UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Rok Štefanič

ARHITEKTURA IN RAZVOJ NADZORNEGA SISTEMA ZA SINHRONIZACIJO NAPRAV POSPEŠEVALNIKA DELCEV SNS

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentorica: doc. dr. Mira Trebar

Ljubljana 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00463/2009
Datum: 01.09.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ROK ŠTEFANIČ**

Naslov: **ARHITEKTURA IN RAZVOJ NADZORNEGA SISTEMA ZA SINHRONIZACIJO NAPRAV POSPEŠEVALNIKA DELCEV SNS**

ARCHITECTURE AND DEVELOPMENT OF SYNCHRONIZATION CONTROL SYSTEM FOR SNS PARTICLE ACCELERATOR

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Kandidat naj v diplomskem delu predstavi zahteve časovnega krmilnika za sinhronizacijo naprav pospeševalnika delcev in poda obstoječo kompleksno rešitev z uporabo modulov VME. Opiše naj arhitekturno zasnovo, razvoj in izboljšave časovnega krmilnika, ki je implementiran v Virtex5 FPGA vezju ter opiše njegove funkcionalnosti po posameznih modulih. Predstavi naj implementacijo krmilnika v realnem pospeševalniku delcev SNS, njegove prednosti in morebitne možnosti za nadaljnje izboljšave.

Mentor:
doc. dr. Mira Trebar

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Franc Solina

Besedilo je oblikovano z urejevalnikom Microsoft Word

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Rok Štefanič,

z vpisno številko 63020156,

sem avtor diplomskega dela z naslovom:

Arhitektura in razvoj nadzornega sistema za sinhronizacijo naprav pospeševalnika delcev SNS

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Mire Trebar

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 8.12.2009 Podpis avtorja:

Zahvala

Zahvalil bi se mentorici doc. dr. Miri Trebar za potrpežljivost in sodelovanje pri izdelavi tega dela. Posebna zahvala gre celotnemu podjetju Cosylab d.d., še posebno pa dr. Jožetu Dediču, za ves trud, ki je bil vložen vame, vse znanje, ki sem ga pridobil in možnost, da sem lahko sodeloval na tem projektu.

Kazalo

Seznam uporabljenih kratic

BNL – Brookhaven National Laboratory

CDR – Clock and Data Recovery

CPU - Central Processing Unit

CRC – Cyclic Redundancy Check

DC – Direct Current

DCM – Digital Clock Manager

DDS – Direct Digital Synthesis

DPRAM – Dual Port RAM

EL – Event Link

FIFO – First In First Out

FPGA – Field Programmable Gate Array

FSK – Frequency Shift Keying

FSM – Finite State Machine

GPIO – General Purpose Input/Output

GPS – Global Positioning System

IOC – Input Output Controller

ISR – Interrupt Service Routine

LEBT – Low Energy Beam Transport

LUT – Lookup Table

MACC – Multiply and Accumulate

MEBT – Medium Energy Beam Transport

MPS – Machine Protection System

MSB – Most Significant Bit

ODDR – Output Double Data Rate

ORNL – Oak Ridge National Laboratory

PLL – Phase Locked Loop

PTP - Precision Time Protocol

RF – Revolution Frequency

RTDL – Real Time Data Link

SNS – Spallation Neutron Source

VME – Versa Module Europe

VITA – VMEbus International Trade Association

Povzetek

Namen dela je podrobno predstaviti zasnovo in izdelavo časovnega krmilnika za inštitucijo ORNL (ang. Oak ridge National Laboratory), kjer deluje pospeševalnik delcev SNS (ang. Spallation Neutron Source), ki je trenutno najmočnejši visokoenergijski izvor nevtronov na svetu. Razložena je vloga časovnega krmilnika kot ključne naprave, ki omogoča delovanje tako velikega sistema. Predstavljena je obstoječa implementacija naprave z opisom funkcionalnosti in tudi pomanjkljivosti, ki so bile odpravljene z načrtovanjem in izdelavo nove, naprednejše in sodobne digitalne naprave.

Celotna naprava je zasnovana z uporabo FPGA tehnologij, ki ustreza tej tematiki in podaja vrsto opisov možnih rešitev. Prikazani so izseki VHDL programske kode, ki podajajo funcionalnost nekaterih ključnih delov naprave. Opisani pa so tudi izzivi s katerimi sem bil soočen pri sami implementaciji, kot tudi posamezne poglavitne rešitve.

Podani so rezultati, kjer je podrobno predstavljena uspešna realizacija naprave, kot tudi njena uspešna namestitev in uporaba na samem pospeševalniku. Opisane so tudi nove možnosti, ki so se porodile po uspešno zaključenem projektu.

Ključne besede: časovni krmilnik, pospeševalnik, izvor nevtronov, FPGA, VHDL

Abstract

This thesis presents the design and implementation of the timing master device for the SNS (Spallation Neutron Source) particle accelerator located at the ORNL (Oak Ridge National Laboratory), which is currently the most powerful neutron source in the world.

It is described why timing master device is a crucial device for the accelerator operation. Given is the functionality and implementation of the currently installed and used device, together with the descriptions of all its flaws which were fixed by designing and implementing a new and modern device.

New timing master device is based entirely on the FPGA technology, where common principles and solutions used are provided in details. Trough the work, portions of the actual VHDL code used on the project describe the functionality of some crucial parts of the implementation. Challenges that were met trough the course of implementation are described as well as their solutions.

Final results describe the status of the project, where the device was successfully implemented and installed at the particle accelerator. Also provided are the new possibilities for additional work that emerged because of the successfully finished project.

Keywords: timing master, particle accelerator, nutron source, FPGA, VHDL

1. Uvod

Zasnova časovnega krmilnika pospeševalnika delcev SNS [**1**] je tesno povezana z vrsto in načinom pridobivanja žarka (toka delcev). Za razumevanje vloge krmilnika je potrebno razumeti celotno pot, ki jo opravi žarek od njegovega nastanka, do njegove končne uporabe. Žarek na svoji poti preide skozi večje število naprav, ki morajo biti popolnoma časovno usklajene. Svojo nalogo morajo opravljati ob točno določenih trenutkih, za kar potrebujejo ustrezno časovno referenco, ki zagotavlja pravilno delovanje.

