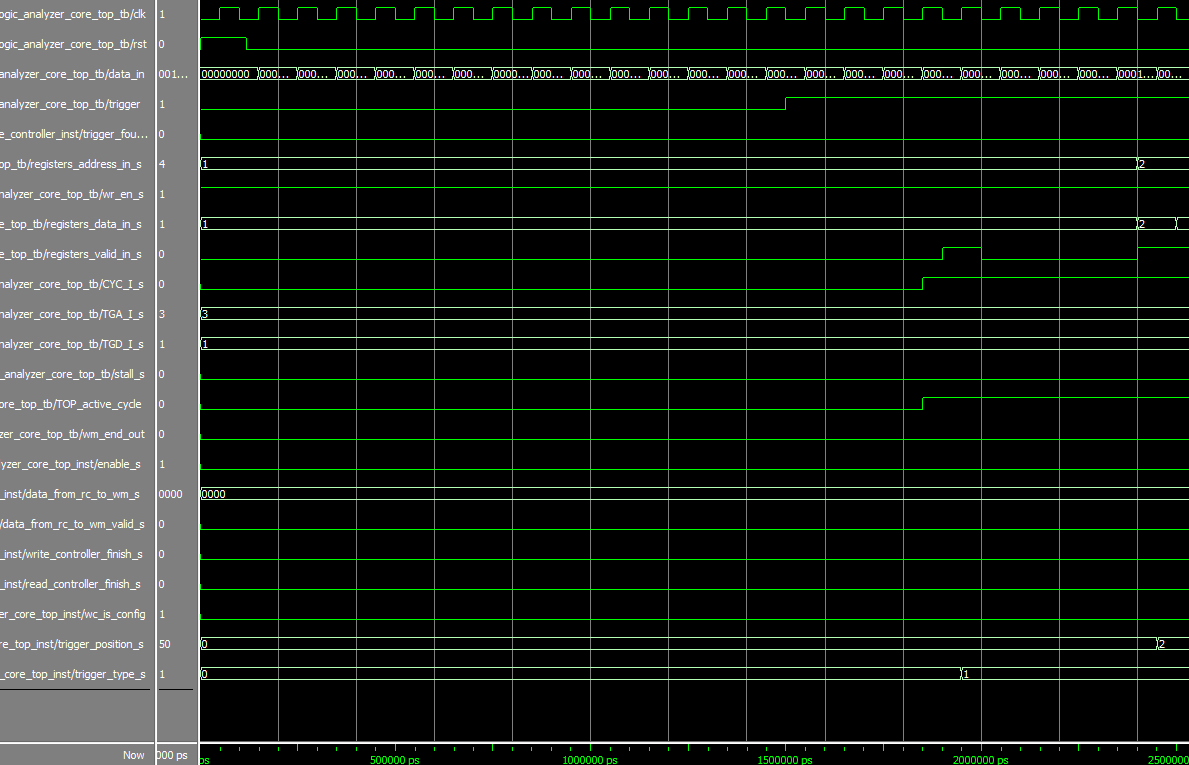
|  |  |  |
| --- | --- | --- |
| Comments | Value | Name |
|  | 1 | reset\_polarity\_g |
|  | 1 | enable\_polarity\_g |
| Each the number og addresses is RAM in | 3 | signal\_ram\_depth\_g |
| "word" width – line in the RAM | 8 | signal\_ram\_width\_g |
| The number of recorded signals is | 4 | record\_depth\_g |
