COMUNICATIA DINTRE CALCULATOR SI O PLACA DE DEZVOLTARE FPGA PRIN PORTUL SERIAL

Calculatoare si Tehn. Informatiei **Prof. Coord: Daniela Fati**

Structura Sistemelor de Calcul **Student: Diaconu Andrei**

Grupa 30237 **Data: 6.1.2022**

CUPRINS

REZUMAT

Una dintre temele studiate de-a lungul anului la aceasta catedra de Sisteme de calcul si structura acestora a fost portul UART – transmiterea seriala. In cadrul acestuia am folosit receptia pe placuta a unui caracter transmis de la calculator.

Am plecat de la baza acestei teme si am construit o comunicare bidirectionala folosind atat Transmiterul cat si Receptorul portului incorport pe placuta noastra.

Asadar acest proiect consta strict in transmiterea unor date de la tastatura unui PC catre FPGA care recepteaza datele , afiseaza codul ascii al datei trimise pe 2 afisoare de 7 segmente pe urma serializeaza data tocmai trimisa de calculator si o trimite inapoi la acesta , urmand sa fie afisata.

Pentru afisarea de care tocmai spuneam am decis sa folosesc unul din cele mai cunoscute emulatoare (Tera Term). Aceasta va face transmiterea si receptia datelor catre respectiv de la placuta. Desigur ca IDE si limbaj am ales Vivado de la Xilinx si VHDL , cele utilizate de a lungul cursurilor si laboratoarelor acestui an. Placuta folosita este Nexys4 DDR.

INTRODUCERE

Un UART este uneori denumit port serial, interfață RS-232 sau port COM. Este una dintre cele mai simple metode de comunicare cu FPGA. Alte metode de comunicare despre care s-ar putea să fi auzit sunt PCI, PCI-Express, USB, etc, dar FPGA folosește un UART pentru că este cel mai ușor și cel mai bine de învățat.

Acest port constă dintr-o componentă receptor și o componentă transmițător. Pentru porțiunea receptor UART, trebuie să luăm datele seriale de pe computer. Acesta este trimis de computer pe rând. Receptorul îl convertește înapoi în octetul original. Aceasta este o conversie a datelor seriale în date paralele. Emițătorul funcționează în sens invers. Trimite câte un octet printr-un singur fir. Un octet are o lățime de 8 biți, așa că pentru a trimite un octet pe un singur fir, trebuie să convertim datele din date paralele (cum sunt stocate în octet) în serie. Aceasta este ceea ce tine de partea transimtorului. Am decis să împart UART în două părți, dupa cum se observa in partea de receptie si de transmisie.

Un UART are mai mulți parametri care pot fi setați. Emițătorul și receptorul trebuie să cadă de acord asupra setărilor de mai jos sau se produce coruperea datelor. Să discutăm despre parametrii UART care sunt setabili:

Baud Rate (9600, 19200, 115200, altele) - dar am ales 115200 Număr de biți de date (7, 8) - dar am ales 8 Bit de paritate (pornit, dezactivat) - dar am ales dezactivat Biți de oprire (0, 1, 2) - dar am ales

1 Controlul fluxului (Niciunul, Activat, Hardware) - dar am ales niciunul

Baud Rate este viteza la care sunt transmise datele seriale. 115200 Baud înseamnă 115200 biți pe secundă. Numărul de biți de date este aproape întotdeauna setat la opt. Un bit de paritate poate fi adăugat după trimiterea datelor, pentru a ști dacă datele au fost primite corect sau nu. Un bit de oprire este întotdeauna setat la 1 și pot exista 0, 1 sau 2 biți de oprire. Controlul fluxului nu este utilizat în mod obișnuit în aplicațiile actuale și probabil va fi setat la Niciunul.

Fundamentare teoretică

In cadrul acestui proiect voi folosi state-uri . Acestea sunt utile pentru a urmări o secvență de pași în interiorul FPGA.

In cazul nostru starile vor fi Idle, Rx_StartBit, RX_DataBits, Rx_StopBit, S_Cleanup , fiecare dintre ele cu o misiune anume. Una depinde de cealalta si una ii determina celeilalte o viitoare folosire in functie de semnalele ajutatoare cum ar fi Data Validation, Serial si Byte Rx, etc.