Naloga časovnega krmilnika je sinhronizirati sprejemnike, ki krmilijo električno-mehanske naprave, ki so uporabljene v procesu pridobivanja in obdelave žarka, hkrati pa mora sprejemnikom zagotoviti informacijo o točnem času ter dogodkih, ki sprejemnikom narekujejo kdaj in kakšne vrste akcij morajo izvesti. Krmilnik je ključna naprava, ki omogoča kontrolnemu sistemu pospeševalnika dejansko dosegati željeno funkcionalnost pospeševalnika.

Nov časovni krmilnik pospeševalnika SNS je implementiran v Virtex5 FPGA [**2**] vezju in za razliko od zastarele rešitve katero je nadomestil, zajema vso funkcionalnost naprave z enim samim integriranim vezjem. Krmilnik je načrtovan kot modul sistema VME [**5**] v katerega je tudi nameščen. Velik del funkcionalnosti, ki je bila do sedaj realizirana programsko, je implementirana strojno, kar je zelo razbremenilo kontrolni sistem in napravi dodalo večjo zanesljivost in robustnost. S krmilnikom upravlja kontrolni sistem, ki se izvaja pod operacijskim sistemom v realnem času vxWorks [**3**] in zagotavlja potrebno krmiljenje.

Diplomsko delo poda opis delovanja pospeševalnika, njegove posebnosti in zahteve, ki jih mora izpolnjevati razvit časovni krmilnik. Predstavljena je zasnova in implementacija časovnega krmilnika, ki je izboljšsana nadgradnja obstoječe rešitve. Opisane so tudi metode testiranja, ki so bile uporabljene, kot tudi opisi težav, ki so se pojavili med razvojem. Delo je potekalo v sklopu obsežnega projekta v podjetju Cosylab d.d., ki je vodilno podjetje na področju razvoja strojne opreme in integracije ter izgradnje kontrolnih sistemov za fizikalne pospeševalnike.

2. Časovni krmilnik v pospeševalniku delcev

2.1 Opis projekta

Nadgradnja časovnega krmilnika

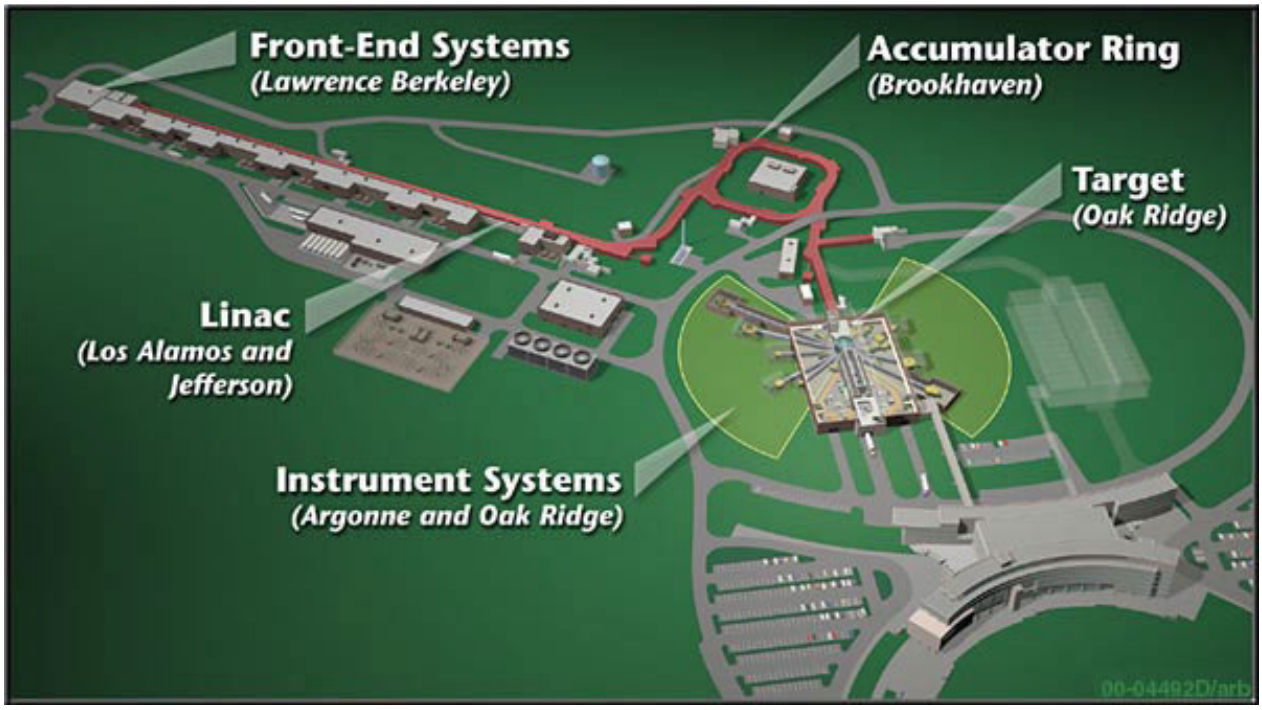
Nov časovni krmilnik je nadgradnja obstoječe naprave, ki nadomešča zastareli časovni krmilnik zasnovanega in izdelanega v laboratoriju BNL [**4**]. Zasnova obstoječega krmilnika se je v celoti opirala na VME sistem, kjer so bile vse poglavitne komponente krmilnika implementirane z uporabo velikega števila VME številnih kartic ter kompleksno in obširno programsko podporo, ki je omogočala delovanje strojnega dela naprave.

Razvita naprava temelji na FPGA integriranem vezju Xilinx Virtex5, v katerem je implementirana celotna funkcionalnost obstoječega časovnega krmilnika, kakor tudi dodatne funkcionalnosti. V celoti je zasnovana tako, da je čim bolj neodvisna od programske podpore, ki omogoča le nujno potrebno konfiguracijo in posodobitev parametrov. Tako naprava neodvisno od programske opreme vedno zagotavlja vsaj osnovno oziroma minimalno funkcionalnost, ki je potrebna da pospeševalnik pravilno obratuje.

Cilj nadgradnje je bil izdelati novo napravo, ki je bolj robustna, učinkovita in popolnoma kompatibilna z obstoječo infrastrukturo, saj mora transparentno komunicirati z vsemi obstoječimi sprejemniki brez kakršnihkoli dodatkov ali sprememb.

