Command Processor Sub-team Report

Author : Ziyang Liu, Lingzhi Xie

Email: [ao18669@bristol.ac.uk](mailto:ao18669@bristol.ac.uk), nc19102@bristol.ac.uk

1. Abstract:

In this project, our sub-team’s main purpose it to implement our command processor module, which could make different information transmission purposes by type specified commands in the PUTTY terminal, lastly print out the requisite responses. According to unexpectable circumstances, some of our original plan was cancelled due to COVID-19.

1. Group composition

Teamwork is very important for engineer, it doesn’t mean everybody doing the same thing or everybody being able to do each other’s jobs. It’s more a means to a synergistic way of working, where the sum is greater than the parts.

Properly managed, teamwork maximizes strengths, bringing out the best in each team member, a key theme on this site. These specific, possibly unique individual strengths are then complimented by the strengths of others, or of the team as a unit.

However, our group is form randomly with members who have different culture background, some of us probably even never talked to each other before, we still overcome the distance between us and try to work together. Although the group environment in our team operate not very ideally underpinned by strong communication, we are very supportive, positive and undergo with plenty of encouragement. And of course, we have several effective methods of establish and discuss about individuals’ task with an opportunity for everybody to feedback and provide opinions and draw on others’ experiences to collectively solve problems like whatsapp, google share, skype or wechat, etc.

1. Task division and individual contribution

**Task division:**

Ziyang Liu (ao18669)------- Transmitter (Tx) Interface and Receiver (Rx) Interface.

Lingzhi Xie (nc19102)-----------Command Processor <=> Data Processor Interface.

Ziyang liu: It is a new and unordinary experience for me as this project is the first group coding assignment. Since I had python individual project in year one, I am interested in python and willing to explore during my leisure time. I’d like to claim myself as coding enthusiast. I was very looking forward for this module and had a lot of interests of VHDL. For this task, I have been doing Txdata transmission, which it should more focus on switching different data source between Rx and byte data from data processor, and also some counter logic design, in commander processor, it required many counters, I specified some of them, and working properly.

Based on Coronavirus pandemic, our original plan on L and P commands had cancelled, but because of our high productive team work and efficiency, we finally successfully implement these commands on testbenches even it is not required, but I do feel more achievement.

There were a lot of difficulties and frustration at the beginning, as I did not well-understand the digital logic design. So that, me and my group mate spent a lot of time to work together, exchange our ideas and discuss this project. Afterall, I realized that teamwork is essential to success and growth.

As much as some people like to think they can do it all on their own, the truth is, they can’t. Having a good team of individuals around is not only helps me to stay grounded but also enables to think in different ways, push boundaries with new ideas and most importantly it provides with a strong support network.

Lingzhi xie: My part is basically work in state\_logic: process. Once the command processer detects the data, it needs to distinguish A or a first and then contract data from data processer. Obviously, I cannot work if there are any modules of transmitter and receiver unfinish. This project requires good interpersonal relationship, ability to communicate and be open, honest and respectful to others ideas.

Me and Ziyang spent most of time work together everyday even during Easter holiday. Working at a distance is very different from working physically collocated. I have to rely on electronic means of communicating with my team mate and the sacrifice personal time is necessary.

To be honest, it is very difficult to start this project since it requires understanding of complex working systems and digital logic in FPGAs, hence, we try to share our ideas, knowledge and skill as we have different talent and understanding about VHDL and it is important that a team works as one, that everyone has the same end goal and are helping each other to reach it.

