SISTEMA LOGIKO PROGRAMAGARRIAK PBL LANA

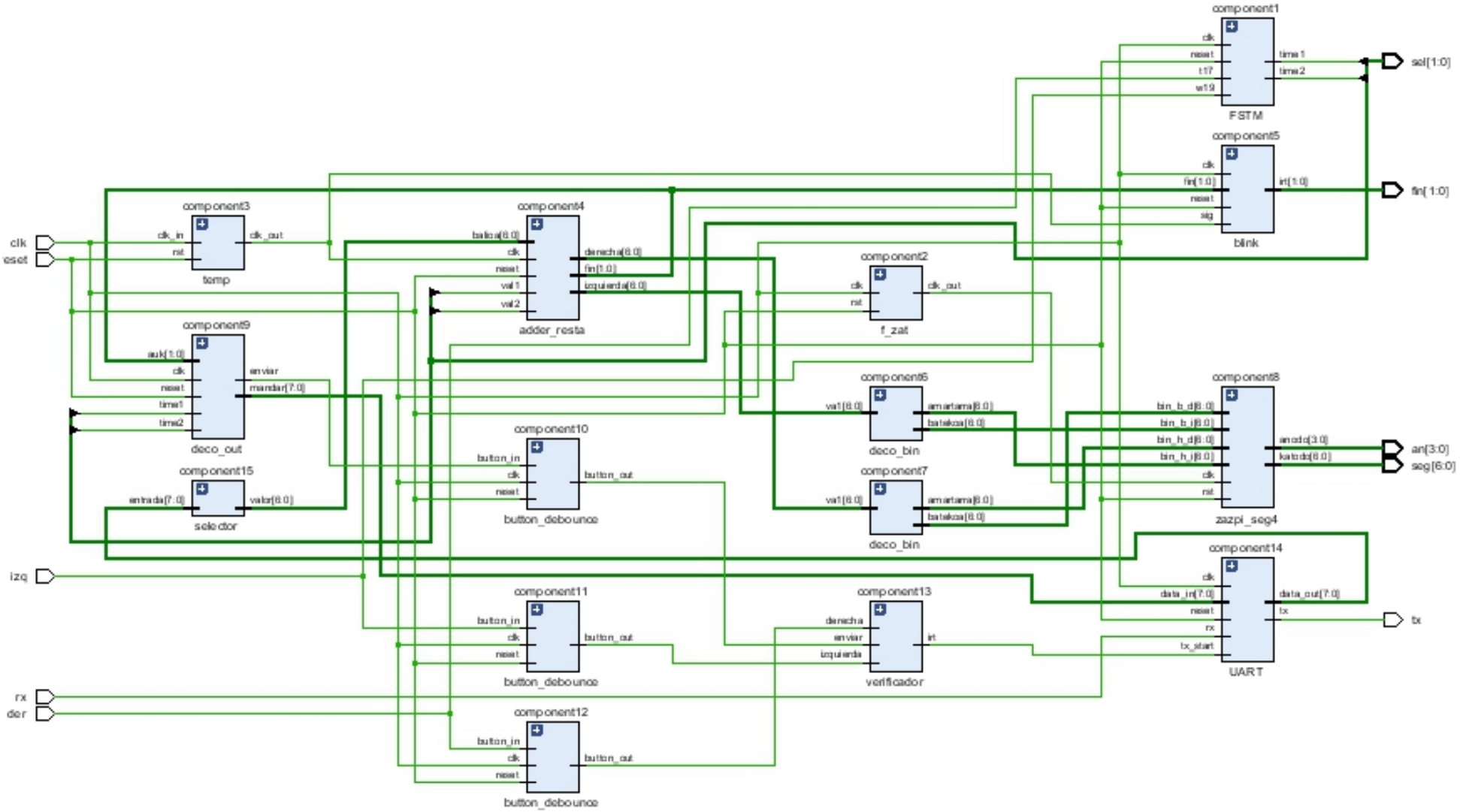
Ingeniaritza informatikoa

Egilea: Mikel Murua

Urtarrilak 3, 2021

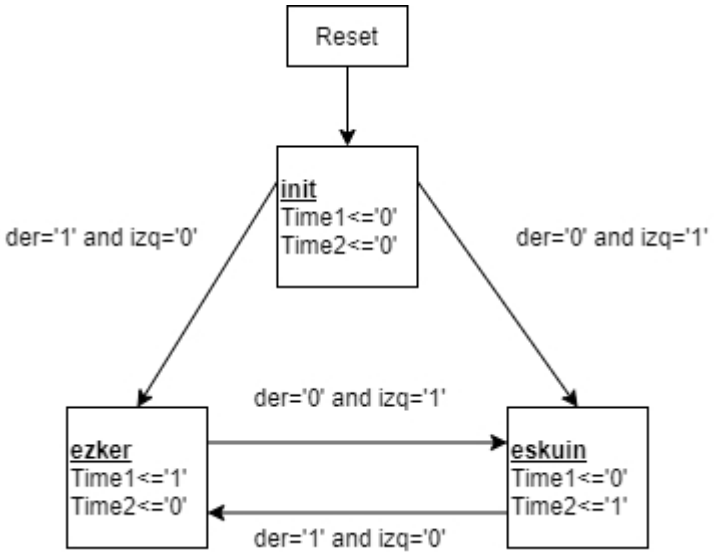
EDUKI TAULA

BLOKE DIAGRAMA



KONPONENTEAK

FSTM



Proiektu honetako egoera makina da, bertan erlojuaren 3 egoerak daude, init esker eta eskuin, initen erlojua geldirik egongo da eskuineko edo ezkerreko botoiak pultsatzen diren arte, gero botoiak behin sakatuta dagokion erlojua aktibatuko da, eta gero bestea sakatzean, aktibatuta zegoena gelditu eta bestea hasiko da kentzen.

TEMP

Frekuentzia zatitzaile normala, bertan plakaren osziladorearen 100.000.000 tik kontatzen dira 1s ren baliokidea den seinale konstante bat bidaltzeko kanpora. Hau oso erabilgarria zaigu adibidez, kenketa egiterako orduan segunduro 1 kentzeko erlojuari, adder _resta konponenentean, edo blink konstante bat egiteko.

Kalkuluak:

$$T = \frac{1}{f} \rightarrow \quad f * X = nahi\ den\ denbora \quad \rightarrow \quad X = 100 * \frac{10^6}{1} = 100.000.000$$

F_ZAT