Opis delovanja pospeševalnika SNS

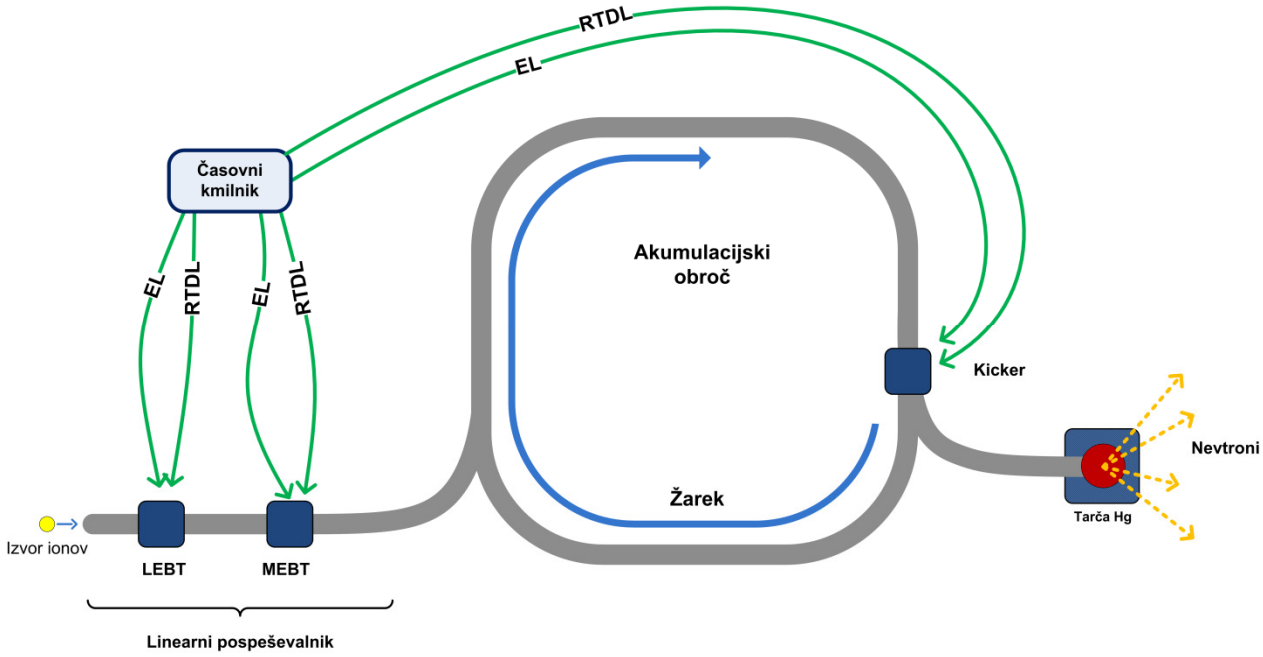
Pospeševalnik delcev SNS (slika 2.1) je sestavljen iz petih večjih delov: izvor ionov, linearni pospeševalnik, akumulacijski obroč, tarča in sistemi z inštrumenti/detektorji.
Delovanje pospeševalnika prikazuje slika 2.2. V začetni stopnji pospeševalnika se nahaja izvor negativno nabitih vodikovih ionov, ki so z linearnim pospeševalnikom pospešeni do visokih energij – hitrosti.



*Slika 2.1 Pospeševalnik SNS [**1**].*

Ioni nadaljujejo pot skozi kovinsko folijo, ki vsakemu ionu odstrani dva elektrona s čimer postane proton. Konstanten tok protonov je nato preko elektromagnetnih in elektrostatičnih razsekovalnikov LEBT (ang. Low Energy Beam Transport) in MEBT (ang. Medium Energy Beam Transport) pretvorjen v žarkovne pulze, ki so vodeni v akumulacijski obroč pospeševalnika, kjer dopolnjujejo že obstoječi žarek. Namen dopolnjevanja žarka je tisočkrat povečati energijo prvotnega žarka. Krmiljenje vstavljanja pulza delcev v akumulacijski obroč mora biti izvedeno z resolucijo 15 ns, saj mora vsak nov pulz ionov natančno pokriti že obstoječ žarek v obroču (žarek zaseda le 2/3 obroča). Takšno dopolnjevanje se odvije 1060-krat in skupaj

traja približno 1 ms. Delovanje pospeševalnika zahteva, da sta naenkrat lahko zasedeni le 2/3 akumulacijskega obroča [**6**], preostala 1/3 obroča pa mora biti prosta. Ta pogoj določa "kicker" magnet, ki vsakih 16 ms (ali 1060 obhodov – dopolnitev žarka) uperi žarek z visoko energijo iz akumulacijskega obroča v živosrebrno tarčo (izprazne akumulacijski obroč), kjer vsak proton v žarku izbije 20-30 nevtronov. Tako pridobljeni tok nevtronov – z razsipanjem (ang. spallation), je končno moderiran in oblikovan, ter nato uporabljen v znanstvenih eksperimentih.



Slika 2.2 Diagram pospeševalnika SNS.

Vsak cikel pospeševalnika ustreza polnjenju akumulacijskega obroča z 1000 pulzi protonov in njegovi izpraznitvi. EL (ang. Event Link) dogodki sinhronizirajo delovanje pospeševalnika in so pridobljeni iz EL tabele, ki jo naslavlja števec obhodov pospeševalnika. Njihovo pošiljanje je sinhrono z delovanjem akumulacijskega obroča. Vsak dogodek je poslan na začetku obhoda akumulacijskega obroča s katerim je orkestrirano celotno delovanje pospeševalnika.