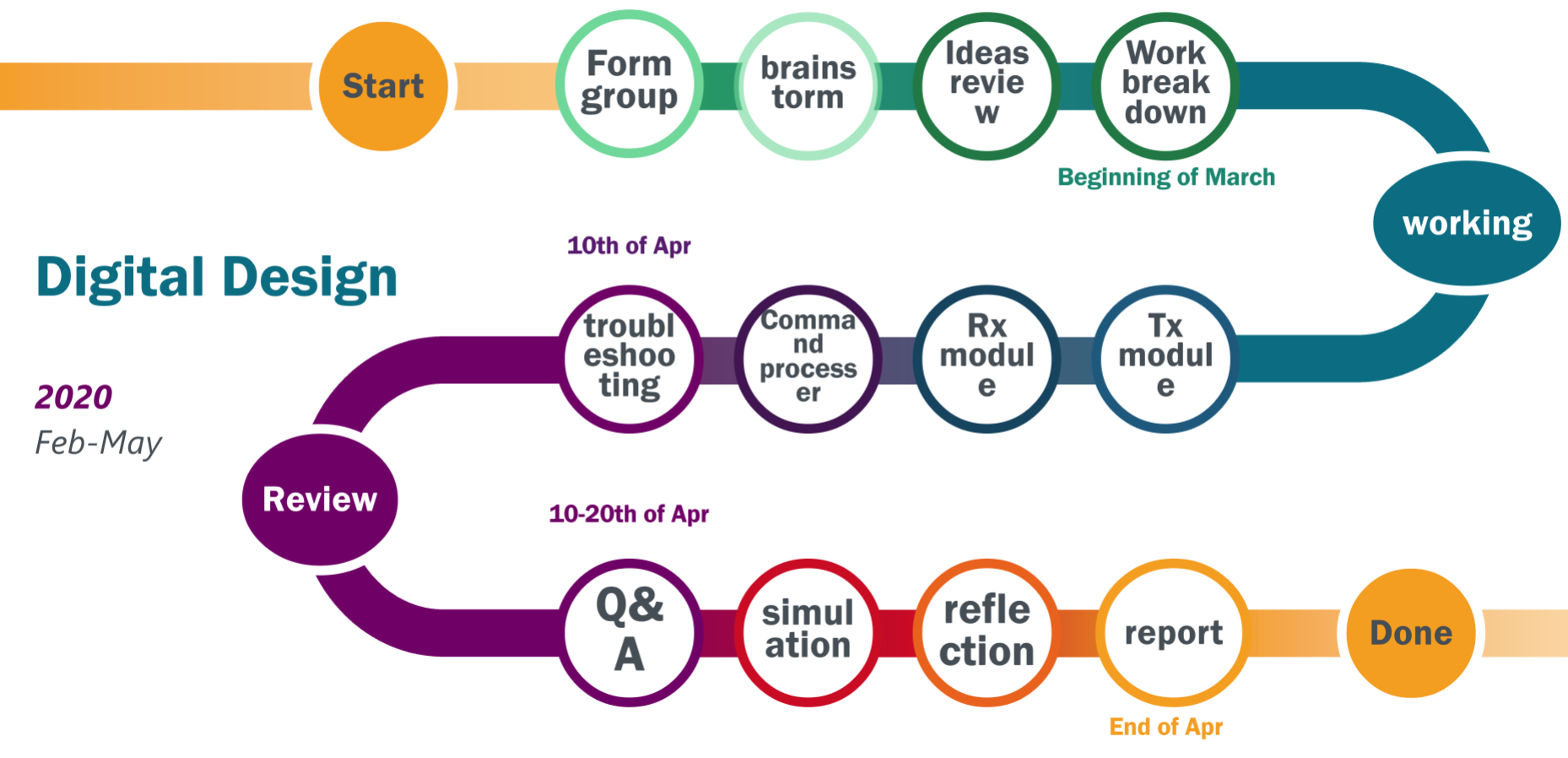
1. Project Plan

A project plan is essential for keeping a project in track, so that we made a timeline for our project. Firstly, we have to brainstorm before setting it as digital design worth for 10 credit, we treated this project seriously and make sure that every team member has the same goal and welling to achieve by effort.

At the first few week, we will set a group and brainstorm together in the class. The most important criteria obviously of this project is to understand each module and prototyping digital logic in FPGAs. We will share our ideas to discuss and review about our understanding.

After we all agree with the objective, we dived the task and start working on it. By following our project timeline, it is reasonable to troubleshoot about our problem by 10th April as we spend more than one month to code and still, we need enough time to finish our report.

Afterwards, it will be a easy job after troubleshooting if our code has no problem, we will finish our report by the end of April as the deadline is 1st of May.



Nevertheless, it has been tough to code and we faced more problem than we thought. The Gantt chart seems not following our project timeline simultaneously. We realized that this project is more difficult than we estimated, so that we started modifying it while brainstorm, sharing ideas and reviewing ideas. Our understanding at the beginning is against the purpose of the project, with help of TAs, we were able to get the real meaning of it which helped us to save much more time. Despite coding, our team had meeting not only during school time but also during self-isolating. None of us will expect this term is ending by work at home and it is surprised that we did quit well.

1. Gantt Chart

This Gantt chart has illustrates how our group work process



1. Command Processor Modules Description

(i). State\_logic Process:

In this part, it is the first process part of our architecture before the entity and signal statements. It describes how our state changes as different condition fulfilled.

In first case, it is named IDLE, which means very first one process, as rx\_now signal equals to 1, it will jumped to next state, as rx\_now = 1, it means information has arrived at the input, so data is transmitted from Rx module, it starts transmission, if not it will still stay at idle state until first Rx data valid signal has detected.

Second case named DETECT\_A, according to specifications, our first goal is to decide how many numbers of BCD should give to the data processor, but before that, A is the signal where we could start our counting BCD. Therefore, there are two As, either capitalized or not, so based on ASCII table, if rxdata is “01100001” or “01000001”, it will jump to next state and start counting BCD.

Next state named LOAD\_VALUE. First we established a counter to count 3 times for BCD, if rxData is between 0 to 9(According to ASCII table), so enable signal should asserted std\_logic 0, after that, wait until rxnow equals to 1 and txdone equals to 1, then jump to next state and starts examine new rxdata transmitted from Rx module is between 0 to 9 whether or not.

At DETECT\_RANGE\_FOR\_Ns state, a new BCD counter has built, which is used to record BCD number in BCD\_ARRAY\_TYPE, also count 3 BCD numbers and make sure these 3 numbers successfully pumped in. However, if rxData is between "00111010" and "00101111" according to ASCII table it will enable the counter to start count, state will jump between DETECT\_RANGE\_FOR\_Ns and LOAD\_VALUE. As when down counter signal counter\_N equals to zero, it means AXXX has successfully received, jump to next state S2.