Beste frekuentzia zatitzaile bat, hau bestearen antzera plakaren osziladorearen frekuentziaren tik ak kontatzen ditu, baina besteak ez bezala, honen seinale konstantea, frekuentzia jakin batera dago, zazpi segmentuko dekodetzaileko zenbakiaren 4 digituak batera ikus daitezen. Anodoaren aldaketa frekuentzia jakin honetara egitean 4 digitu oso ikusten baitira, inungoko distira arrarorik gabe. Efektu hau 100.000 clock tik-ko periodo batean 50000 tik ero. 1 bat bidaltzen lortzen da.

ADDER_RESTA

Konponentearen izenak dionez moduan, honek 2 erlojuen balioen kenketaz arduratzen den elementua da, val1 eta val2 balioen bitartez kontrolatzen dugu kenketa.

- Val1 -> eskerreko erlojua kontrolatzen du (1-aktibatuta 0- itzalita)
- Val2 -> eskumako erlojua kontrolatzen du (1-aktibatuta 0- itzalita)

Horretaz gain konponente honek ere beste 2 funtzio garrantzitsu ere baditu:

- Erloju bietatik bat 0 baliora iristen denean, erlojuen kenketa blokeatu egiten du. Eta hau Reset eginez soilik desblokeatu daiteke. Kasu honetan hasierako balioak hartzen ditu berriro.
- Beste funtzioa , ze erlojuk amaitu duen abisatzea da, horretarako fin seinalea erabiltzen dugu.
    - fin="10" (ezkerrekoa erlojuak amaitu du).
    - fin="01" (eskumako erlojuak amaitu du).