Vsakih 600 ciklov polnjenja in praznjenja akumulacijskega obroča je upoštevano kot super cikel, kjer je razpošiljanje dogodkov pogojeno z maskiranimi vrednostmi v SuperCycle tabeli, ki določa vzorec ponavljanja dogodkov in se lahko med posameznimi super cikli razlikuje. Namen takega

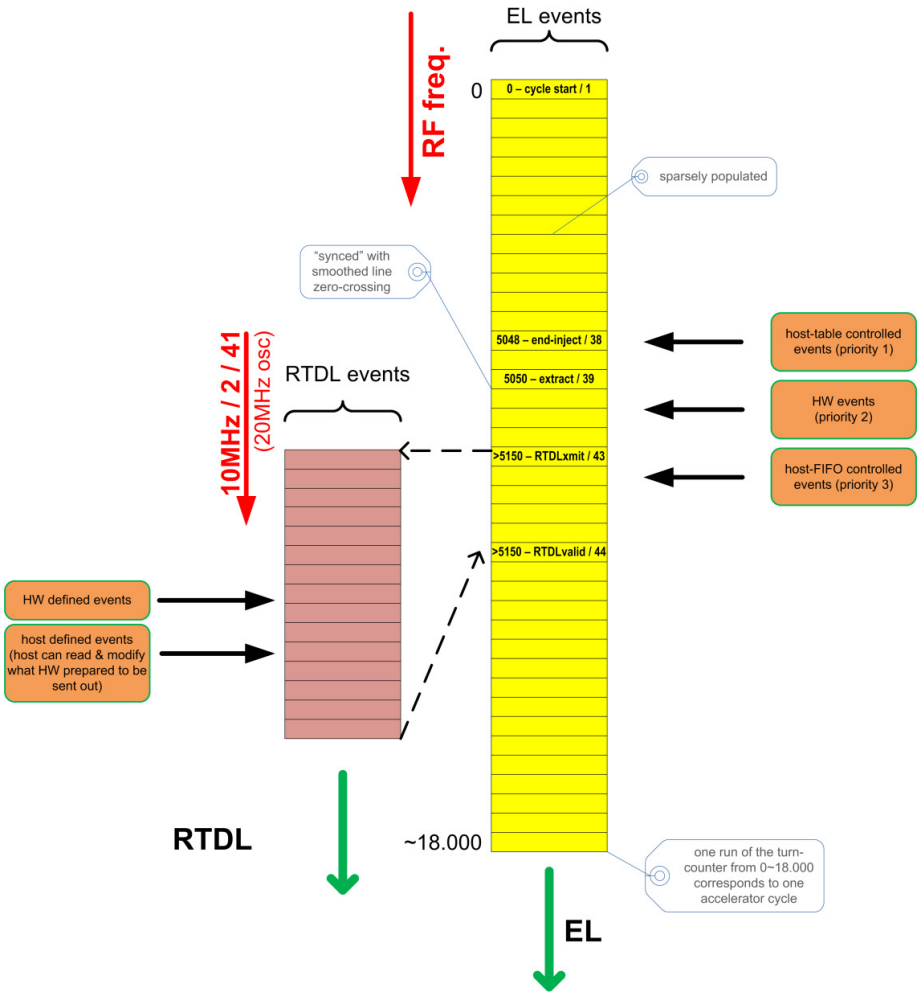
delovanja je generirati super cikle pospeševalnika, kjer ima žarek v vsakem ciklu drugačno energijo, kar omogoča izvajanje različnih fizikalnih eksperimentov.

Vse informacije poslane po dogodkovni in podatkovni povezavi sprejemajo časovni sprejemniki. To so preproste naprave, ki imajo nalogo, da dekodirajo tok podatkov iz povezav in na njihovi osnovi krmilijo ključne naprave, ki upravljajo s fizičnim krmiljenjem pospeševalnika.

2.1.1 Delitev časovnih krmilnikov

Tabelno vodeni časovni krmilniki

Časovni krmilnik SNS deluje na podlagi dogodkovne tabele (EL tabela), zatorej je tabelno voden sistem v realnem času.



Slika 2.3 Tabelno voden sistem.

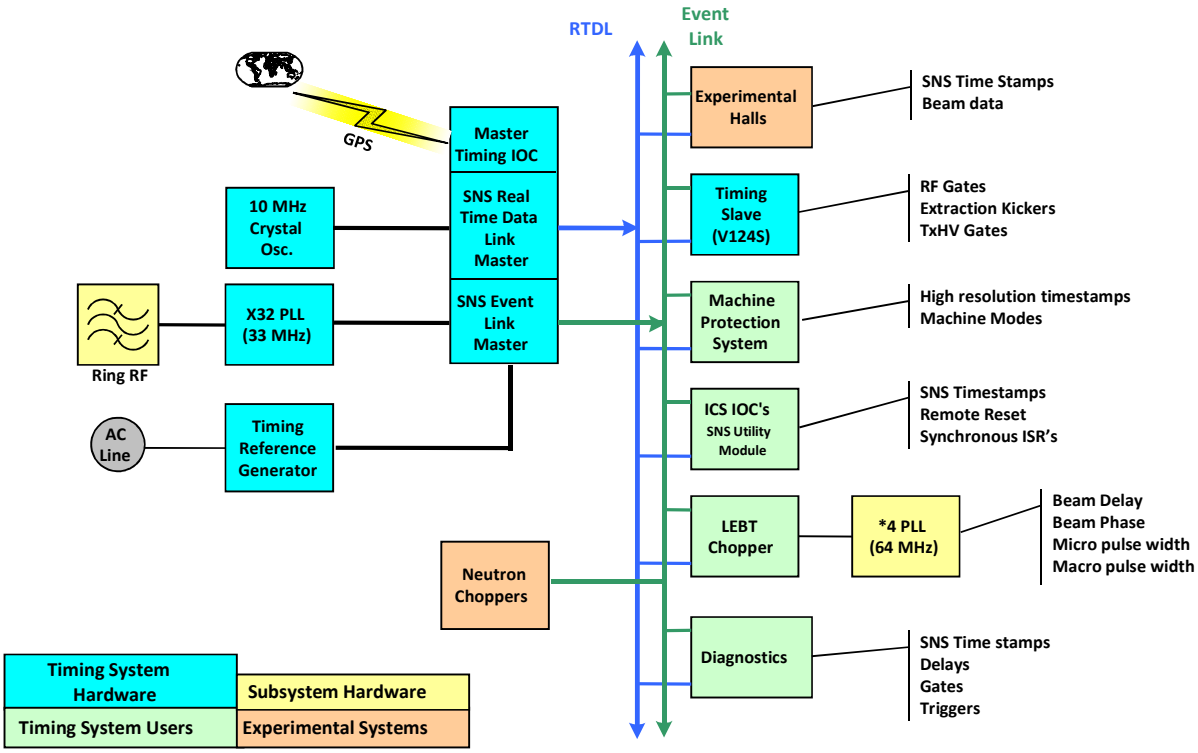
Slika 2.3 prikazuje zasnovo tabelno vodenega sistema. Vse informacije so vsebovane v EL tabeli, ki določa vrstni red dogodkov in s tem delovanje celotnega pospeševalnika. Dogodki iz tabele so pridobljeni z naslavljanjem tabele, kjer je indeks predstavljen s števcem obhodov delcev v akumulacijskem obroču. Poleg same tabele dogodkov, sta na voljo tudi dva stranska izvora dogodkov v obliki FIFO pomilniških elementov. V le te so injecirani dogodki, ki se generirajo dinamično med samim delovanjem časovnega krmilnika. Med izborom dogodka, ki bo v danem trenutku dejansko razposlan, pa so upoštevene prioritetne uteži posameznih izvorov. Pri opisani zasnovi delovanja tabelnega razpošiljanja dogodkov je potrebno pred samim razpošiljanjem dogodka izvesti tudi maskiranje in preverjanje veljavnosti trenutnega dogodka in ali je pošiljanje dovoljeno.