In S2, reset the BCD counter, to ensure it initialized for next new cycle transmission. Jump to S3.

In S3, from data processor, dataresults and maxIndex signal will give input to the command processor. New signal or variable is used to store these values. Then wait until seqDone signal equals to 1, jump to next state, it means data processor has finished all his work and wait for L or P commands.

In S4, reset our down counter, initialized it if there is a new sequence of AXXX. In this case, use if statement to divided into different situations. Firstly, if new rxData is L or lowercase ‘l’, it will jump to OUTPUT\_LIST\_VALUE state. Secondly, if it is P or ‘p’, go to OUTPUT\_PEAK\_VALUE state. Moreover, if comes around with A or ‘a’, it will jump back to LOAD\_VALUE state, and wait for a new sequence of ANNN.

In OUTPUT\_PEAK\_VALUE state, it assert output signal to 1, in order to print out peak value of dataresult, which is forth element in Dataresults BCD array type. Then it will go to OUTPUT\_MAX\_INDEX state, at this time, another counter is required for output purposes, as there are 3 elements of index from the max\_index output, and after all of these 3 max index has successfully printed, it will ended in DESTINATION state, then jump back to idle state again, and wait for next information has reached from Rx module.

In OUTPUT\_LIST\_VALUE state. In this case, our aim is to list all seven bytes stored in dataresults signal, then one more counter is needed, after it count 7 times, txdata will print out those 7 bytes on the PUTTY terminal, then jump to destination state.

In Destination state, it will come back to idle(start), wait for next command.

As when current state is OTHERS, it will back to IDLE automatically.

Sub-chapter editor: Ziyang Liu

(ii). STATE\_FLIPFLOPS Process:

In this part, it simply described how our current state changes between nextState signal, as global reset equals to 1, current state will jump back to IDLE, as every time jump to next state, it will require one clock cycle.

Sub-chapter editor: Ziyang Liu

(iii). OUTPUT\_MAX\_INDEX\_TRANSFORMATION Process:

This process demonstrates how we transform max index signal to 8-bit txData output, and also count 3 times to ensure all 3 max index number has successfully transmitted. Same as global reset equals to 1, counter set to 0, MAX\_INDEX\_8\_bit signal set to X”00”. As output\_maxindex signal received, it will start transform the max index data to txdata, which means from BCD array type information to 8-bit Txdata, that it why we made so many if statement for 0 ~ 9 conditions, and every time it output the txData, counter will plus one, until all 3 of them has received it will stop count, which stated in the case statement(state\_logic).

Sub-chapter editor: Lingzhi Xie

(iv). OUTPUT\_COUNTER and BYTE\_COUNTER\_LOGIC Process:

In output-counter process, we specified a up counter, this counter is to ensure all 7 bytes from dataresults successfully output.

Same as byte-counter-logic process, it is used to count how many bytes has transmitted, in order to record the number to ensure the conditions has not met.

Sub-chapter editor: Lingzhi Xie

(v). TX\_DATA\_OUTPUT Process:

This process defines how we deal with txdata port when each signal has reached, and this part is very important to our transmission between txdata and information required to print out. Firstly, global reset, as rxdone goes to 0, and txdata goes to X”00” which means 8-bit “00000000”, and start signal to 0.

First if statement, as seqdone signal reached 1, start signal should goes to zero, it means one whole transmission has completed, start signal should reset to zero and wait until new sequence of ANNN has occurred.

Second if statement, as if rxnow = ‘1’ and txdone = ‘1’, this means rxdata has ready for transmission, and txdone is initially high. Then Rxdata could directly goes to Txdata, and rxdone should set to 1, it symbolized our transmission between rxdata and txdata has finished, rxdone set to 1, it will set the rxnow to 0, like it is telling Rx module, I have finished this work, wait for the next.

Next if statement, as list\_enbale equals to 1, it means it is jump to OUTPUT\_LIST\_VALUE state, and it should output all 7 bytes stored in dataresults signal, every time txdata has output, we should set rxdone signal to 1, to ensure the reset of rxnow signal.

In output-peak-values if statement, it simply output the index 3 from the length 7 list dataresults signal.

In ourput-maxindex if statement, it ensure output every 8-bit data to txdata port.

However, if dataready equals to 1, it means byte signal from data processor has arrived, directly print it out.

Next statement, it shows when we should set start signal to 1, as when down counter for BCD numbers reached 0 and also seqdone is 0 means it has not started transmission yet.

And every time, rxnow signal equals to ‘0’, rxdone should set to ‘0’, which means there is no data from rxdata, it has not arrived yet, need wait until the rxnow signal goes to 1.

Sub-chapter editor: Ziyang Liu

(vi). TX\_TRANSIMISSION Process:

In Tx-transmission, it controlled how Tx module works between the signals txdone and txnow. In Tx module, txdone always remains initially high, therefore txdata initially ready for the transmission. As global reset, txnow set to 0, then txdone will automatically goes to 1. However, txnow signal is behaves like a trigger operation, and should not be asserted unless txdone is high, that explained how I construct this if statement. Lastly, txdone signal goes low once txnow has begun and remains 0 until it completed, therefore, as txdone goes to 0 then txnow asserted 0.