BLINK

Izenak dioen moduan, irteerako led-ak dirdira eginarazten duen konponentea da, adder_resta konponentearen fin seinaleak aktibatzen du, horrela amaitu duen erlojuaren led-ak dir-dir egin dezan. Dirdiraren frekuentzia segundo bakarrekoa da, eta honetarako pasatzen zaion erlojua f_zat konponentearen 1s -ko seinalea da.

DECO_BIN

Konponente honen helburua, erloju baten balioa hartzea eta zazpi segmentuko deskodetzaileak ulertu ditzakeen seinaletan bihurtzea da.

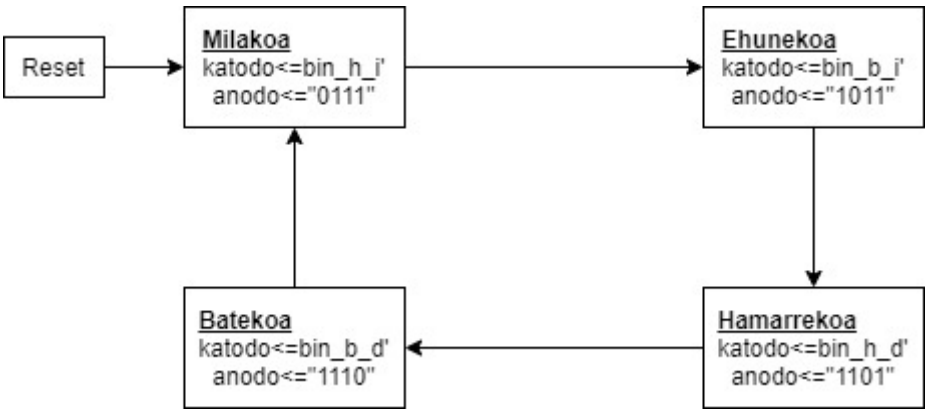
Kontuan hartuz, 7 segmentuko katodo bakoitzak digitu 1 erakutsi dezakeela bakarrik, deco bin honetan zenbakia, bere 2 digituetan banatzen da. Atal hamartarreko digituak eta bateko ataleko digituak.

Erloju balioa	**Hamartarra**	**Batekoa**
74	7	4

Zenbaki hamartarraren balioa hamartarra irteeratik kanporatzen da zazpi segmentuko deskodetzailerantz, eta zenbaki bakoitia batekoa irteeratik kanporatzen da, leku berdinerantz. Horrela zenbaki guztia bidali dezakegu 7 segmentuko displaierantz. Hemen erlojuak eduki ditzakeen 100 egoerak biltzen dira. 0 tik 99 ko zenbakiak alegia. 2 deco bin erabiliko dira , 1 erloju bakoitzaren balioentzat.

ZAZPI_SEG4

Zazpi segmentuko deskodetzaile bat da, baina pixka bat moldatuta dago 4 anodoak batera ikusi daitezen. Lehen komentatu dudan moduan , deskodetzaileren katodo bakoitzak digitu bat soilik erakutsi dezake, eta aldi berean, zenbaki (anodo) bakarra erakutsi daiteke bakarrik. Horregatik 4 zenbakiak batera ikusteko hurrengoko egoera makina erabiltzen da:

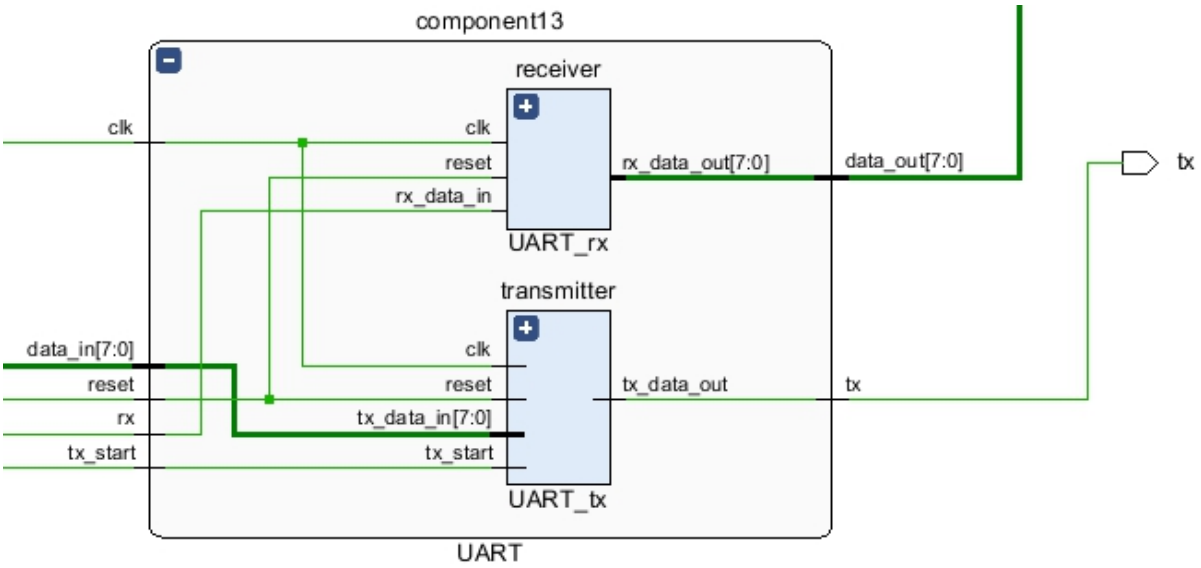


Horretaz gainera, 2 deco_bin konponentetatik 4 balio jasotzen ditu, balio bakoitza katodo balio bat izanez.

- Bin_h_i-> ezkerreko erlojuaren balio hamartarra
- Bin_b_i-> ezkerreko erlojuaren bateko balioa
- Bin_h_d-> eskumako erlojuaren balio hamartarra
- Bin_b_d-> eskumako erlojuaren bateko balioa

Bertan egoera bakoitzak deco_bin konponenteetatik datozen seinaletatik 1 erakusten dute egoera bakoitzean. Hortaz 4 digituak batera ikusteko egoera arteko aldaketa, aurretik azaldutako Temp konponentearen frekuentzia jakin horretan egin behar dugu, gure begiei 4 digituak batera ikustea eragiten dion efektu optikoa gertatzeko.

UART



Konponente honen eginbeharra barnean dituen Uart_tx eta Uart_rx konponenteen erabilpena erraztea da, bere irteera eta sarrera garrantzitsuenak hauek dira:

- Tx_start: Linea seriean idazketa astea markatzen duen seinalea da.
    - 1- linea seriean idazten du
    - 0-Ez du linea seriean idazten
- Data_in: Linea seriean behin tx_start aktibaturik dagoenean, linea serietik bidaliko den edukia da.
- Tx : Linea seriearen idazketaren logika duen irteera da.
- Rx : Linea seriearen irakurtzearen logika duen sarrera da.
- Data_out: Linea serietik irakurtzen den azken karakterea, irteera honetatik pasatzen da. 8 biteko formatuan.

UART_TX

Uart_tx konponentea , UART-aren transmisorea da bere barnean dituen prozesuak hurrengokoak dira:

baud_rate_clk_generator

Plakak ordenagailuaren portu seriearekin komunikatzeko baudioak emen definitzen dira hurrengoko formularen bitartez.