Krmilniki s časovnim najavljanjem dogodkov

Drugi izmed najbolj pogosto uporabljenih načinov časovnega krmiljenja je vnaprejšnje najavljanje dogodkov z vnaprej znanimi točno določenimi časi obravnavanja ali izvršitve. Pri takem krmiljenju je pogoj, da so vsi časovni sprejemniki usklajeni z glavnim časovnim krmilnikom – sprejemnikova lokalna ura v vsakem trenutku kaže enak čas, kot ura glavnega časovnega krmilnika. Za doseganje popolne časovne sinhronizacije je na voljo vrsta metod, najodmevnejša med njimi pa je uporaba protokola PTP [**7**] (ang. Precision Time Protocol), ki določa precizno časovno sinhronizacijo porazdeljenega sistema.

Naloga glavnega časovnega krmilnika je razpošiljanje posameznih dogodkov hkrati z njihovo pripadajočo časovno oznako (ang. timestamp), ki določa čas ob katerem naj sprejemniki reagirajo na določen dogodek. Pri takšnem načinu delovanja, so vsi sprejemniki vnaprej seznanjeni z dogodki in na njih odreagirajo, ko njihova lokalna ura doseže vrednost, ki je navedena v časovni oznaki vnaprej poslanega dogodka.

2.1.2 Opis obstoječega časovnega krmilnika



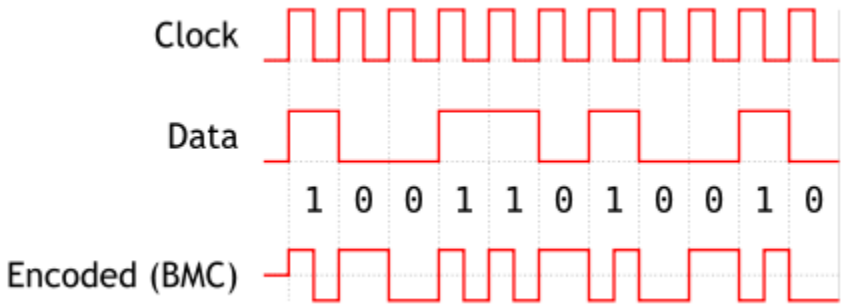
Slika 2.4 Zasnova obstoječe naprave.

Časovni krmilnik na sliki 2.4 sestoji iz dveh serijskih diferencialnih povezav, ki sta vodeni do vseh pospeševalniških sistemov – sprejemnikov:

Po EL dogodkovni povezavi se razpošiljajo 8 bitni dogodki (enumeracije), katerih zaporedje določa pospeševalniški cikel (ang. Machine Cycle). Sinhronizirani so z delovanjem akumulacijskega obroča – RF frekvenco (ang. Revolution Frequency). Tako je z uporabo 256 možnih dogodkov popolnoma opisana celotna funkcionalnost pospeševalnika.

Poleg razpošiljanja dogodkov, pa EL povezava hkrati omogoča tudi sinhroniziranje sprejemnikov glede na krmilnik, s čimer je doseženo, da se akcije na vseh sprejemnikih izvedejo sinhrono. Takšna funkcionalnost je dosežena s kodiranjem 8-bitnih vrednosti dogodkov v 32-bitne vrednosti, z uporabo bi-faznega kodiranja (slika 2.5), ki omogoča, da je poleg informacije v

signalu vsebovan tudi urin signal, ki ga uporabljajo sprejemniki. Bi-fazno kodiranje zagotavlja, da signal na EL povezavi opravi vsaj en prehod iz visokega v nizko stanje ali obratno, med vsakim podatkovnim bitom zaradi česar je možno izvajati obnavljanje urinega signala s CDR [**8**] vezji (ang. Clock Data Recovery). Taka vrsta kodiranja je oblika prenosa informacije s spreminjanjem frekvence signala [**9**] (ang. Frequency Shift Keying), kjer je frekvenca signala ene logične vrednosti dvakrat večja od frekvence signala druge logične vrednosti.