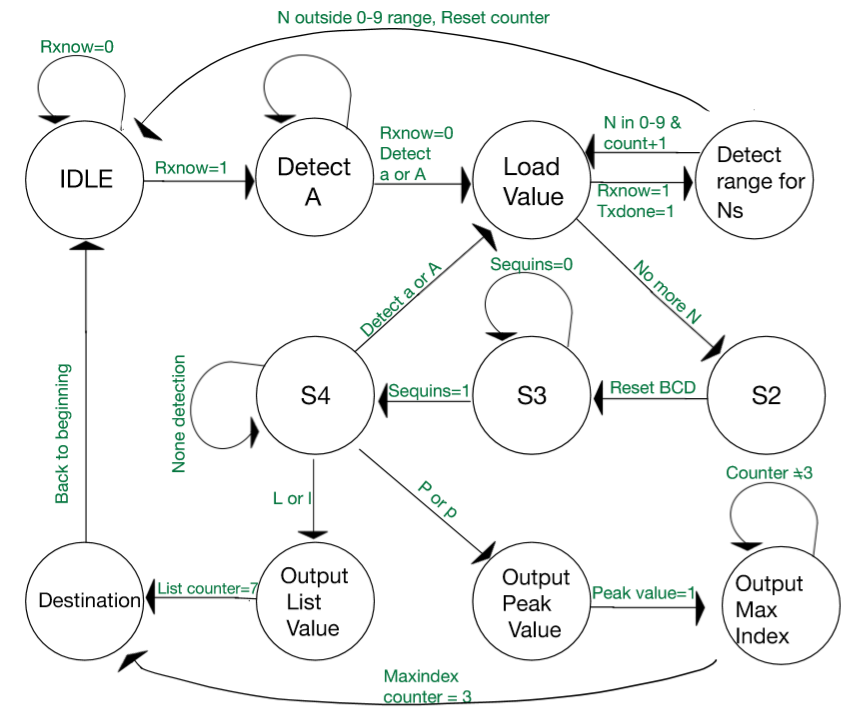
Sub-chapter editor: Lingzhi Xie

(vii). Down-counter Process:

Last component, down counter. It count 3 BCD numbers from Rx first, then convert 8-bit number to BCD array type. And also INTEGER\_STORED signal is used to iteration of BCD numbers, this is used to judge whether NNN times transmission from byte signal to txdata signal(BCD number from ANNN) has finsihed.