$$baudioak = F\ plaka/(16 * nahi\ den\ frekuentzia)$$

$$baudioak = \frac{100.000.000}{16 * 115.000} = 868$$

tx_start_detector

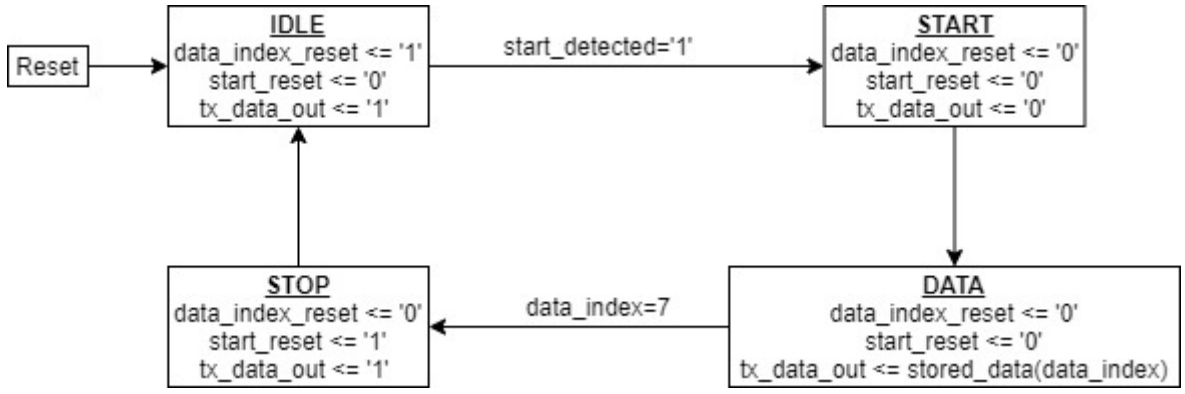
Prozesu honek 2 eginbehar nagusi ditu:

- Tx_start sarreraren aktibazioa, hau da transmisioa asteko seinalea gertatu den edo ez begiratzen du. Gertatu bada start_detected=’1’ era jarriko da, transmisioa hasiz.
- Bidali beharreko informazioa, data_in sarreratik iristen dena, store_data seinalean gordetzen du, transmisioaren datuak galdu ez daitezen

Data_index_counter

Kontadore sinple bat da 0 tik 7 ra mugitzen dena, gehiketa baudioen seinalea iristen zaionean egiten da. Bere eginbeharra konponentean , store_data 7 bit eko bektorearen bitak banan banan tx_data_out irteerara pasatzea da baudioek markatutako frekuentzian, horrela datuak linea serietik bidaltzeko.

UART_tx_FSM (finite state machine)



Ikusten den moduan, egoera makina, IDLE egoeran hasten da, hemen hasiera seinalea jaso arte itxaroten du START egoerara pasatzeko.

Seinalea jasotakoan DATA egoerara pasatzen da bertan Data_index_counter prozesua erabiliz, stored_data seinaleko edukia linea seriera bidaltzen du. Data_index, stored_data seinalearen bektorearen bitak kontrolatzen duen markadorea, bektorearen amaierara iristen denean. Hau da stored_data 7 bit dituenez, 7.garren posiziora iristen denean. Bidali beharreko eduki guztia bidali dela kontsideratuko da, STOP egoerara helduz.

STOP egoeran komunikazioaren amaierako bita jarriko zaio transmisio edukiari eta transmisioa amaitzean, seinale guztiak berrezarriko ditu, berriro hasierako seinalea itxoiten egon dadin IDLE egoerara bueltatuz.

UART_RX

UART_rx linea serietik informazioa irakurtzen duen konponentea da, bere barnean dituen prozesuak hurrengokoa da:

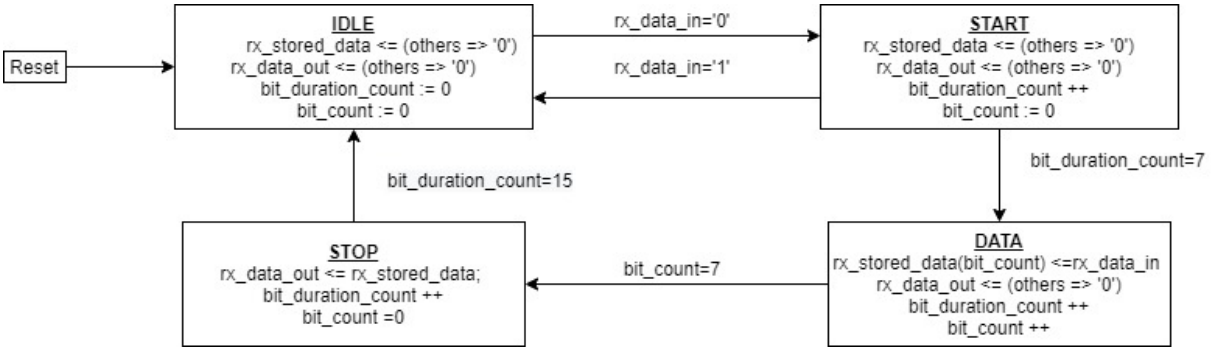
baud_rate_x16_clk_generator

Plakak ordenagailuaren portu seriearekin komunikatzeko baudioak emen definitzen dira hurrengoko formularen bitartez. Haurreko UART_tx elementuan erabilitako formularen desberdina da.

$$baudioak = ((\frac{\mathrm{F_{plaka}}}{\text{nahi den frekuenzia}}) / 16)$$

$$baudioak = (100\,000\,000 / 115\,200) / 16 = 54.25$$

UART_rx_FSM



Reset egitean IDLE egoeran hasten gara, emen kontadoreak eta datuak gordetzeko aldagaiak erreseatetzen ditugu eta linea seriean zerbait idatzi den konfirmatzen duen bita itxoiten gelditzen gara. Hau jasotzean START egoerara pasatzen gara.

START egoeran , idazketa benetan egin den konprobatzen da, hasierako bitak duen seinalea denbora batez mantentzen dela konprobatuz, neurketa hau bit_duration_count seinaleak darama. 7 tik-ez mantentzen bada, DATA egoerara pasatzen da, ez bada mantentzen alarma faltsu moduan kontsideratzen da eta IDLE egoerara itzultzen da.

DATA egoeran rx_data_in sarrerako edukia bitez bit irakurtzen da eta rx_stored_data barruan gorde, bit bakoitza irakurtzen denean bit_count 1 ez gehitzen da, horretaz gainera, denbora kontatzen jarraitzen dugu. rx_data_in en 7 bitak irakurtzen STOP egoerara pasatzen gara.

STOP egoeran, bit_duration_count denbora kontagailua begiratzen dugu, 15 en desberdina bada, rx_data_out en gordetako edukia ez duela balio esan nahi dut IDLE egoera ra bueltatzen da, irteerara bidali baino lehen edukia ezabatuz. Aldiz denbora kontagailua 15 bada, rx_stored_data seinalean gordetako edukia rx_data_out irteerara pasatzen da, irakurritako edukia erabilgarri utziz,8 biteko formatuan top Design-eko beste konponente guztientzat.

DECO_OUT

Konponente honek , Uart konponenteari, erlojuaren egoera bakoitzean linea serietik idatzi behar duen seinalea pasatzen dio, mandar irteeraren bitartez.

Horretaz gainera botoia sakatzean edo botoia sakatu gabe bidali behar duen ere zehazten du enviar irteeraren bidez. Adibidez amaierako seinaleak, botoiak sakatzean agertzen dira, horretaz gainera dagokion amaieraren karakterea gelditu gabe bidaliko du, reset ematen zaion arte.

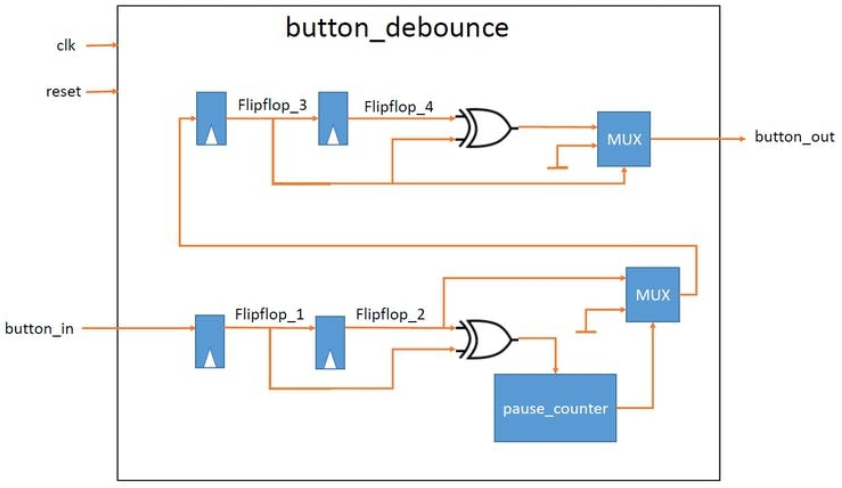
Erlojuaren egoera	Uart-eko data_in sarrerara bidaltzen duena
Eskerreko erlojua aktibatu	i
Eskumako erlojua aktibatu	d
Ezkerreko erlojua amaitu	g
Eskumako erlojua amaitu	h