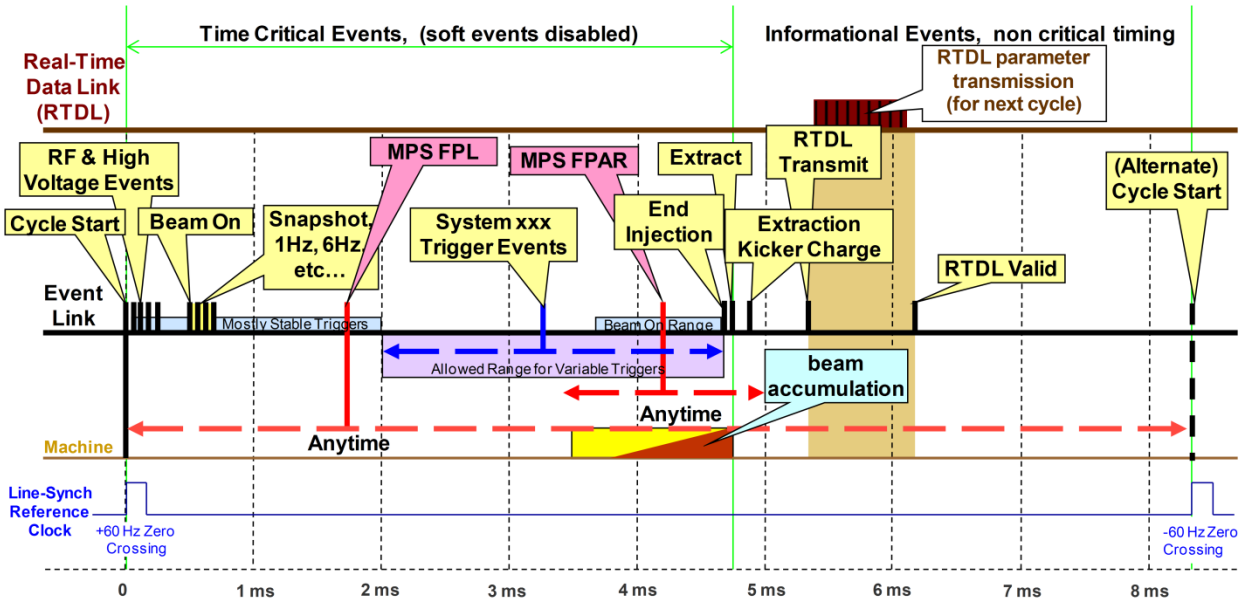


Slika 2.5 Bi-fazno kodiranje.

Za obnavljanje urinega signala pri uporabi takega kodiranja ni potrebno spremljati samih logičnih nivojev signala, temveč je vsa potrebna informacija o urinem signalu pridobljena s spremljanjem prehodov signala – obeh front. Poleg tega pa ima tak signal tudi ničelno DC (ang. Direct Current) napetostno povprečje kar povzroča minimalni elektromagnetni šum na liniji.

Sprejemniki morajo imeti za nemoteno delovanje ves čas na voljo urin signal, zaradi česar mora časovni krmilnik po EL povezavi neprestano pošiljati neko informacijo. Čeprav v delovanju nastopijo obdobja, ko krmilnik ne pošilja nobenih dogodkov mora v takem primeru vseeno pošiljati neko privzeto vrednost, ki ne označuje nobenega dogodka. Običajno je taka vrednost 255, ki zagotavlja izhodni signala z največjo frekvenco spreminjanja – vrednost vsebuje same logične enice, kjer je v kodirani vrednosti vsaka enica predstavljena z enim prehodom signala iz visokega v nizko stanja ali obratno. Tako lahko sprejemniki neprestano izvajajo obnavljanje urinega signala, hkrati pa pridobijo tudi vrednost 255, ki pa je označena kot privzeta in jo kot vrednost dogodka zavržejo.

RTDL [**10**] (ang. Real Time Data Link) podatkovna povezava oddaja niz 24-bitnih podatkovnih okvirjev pred začetkom pospeševalniškega cikla. Podatkovni okvirji vsebujejo informacijo o naslednjem pospeševalniškem ciklu, kot na primer trenutni čas, tip pulza, informacijo o kakovosti zadnjega pulza itd. Časovni krmilnik pred pošiljanjem informacije po RTDL povezavi, le te vnaprej najavi s posebnim dogodkom poslanim po EL povezavi, s čimer obvesti sprejemnike o prihajajočih podatkih. Ta najava sicer za samo delovanje ni ključna, vendar je realizirana zaradi obstoječe implementacije, da je doseženo kar najbolj transparentno delovanje. Po končanem pošiljanju podatkov po RTDL povezavi je zaključitev pošiljanja zaznamovana s posebnim dogodkom prav tako poslanim po EL povezavi.



Slika 2.6 Potek pošiljanja informacij po EL in RTDL povezavah.

Slika 2.6 prikazuje časovni potek delovanja pospeševalnika, vključno s potekom določenih dogodkov, ki jih razpošilja časovni krmilnik. Podan je potek pospeševalniškega cikla, ki se vsakič prične z dogodkom CycleStart, kateremu sledijo ostali dogodki, ki so bodisi odgovorni za krmiljenje naprav pospeševalnika, ali pa skrbijo za zaščito pospeševalnika. Takšna interakcija je v tej kritični sekciji izvedena preko sistema za zaščito MPS (ang. Machine Protection Sytem), ki je popolnoma samostojen sistem in neprestano nadzoruje posamezne sklope pospeševalnika ter v

primeru napak ukrepa in preko časovnega krmilnika zahteva takojšnje razpošiljanje ustreznih dogodkov.

Časovni krmilnik je v celoti krmiljen s strani VME IOC (ang. Input Output Controller) gostitelja, kateremu so ob oddaji določenih dogodkov ali pojavitvi ustreznih vhodnih signalov posredovane prekinitvene zahteve, na katere se le ta odzove s programskimi dostopi.

Na VME sistemu teče realno časovni operacijski sistem vxWorks, pod katerim teče kontrolni sistem EPICS [**11**], ki v celoti upravlja s strojno opremo časovnega krmilnika. Namen kontrolnega sistema je upravljati s časovnim krmilnikom na:

a) najvišji ravni, kot je izvajanje relativno počasnih časovno nekritičnih kontrolnih zank, spreminjanje nastavitev, ter dostavljanje podatkov mrežnim odjemalcem.
b) najnižji ravni, saj ima dejansko opraviti s samim krmiljenjem števnih in drugih kartic, s katerimi je dosežena vsa funkcionalnost krmilnika. Tako mora kontrolni sistem periodično programsko zajemati gole podatke s kartic, jih pretvarjati, računsko in logično obdelati, ter zapisovati nazaj.

Prav tako pa se mora sistem deterministično odzivati na sprejete prekinitvene zahteve. Taka funkcionalnost je lahko dosežena, ker celotni sistem VME uporablja realno-časovni operacijski sistem, ki omogoča popoln nadzor nad CPU enoto in njenimi perifernimi napravami.