Sub-chapter editor: Lingzhi Xie

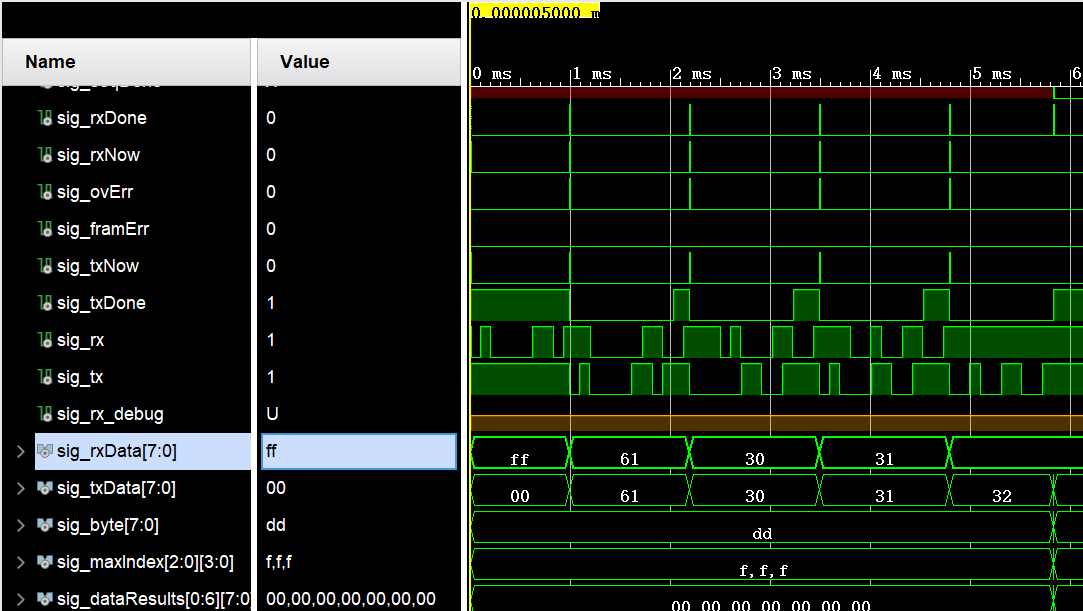
1. State diagram



1. Simulation results

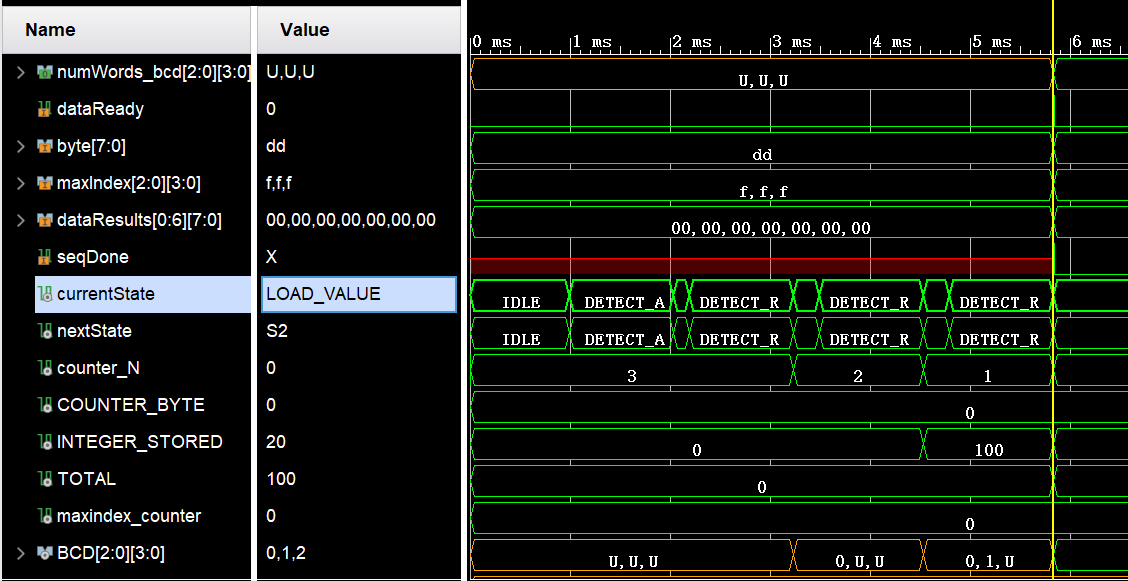
At this stage, we ran our cmdProc.vhd file in the commander\_interim test bench, and due to our team’s high productivity and efficiency, our cmdProc also could pass the full system(including L and P commands), which gives sensible output. However, due to COVID-19 pandemic, as our original L and P was cancelled, so this simulation result only comes from interim test bench.

1. First 5 milliseconds.

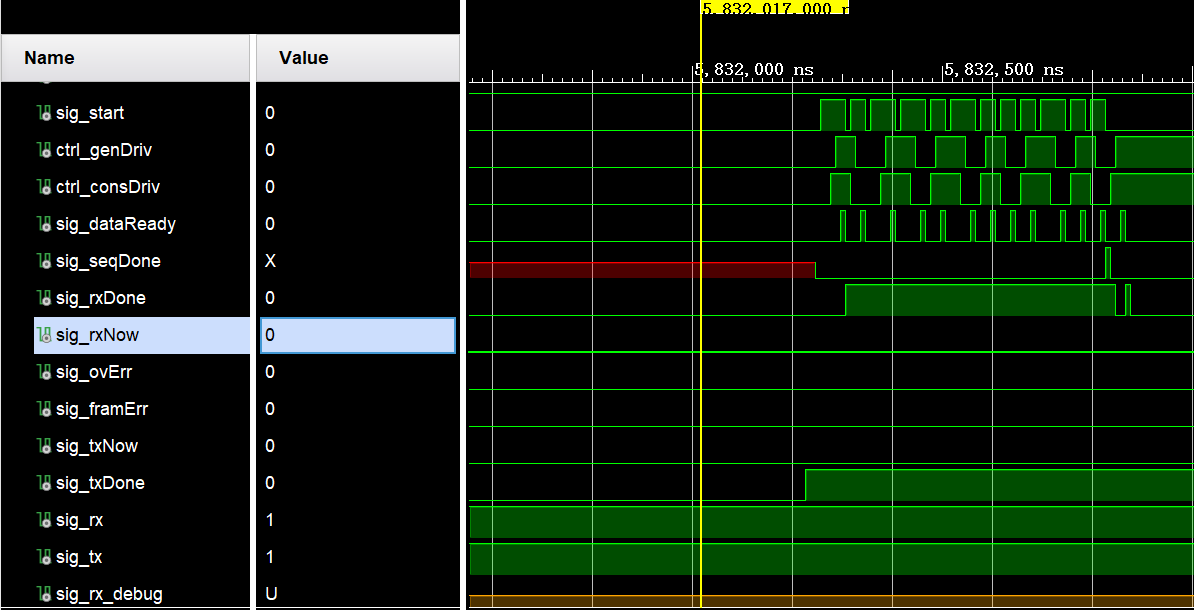


For the First 5 milliseconds, it is clear that first A has successfully detected(61 from txData), and 30, 31, 32 after ASCII transform, it is 0, 1, and 2, it gives exactly the same result as noted in testbenches code(A012).