| BUS width of the information entering the component | 4 | data\_width\_g |
| Number of signals we record in each iteration | 8 | num\_of\_signals\_g |
| Output width is exactly like input width | 0 | power2\_out\_g |
| Output and input width are the same | 1 | power\_sign\_g |
| Type depth. We got 3 WM and 3 WS -> 6 total whisebone entities | 1 | Type\_d\_g |
| Length depth. | 1 | Len\_d\_g |

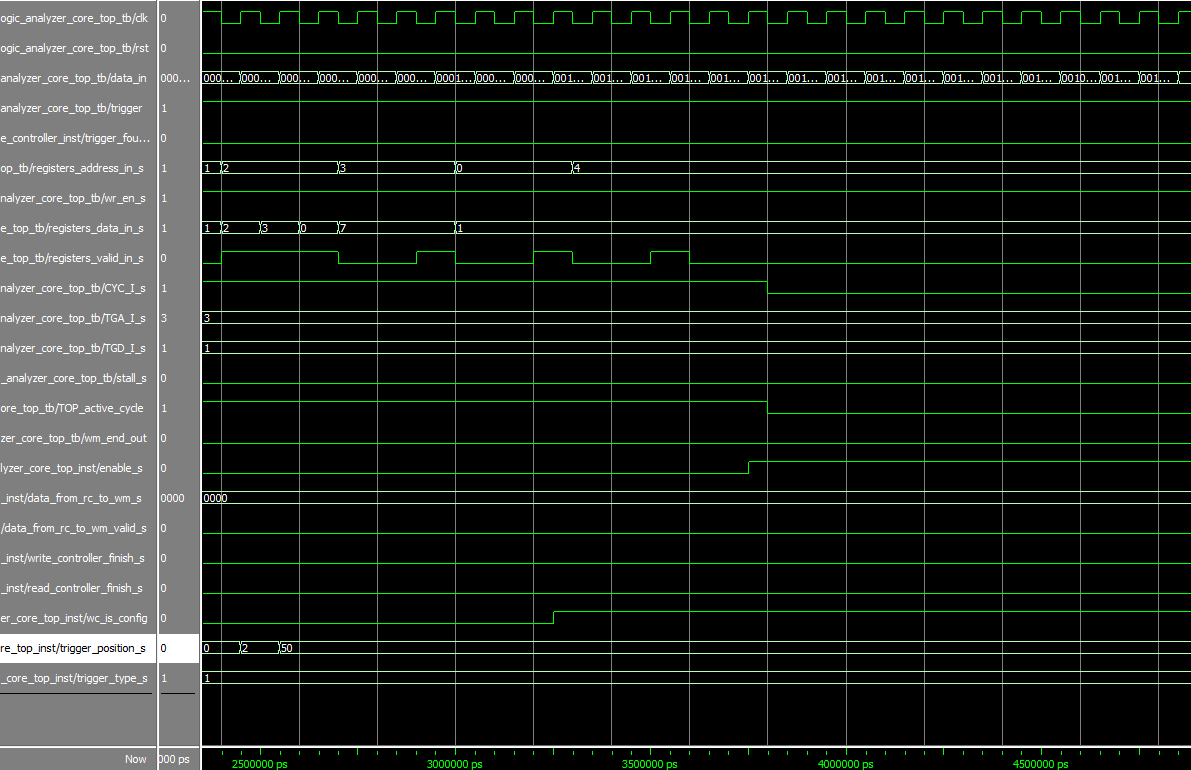
תיאור הבדיקה:

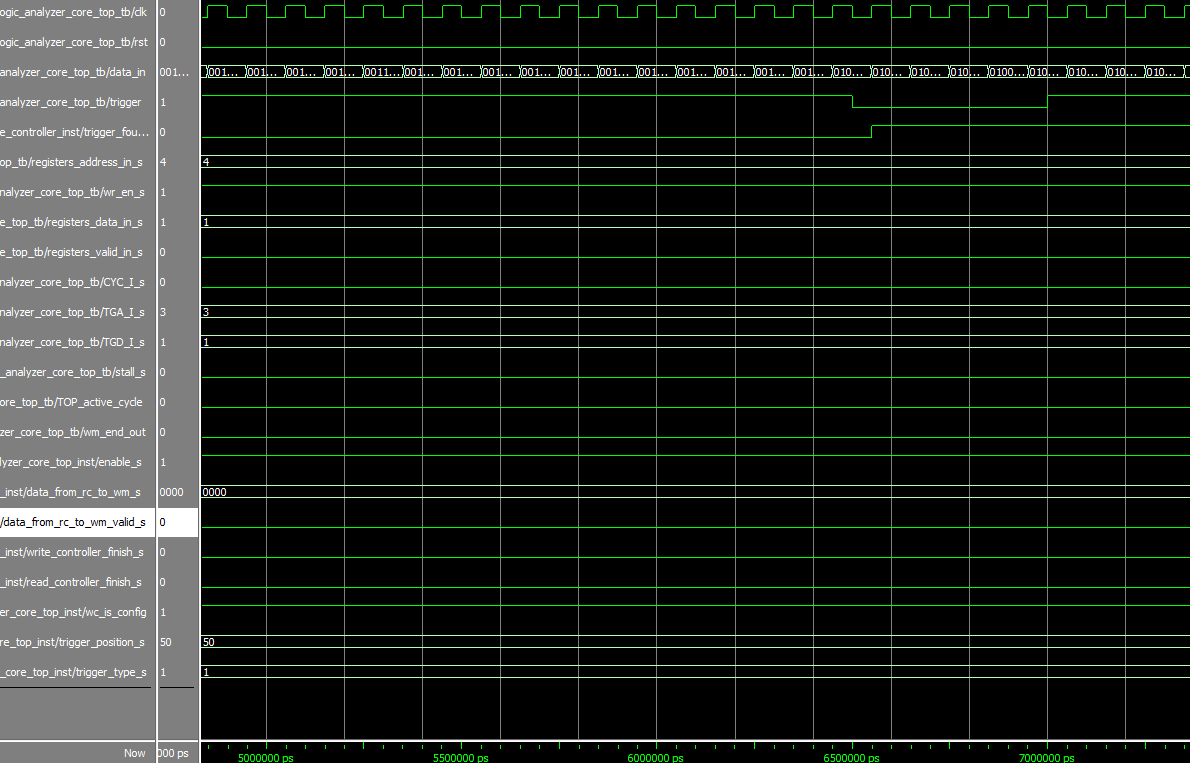
בסימולציה זו רוחב ה BUS עדיין קטן ממספר הסיגנלים הנדגמים, כיוון שרוחב המידע הנשמר הוא פעמיים רוחב הBUS אנו מוציאים את המידע במשך שני מחזורי שעון, כאשר סדר היציאה הוא משמאל לימין (כלומר מ LSB עד ל MSB).