BUTTON_DEBOUNCE

Botoien debouncerra da, hau da botoiak sakatzean, seinalea behin bakarrik errepikatzen dela ziurtatzen duen elementua da.

Bere funtzionamentuaren inguruan,diagraman ikusten den moduan , flipflop ugari erabiltzen dira button_in sarreratik, botoi/deco_out konponeneteek sortuta datorren seinale zaparrada mozteko, hau segurtasunez egiteko, seinalea flipflop_1 ean gordetzen da, gero flipflop_2 seinalean gordez, honek pause_counter procesua aktibatzen du 1 era jarriz, gero Mux procesuan, kontadore txiki bat erabiltzen da 10.000 tik eko delaia sortzeko, horrela flipflop_2 ko balioa flipflop_3 ra pasatzean delai hau sortzen bada, eta botton_in seinalea flipflop-aren berdina bada pause_counter ez da aktibatuko kanpotik datorren seinalea moztuz.

Gero honen ostean flipflop_3 eta 4 atalean hartutako seinalea modu laburrean 1 bakarrik bidaltzeko procesu berdina erabiltzen da.



horrelako 1 dago botoi bakoitzagatik. Nire kasuan 3 ditut, bat eskumako botoiarentzat eta beste bat ezkerreko botoiarentzat. Horretaz gainera deco_out konponentetik ateratzen den seinalea behin bakarrik bidaltzeko ere balio digunez , hor ere erabiltzen dut.

VERIFICADOR

Datuak bidaltzeko seinalea dagoen kontrolatzen duen konponentea da, or logiko bat eginez botoietatik datozen seinaleei, eta beste or logiko bat, deco_out etik datorren seinalea konprobatzeko ere balio digu. Alegia konponente honek idazketa aktibatzen duten egoeren seinaleak detektatzeko balio digu, eta seinale detektatzen duenean aktibazioko seinalea bidaliz UART konponentearen tx_start sarrerari, horrela idazketa asteko.

SELECTOR

Emen Deco_out konponentearen berdina egiten dugu, baina Uart konponenteari , linea seriean zer idatzi behar duen ordez. Linea seriean idatzi dugun karakterea jaso egiten du, eta balio hori kontuan hartuta, erlojuari hasierako balio bat edo beste bat pasatzen dio , valor irteeraren bitartez. Balio hau reset egitean aplikatzen zaio erlojuari.

Definitutako balioak hurrengokoak dira:

Linea serieko balioa	**Erlojuari pasatzen zaion hasiera balioa**
a	33
b	99

BIBLIOGRAFIA

- https://reference.digilentinc.com/basys3/refmanual?_ga=2.145691956.1878583647.1608578818-1695917639.1608578818
- https://reference.digilentinc.com/reference/pmod/pmodrs232/reference-manual
- https://www.youtube.com/watch?v=sTHckUyxwp8
- https://vhdl.es/puerto-serie-en-vhdl/
- https://www.youtube.com/watch?v=LjVHW_GB4X0
- https://core-electronics.com.au/tutorials/serial-monitoring-with-tera-term.html
- https://github.com/Digilent/Basys-3-GPIO/blob/master/src/hdl/UART_TX_CTRL.vhd?_ga=2.187126056.1878583647.1608578818-1695917639.1608578818
- https://www.nandland.com/vhdl/modules/module-uart-serial-port-rs232.html
- https://www.hackster.io/alexey-sudbin/uart-interface-in-vhdl-for-basys3-board-eef170#comments