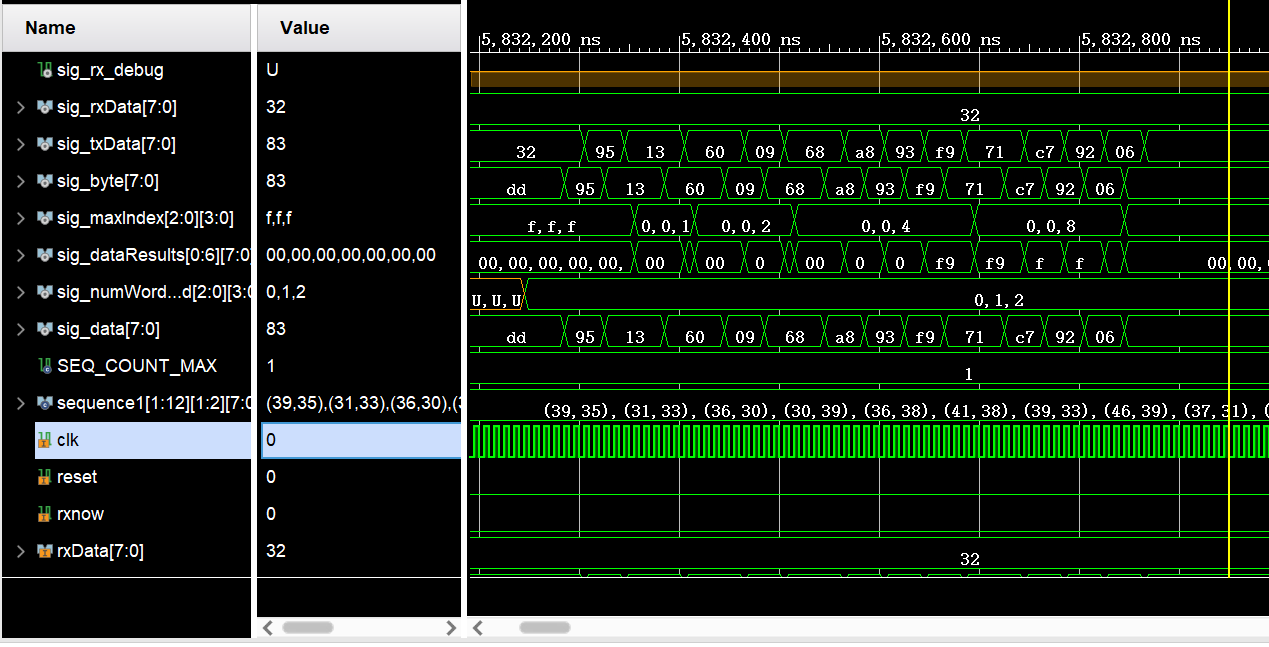
Now the graph below shows our state had successfully output the state changes, and counter is also working properly.