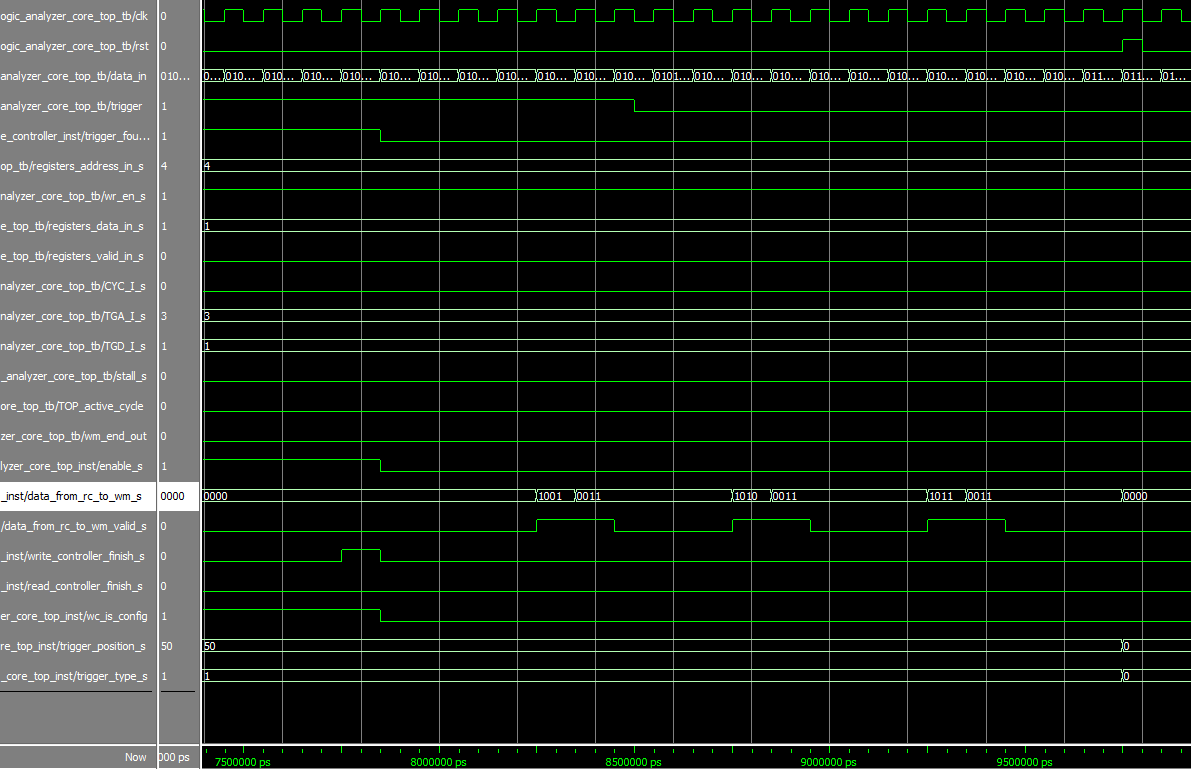
שאר חלקי הבדיקה דומים לטסט הקודם.

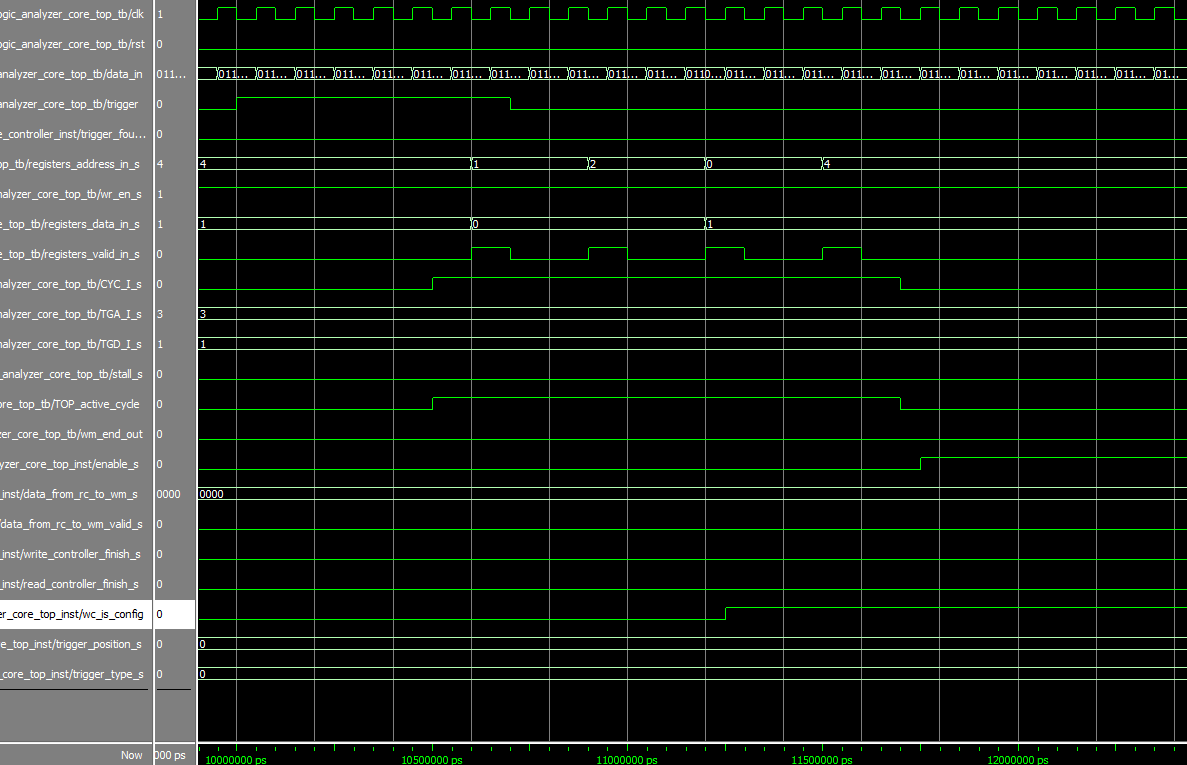
תוצאות הסימולציה:

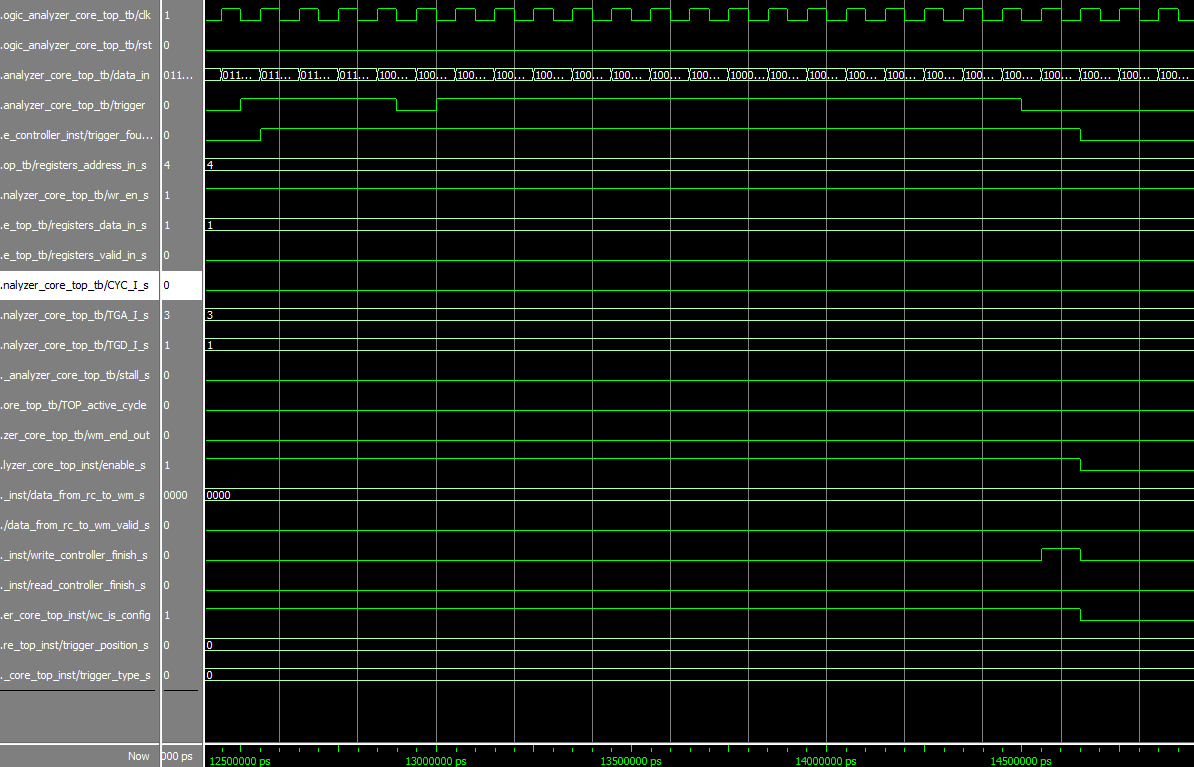


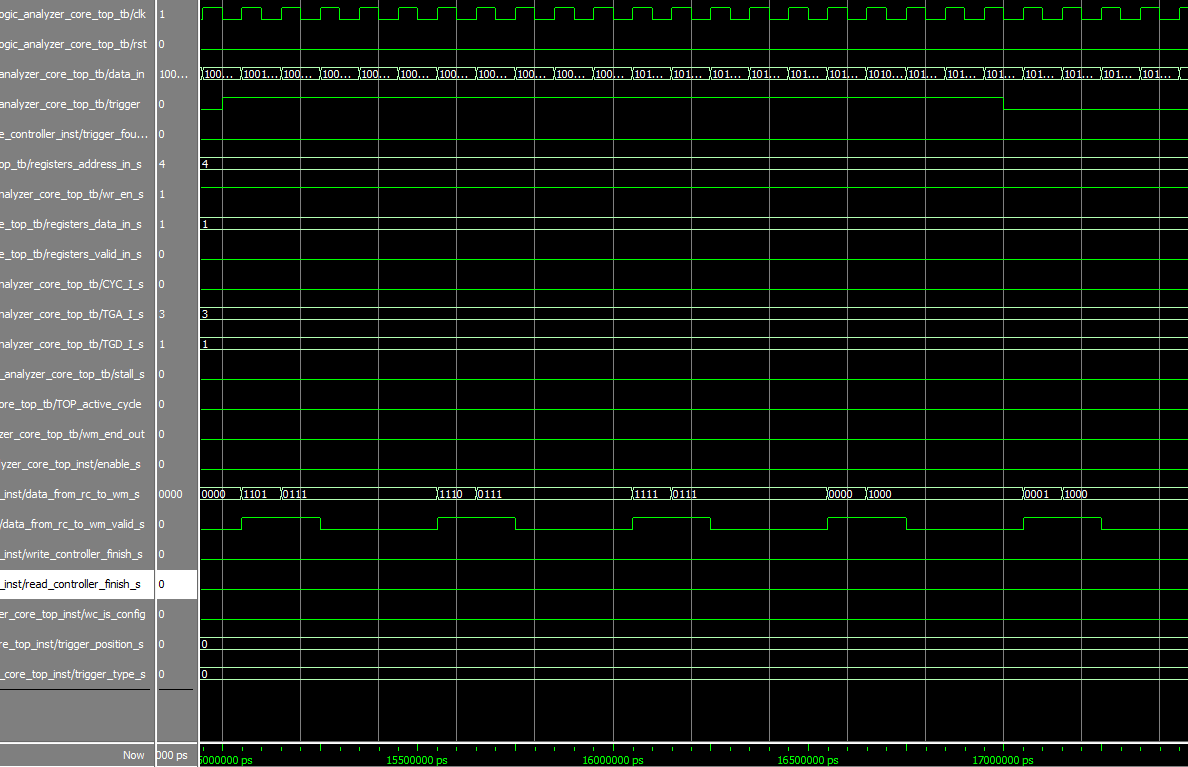


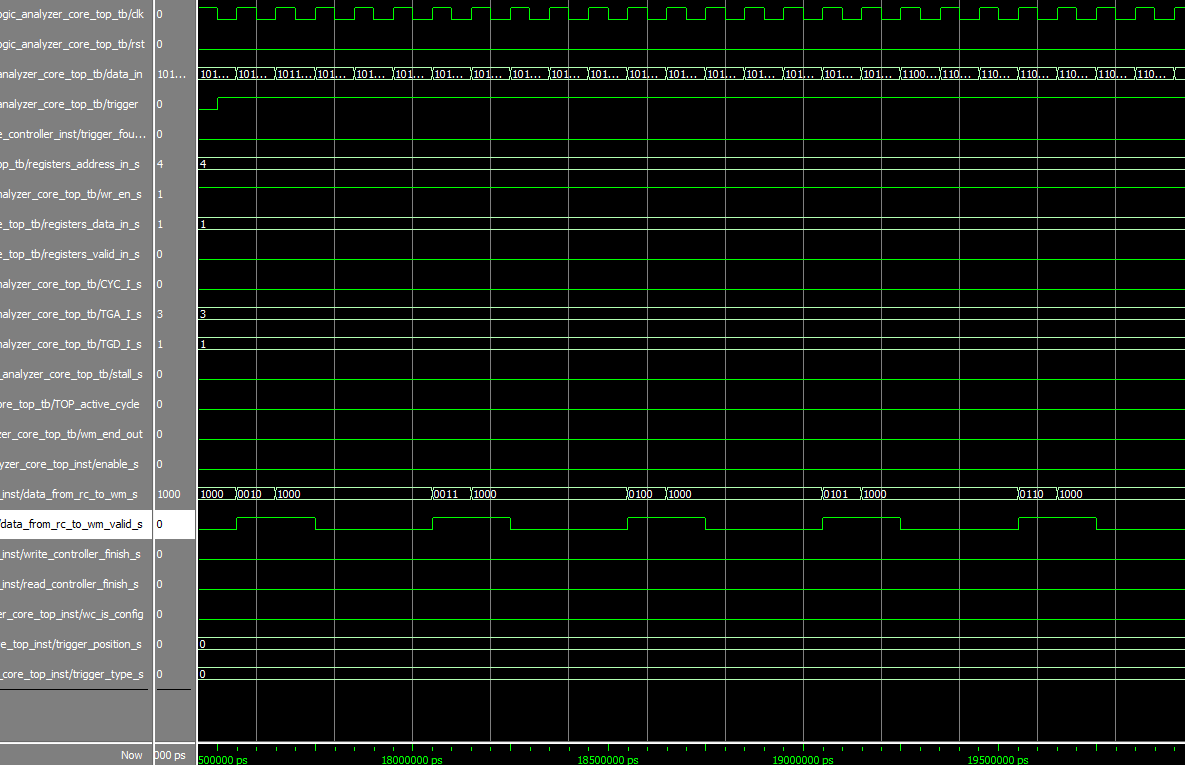




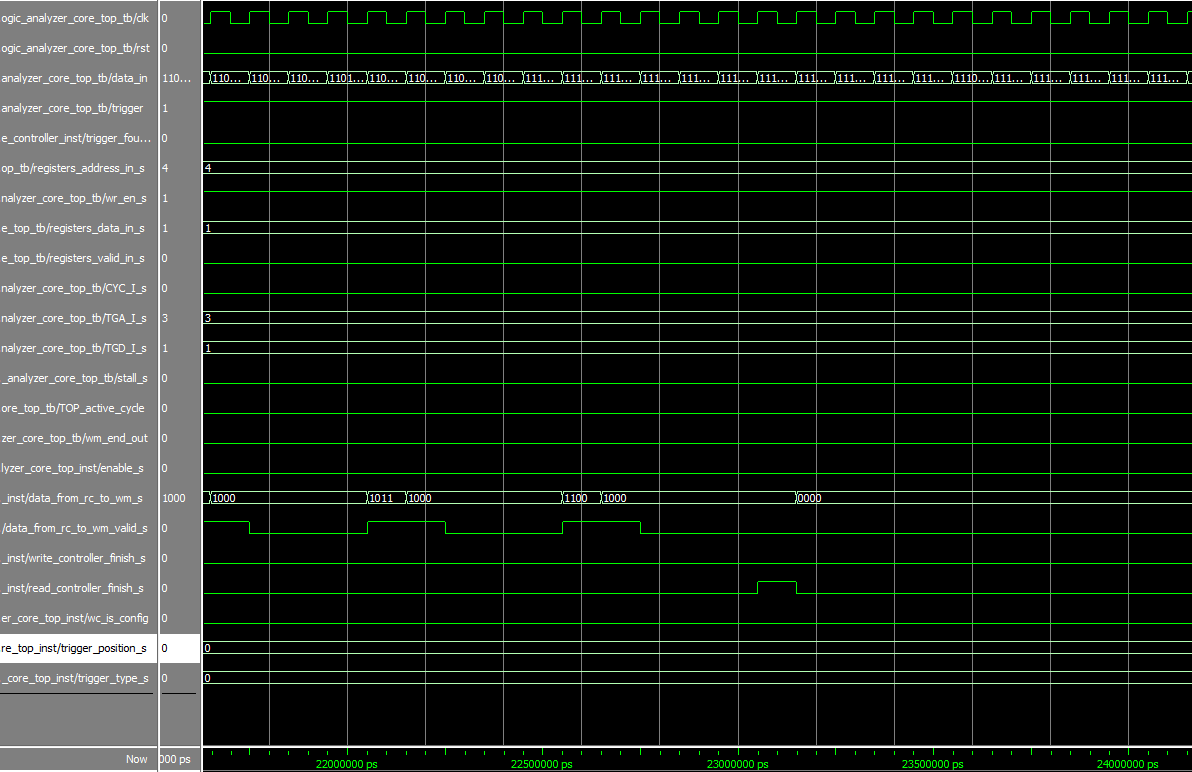












ניתוח:

נתייחס כאן רק להוצאת המידע כיוון ששאר האלמנטים כבר נבדקו והוסברו בעבר.

כפי שאמרנו רוחב היציאה קטן מרוחב המידע הנשמר, ובנוסף רוחב הכניסה הוא בדיוק פעמיים רוחב היציאה- כלומר דרושים לנו שני מחזורי שעון בכדי להוציא את כל המידע. ניתן לראות כי אות VALID של הוצאת המידע (data\_from\_rc\_to\_WM\_valid) גבוה עכשיו למשך שני מחזורי שעון בכל פעם וזאת בכדי לאפשר למידע לצאת בשלמותו. אנו מתחילים להוציא את המידע מכיוון ה LSB. בסימולציה זו, בניגוד לקודמת, רוחב המידע היוצא הוא בדיוק חצי מרוחב המידע הנכנס ולכן אנו לא זקוקים "להשלמת אפסים" בהוצאת המידע האחרונה.

אנו עובדים בסימולציה בייצוג בינארי של המידע בכדי לראות בצורה טובה יותר את חלקות המידע.

בקונפיגורציה הראשונה אות טריגר עולה במידע מספר 01000001 (65), מיקום הטריגר הוא 50 , סוג הטריגר הוא ירידה. ניתן לראות כי לאחר ירידת אות הטריגר המערכת ממשיכה לשמור את המידע הנדרש ולאחר מכן מתחילה להוציא את המידע, כל פעם למשך שני מחזורי שעון, בפעם הראשונה המידע היוצא הוא 1001,0011(57) מחולק ליציאה 1001 וליציאה 0011 שמרכיבות את המידע "השלם", לאחר מכן רק היציאה הראשונה מתקדמת בכל פעם באחד (ספירה בינארית) מה שמצביע בעצם על כך שהמידע היוצא הוא מספרים עוקבים (כפי שאנו יודעים שהמידע הנכנס מכיל) ולכן המידע היוצא תקין.

בקונפיגורציה השניה מיקום הטריגר הוא 0 וסוג הטריגר הוא עלייה, ובאמת אנו רואים כי לאחר עליית הטריגר במידע מספר 011111101 המערכת ממשיכה לשמור את המידע הרלוונטי ולאחר מכן המידע היוצא מתחיל בסיגנלים 0111, 1101 ומתקדם בכל פעם באחד, שוב, בספירה בינארית. כלומר גם כאן המידע היוצא תקין.