In linii mari procedeul implementat este urmatorul :

Receptorul caută mai întâi marginea de cădere a bitului de pornire. Aceasta indică faptul că se transmite un octet. Apoi așteaptă o jumătate de perioadă de biți pentru a se alinia la centrul datelor UART. Motivul pentru a folosi centrul biților este că atunci este cel mai puțin probabil să vedem tranziții și cel mai probabil să obținem o mostră bună de date UART.Odată ce receptorul este aliniat la centru, eșantionează datele de pe linie și stochează fiecare bit într-un registru de octeți.

Incrementează un index pentru a urmări ce bit este primit. Odată ce toți cei 8 biți de date sunt primiți, acesta primește un bit de oprire, apoi revine la starea IDLE pentru a aștepta următorul octet de date.

Idle este stateul in care totul sta pe loc, daca niciun alt state nu se identifica programul ajunge aici, tot ce tine de cod se initializeaza, de la Clk_counter pana la datele ce urmeaza a fi trimise de calculator si recepitonate de placuta FPGA.

Iesirea din aceasta stare e determinata de RX_Serial care va fi HIGH level in mod normal , insa in cazul in care se va face 0 aceasta ne

indica ca a fost detectat un bit de start. Pentru a verifica in urmatorii pasi trecem la stare de StartBit in care intalnim ca o prima conditie Clk counterul si clk per bit. Clk counterul are un nume sugestiv si reprezinta tacturile de ceas efectuate.

Pentru inceput cautam ca acest counter sa fie jumatatea CLK per bitului . Clk per bit reprezinta numarul de clkuri de a lungul unui baudrate . Asadar intr un ciclu de transmitere de bit sunt exact 868 de astfel de clkuri insa noi dorim sa verificam daca la jumatatea acestuia RX_Serial a ramas tot 0. In caz afirmativ cu siguranta PC incepe sa trimita date catre placuta, in caz negativ se revine in stare de Idle.

Ajunsi la partea de Data Bits, aici incepe transmiterea propriu zisa de biti. Un bit a fost complet transmis dupa 868 de clk-uri , asdar trebuie sa numaram de 7 ori pt cei 7 biti aceste clk-uri. Ne ajutam de un Bit_index si ne oprim doar in momentul cand acesta a depaist 7. Moment in care trecem la Stop Bit

In Stop bit asteptam tot 868 de clk-uri insa doar odata, pentru un ultim bit care ne asigura ca datele au fost transmise cu succes, si pot fi afisate pe FPGA. Seteaza datele ca fiind valide si face Cleanupul care reprezinta trecerea inapoi in IDLE si resetarea valoriilor auxiliare modificate pe tot parcurusul elaborarii (receptiei).

Ce tine de state-uri partea a doua despre care am si zis ca este transimotrul este in mare parte la fel , insa are o logica inversa , deoarce data este serializata si trebuie converitata si serializata la loc. Aceasta se face analog bit cu bit , si prezinta aceleasi “simptome” de oprire si detectare a bitilor marginali cat si a transmisei inverse de la placuta la PC , respectiv emulatorul Tera Term.

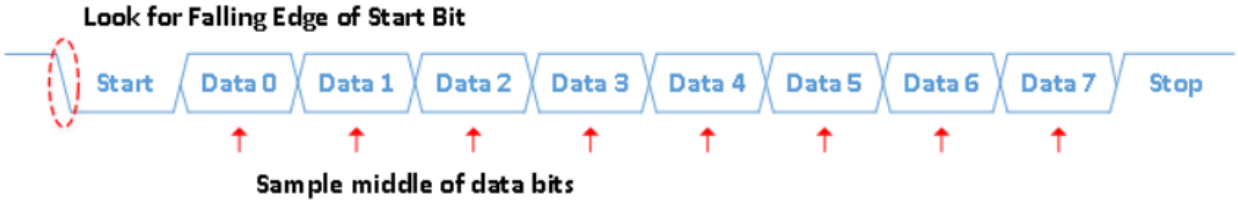
Acest design inainte de a fi pus pe placuta , trebuie simulat sa I vedem comporatmentul, asadar am construit un testbench, pentru o depanare mai usoara a codului in schimbul celui pe hardware unde nu

putem vedea exact ce se intampla in saptele simularii. Aici se vad semnalele la fiecare pas /

Receptorul UART trimite datele primite la ieșirea o_RX_Data. Aceste date trebuie trimise la modulul Binary to 7-Segment pe care l-am creat anterior. Ar trebui să conectăm cei 4 biți inferiori în segmentul inferior și cei 4 biți superiori în segmentul superior.