1. After 5 milliseconds.

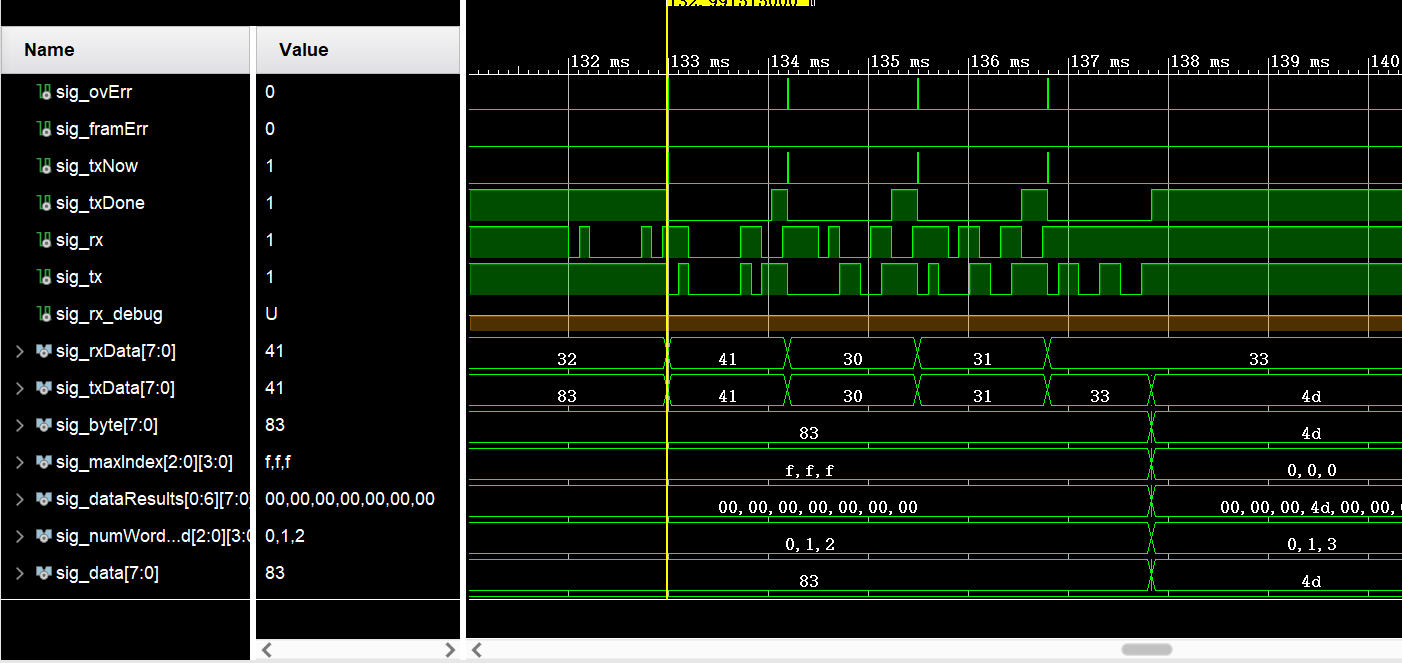


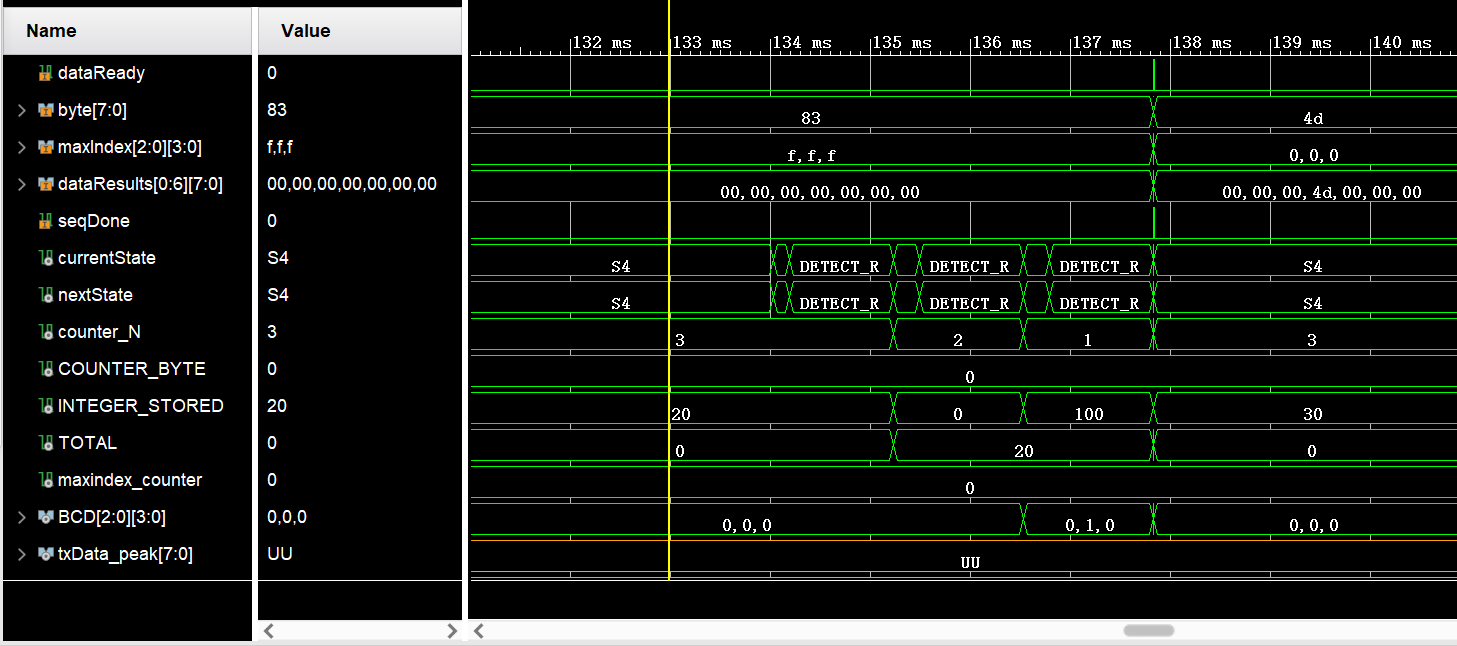
After start signal gives 1, it starts transmit the byte data from data processor port, every time a success start signal will let data processor asserted 1 in dataReady input port which shows above.