2.2 Primerjava nove in obstoječe naprave

Zahtevana funkcionalnost razvitega časovnega krmilnika je povsem enaka kot je bila opisana pri obstoječem sistemu, zasnova in implementacija pa sta povsem drugačni:

a) Obstoječa rešitev zajema implementacijo z uporabo večih modulov VME – števnih kartic, ki so med seboj povezane s fiksnimi zunanjimi žičnimi vodniki in zavzema celotno VME ohišje, novo rešitev v celoti predstavlja ena sama tiskanina.
b) Pomembna razlika se odraža z neodvisnostjo delovanja nove naprave od kontrolnega sistema, saj ta za upravljanje s strojno opremo ni več potreben in je njegova naloga le osveževanje podatkov v tabelah in izdajanje zahtev za njihovo pošiljanje. V primeru, ko VME gostitelj in posledično kontrolni sistem, ki upravlja s časovnim krmilnikom odpovesta, le ta kjub temu ohrani privzeto funkcionalnost (pospeševalnik kljub temu

preneha proizvajati žarek). Obstoječa rešitev v primeru kakršne koli napake v kontrolnem sistemu oziroma nedelovanja VME gostitelja, popolnoma odpove kar je usodno za delovanje pospeševalnika.

c) Poleg neodvisnosti pa je pomembno tudi dejstvo, da je veliko funkcionalnosti iz programske podpore in samega kontrolnega sistema preseljeno neposredno v strojni del. Tak primer je izračunavanje popravkov za sledenje fazi omrežne napetosti, ki ga opravlja Master Reference Generator modul, kjer so bili doslej popravki za prilagoditev periodično izračunani v kontrolnem sistemu na VME gostitelju, sedaj pa se popravki izračunavajo popolnoma strojno.

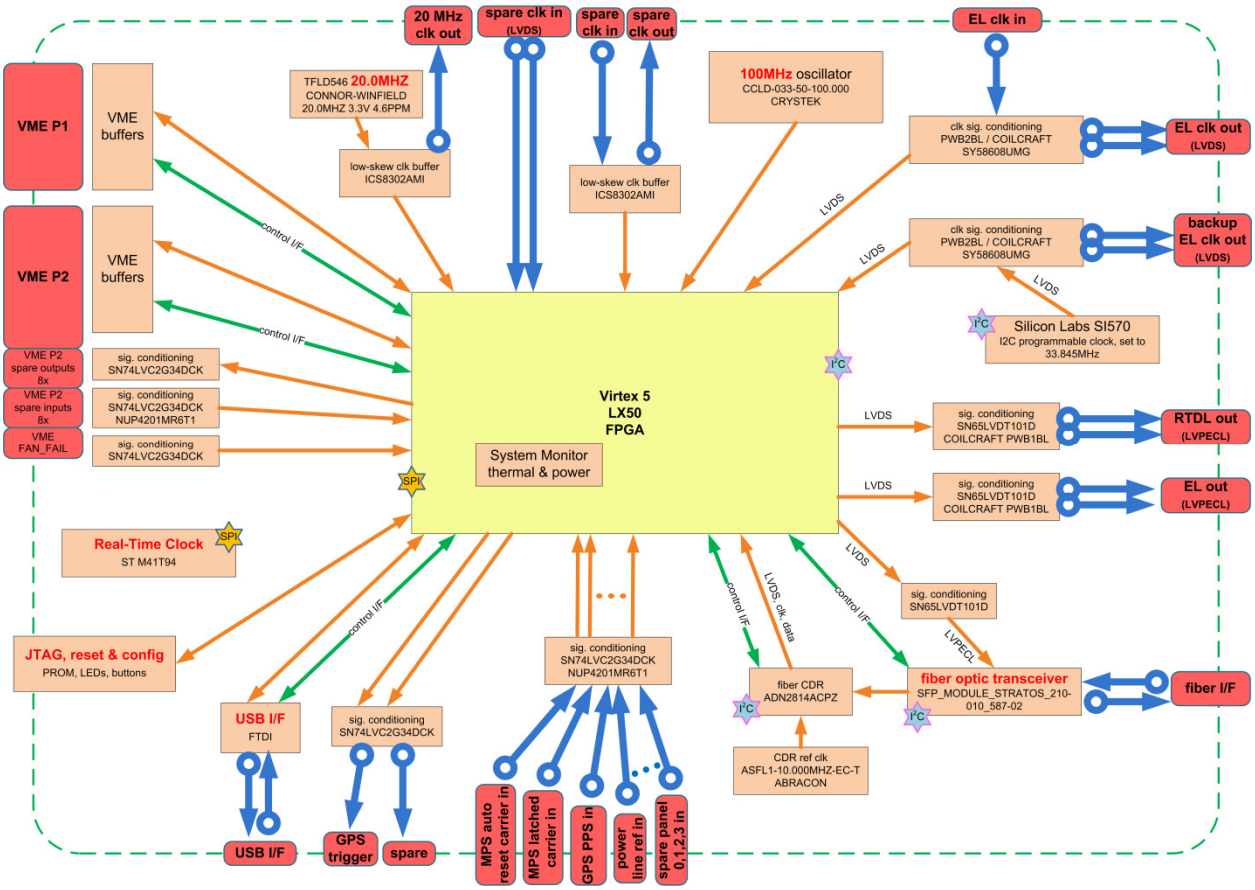
2.3 Načrtovanje in zasnova časovnega krmilnika

Strojna zasnova

Za razvoj nove naprave je uporabljeno Xilinx Virtex5 FPGA vezje, v katerem je implementirana celotna funkcionalnost obstoječe naprave. Poleg tega pa ima naprava vključene še razširitvene priključke, kot so VME vodilo, EL in RTDL povezava.

Časovni krmilnik na sliki 2.7 je realiziran kot VME razširitvena kartica in vsebuje popolno podporo standardiziranemu VME sistemu.

VME sistem je že vrsto let industrijski standard za kritične sisteme saj ponuja zelo stabilno in robustno platformo. Sistem sestavlja priključitveno ohišje, ki vsebuje tiskanino s skupnim podatkovno naslovnim vodilom, arbitrskim vodilom, prekinitvenim vodilom ter pomožnim vodilom in služi kot držalo za vse kartice – module, ki delujejo v sistemu. Za delovanje VME modulov ni potrebna uporaba vseh vodil kar omogoča, da časovni krmilnik uporablja samo 32-bitno naslovno vodilo, 32-bitno podatkovno vodilo, 7-bitno prekinitveno vodilo ter kontrolne signale za rokovanje med prenosi. Vsi signali so z VME vodila preko elektronskih blažilnikov vezani nepostredno na FPGA vezje, kjer je implementirana vsa komunikacijska podpora za rokovanje z vodilom in opravljanje prenosov. Ker sistem sam po sebi poleg drugih modulov nima aktivnih elementov mora biti CPU enota vstavljena v ohišje kot svoj VME modul, ki je z vidika vodila popolnoma enakovreden vsem ostalim modulom.