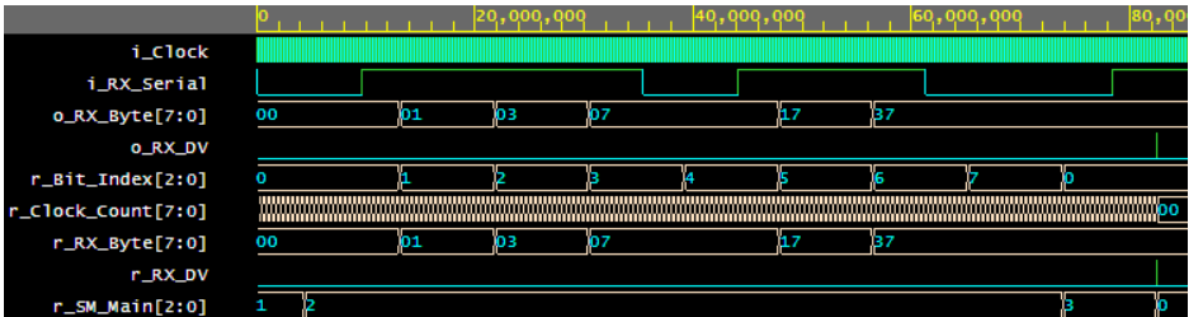
Programul este unul soft dar trebuia legat la aparatul propriu zis care ne arata daca este unul aferent sau nu. Asadar am fost nevoit sa mai fac un program .vhd care contine fix legaturile in placuta

Clk, RX, TX si An-Seg. An , Seg sunt noutatile acestea reperezentand niste vectori logici pe 2 repsectiv 8 biti. 8 biti pt data trimisa sa stie cum sa activeze displayul de 7 segmente in functie de cifra ar trebui in hexa, iar led-ul pentru a detecta cifra (prima sau a doua)



REZULTATE EXPERIMENTALE

Rezultatele in urma modificariilor in special a celor legate de clk counter in care au aparut mici buguri, au fost cele asteptate.



Putem urmari atat indexul de bit care creste conform codului pana la 7 si se opreste cat si valorile transimte pe biti, validarile si bitii de start/stop.

In ceea ce priveste placuta noastra FPGA care este un Nexys4 DDR cu assignari strict pe cele doua afisoare , Clk si datele (RX si TX) am avut o mica supriza realizand dupa cateva incercari ca nu afisa nimic. Acest lucru se datora faptului ca RX trebuia conecatat la input si TX la output. Inversarea assignarii acestora in fisierul de constrangeri xdc , mi –a dat batai de cap insa am reusit sa imi dau seama exact si remediez problema, si rezultatul este unul aferent atat vizual pe FPGA cat si in testbench.

CONCLUZII

Mi-a placut acest proiect pentru ca a fost o continuare, un upgrade la programul propus ca si exercitiu de laborator. Acolo un simplu caracter de afisat mi-a pus multe semne de intrebare care in mare ma bucur ca le am putut pune la punct in proiectul curent.

Referitor la alte extensii se poate afisa pe segmente chiar caracterul trimis, nu codul lui ascii, sau se pot folosi switchurile pentru transimiterea unei data de la placuta la terminalul (TERA), insa raman niste planuri de viitor.

Sunt multumit de ce am reusit sa fac , insa nu am avut o placuta la indemana sa pot testa mai des, sa mi dau seama exact ceea ce se intampla . Realizarea unui testbench a fost salvarea mea in ceea ce priveste acest context.

BIBLIOGRAFIE

https://users.utcluj.ro/~baruch/ssc/proiect/Ghid-Proiect.pdf

https://www.researchgate.net/publication/353931795_UART_Implementation_using_FPGA

http://www.engr.siu.edu/haibo/ece428/notes/ece428_uart.pdf

https://www.fpga4fun.com/SerialInterface.html