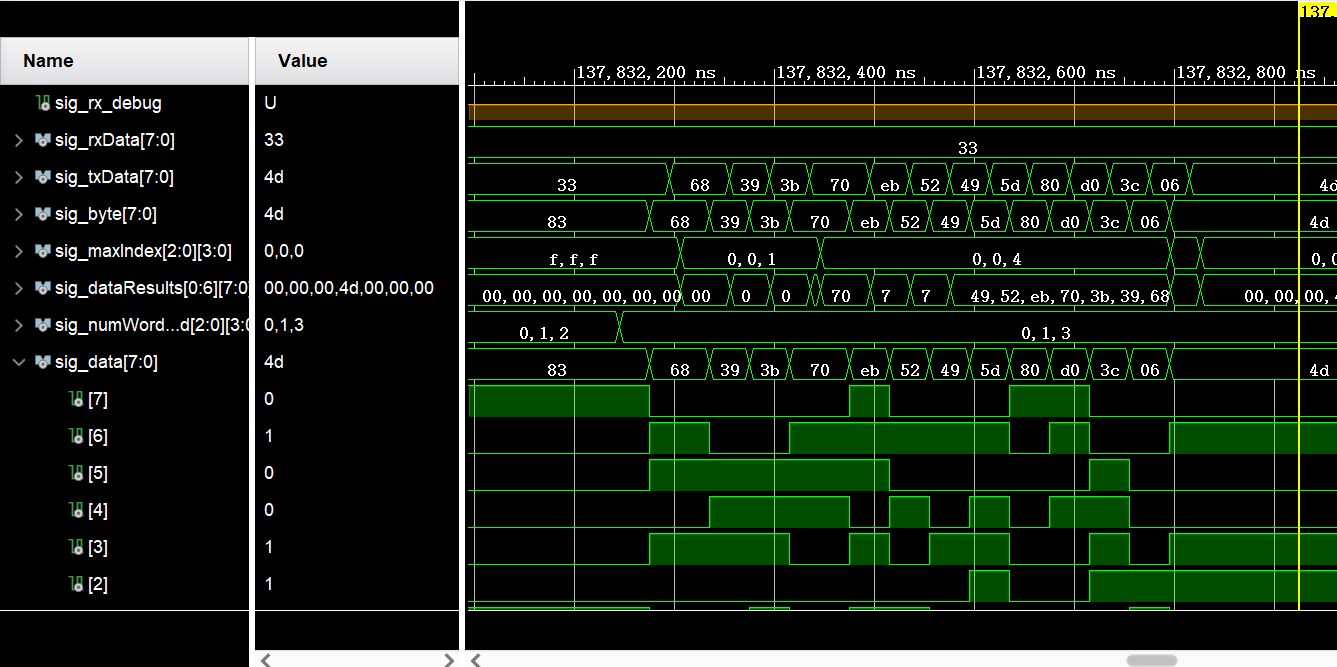
And we could see, rxData still remained in 32, but txData start output the byte data from sig\_byte, and there are exactly 12 byte data has successfully printed out.

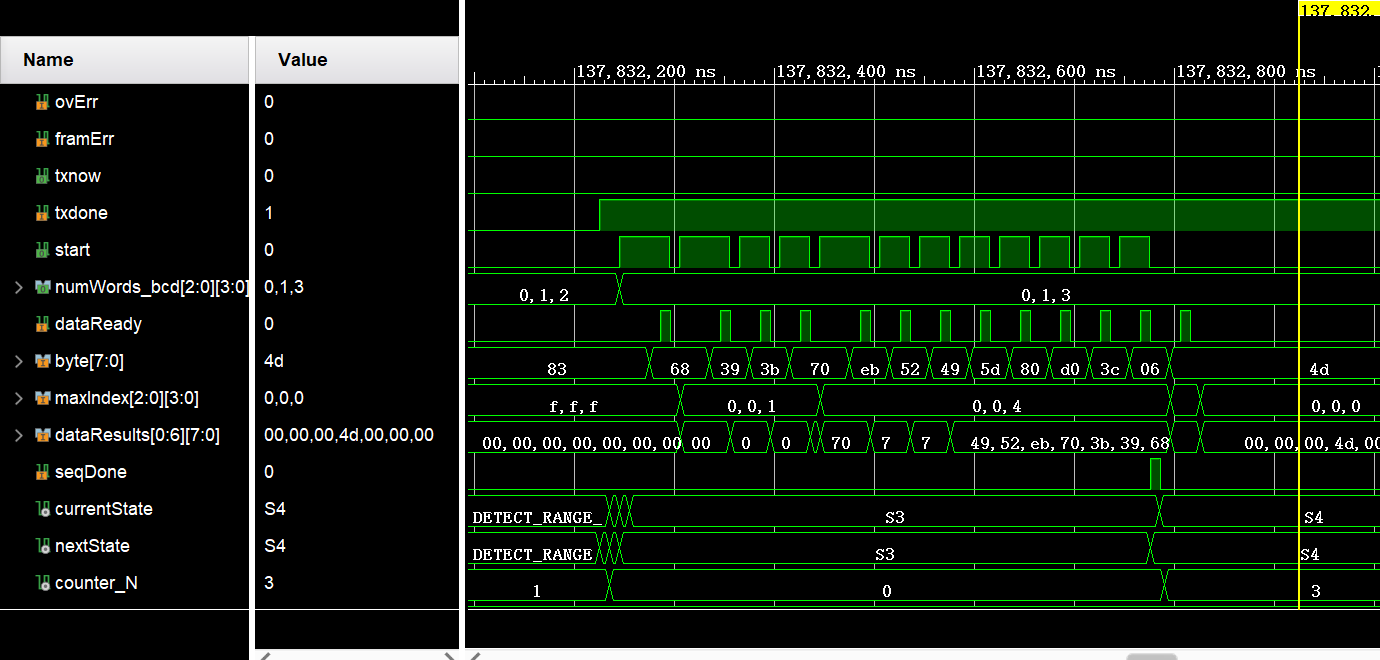
1. Second Sequence after 130 milliseconds.





However, after approximately 130 milliseconds, as test bench arrived at 133ms, it shows a new sequence of ANNN(N stands for BCD number), it clearly shows current state has return to IDLE after S4, because L or P commands is not received, but A character has detected, therefore it goes back to IDLE state. And it shows A, 0, 1 and 3 in its output, and also down counter has initialized to count BCD numbers. After that it will print out all 13 bytes data from data processor.





In this case, it successfully print out all 13 bytes from data processor, as dataReady and start signals have 13 edges in S3 state.