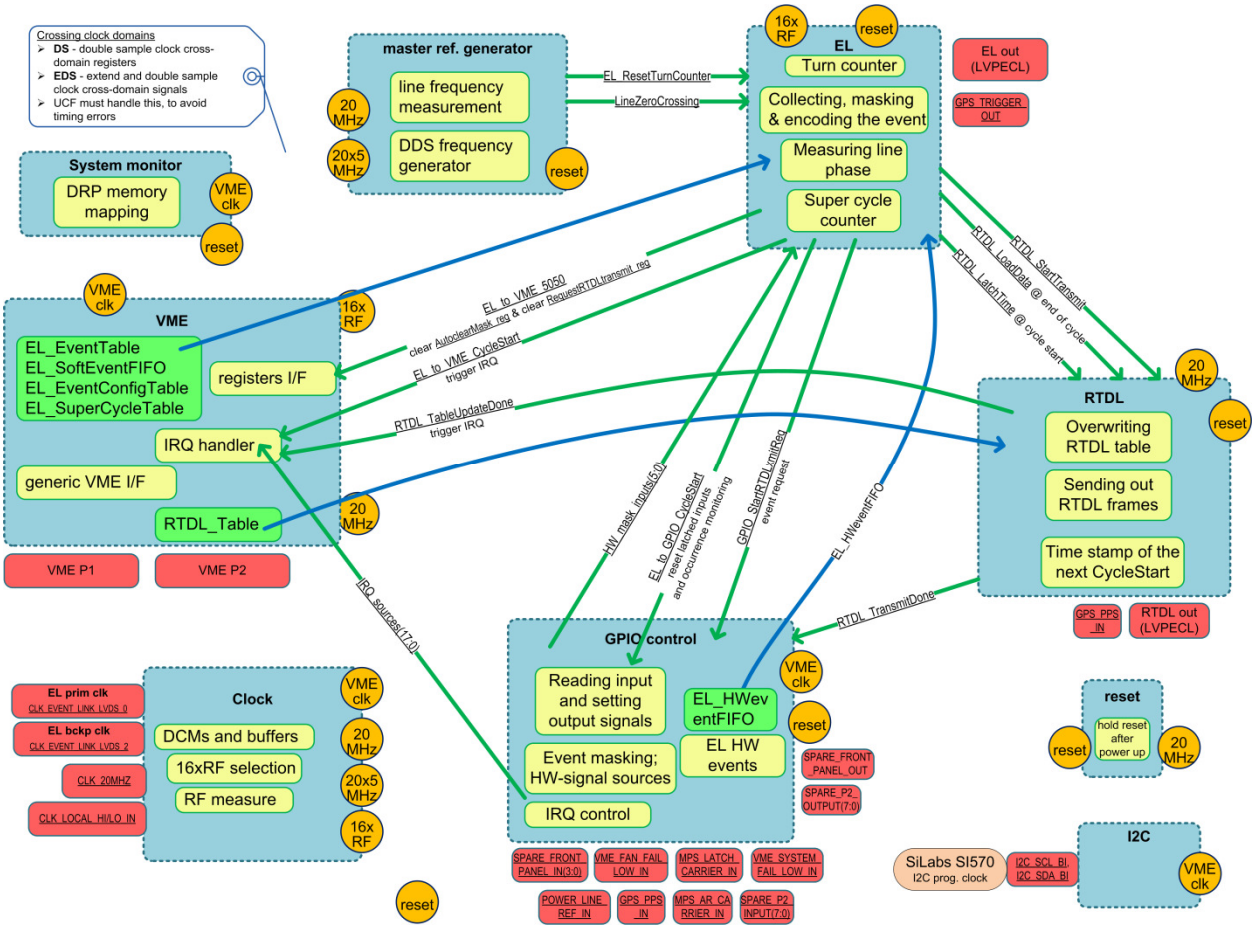
Slika 2.7 Strojna zasnova časovnega krmilnika.

Signali izvedeni iz FPGA vezja so najprej ustrezno električno prirejeni in nato vodeni na posamezne razširitvene priključke. Najpomembnejši del tako predstavljata EL in RTDL povezavi, ki sta realizirani z dvožilnim diferencialnim ščitenim vodom, ter sta galvansko popolnoma ločeni od FPGA vezja.

Tiskanina časovnega krmilnika vsebuje vse komponente za doseganje popolne funkcionalnosti, hkrati pa so na voljo tudi dodatne komponente, kot so programabilni oscilatorji za generiranje pomožnih urinih signalov, temperaturni senzorji in podpora serijski komunikaciji preko USB vodila.

Zasnova VHDL implementacije

Arhitekturna zasnova novega časovnega krmilnika je razdeljena na posamezne module, ki so med seboj povezani v obliki cevovodov in povratnih zank ter so defnirani glede na del funkcionalnosti naprave, ki jo opravljajo. Izvajanje vseh modulov na sliki 2.8 je popolnoma vzporedno in med njimi ni striktnih pogojenosti.



Slika 2.8 Zasnova VHDL implementacije.

Poglavitni moduli, ki sestavljajo časovni krmilnik (slika 2.8) so:

- *modul Clock*

Ker celotna zasnova časovnega krmilnika zahteva uporabo štirih različnih urinih signalov je njihovo generiranje in obdelava implementirano v samostojnem modulu, katerega izhodni urini signali so uporabljeni skozi celotno implementacijo. Modul prav tako zagotavlja signal, ki je uporabljen v *Reset* modulu, kjer je genereriran signal za ponastavitev celotnega časovnega krmilnika. V modulu pa se izvajajo tudi meritve periode primarnega in sekundarnega urinega signala.

- *modul GPIO*

Vsi vhodni in izhodni signali so v celoti obravnavani v GPIO modulu in šele nato posredovani ostalim sklopom. Modul poskrbi za pravilno obravnavanje posameznih signalov z vidika prečkanja urinih domen, saj izvirajo izven FPGA vezja. Modul poskrbi za obravnavanje proženja signalov, saj so le ti največkrat uporabljeni v načinu, kjer nas zanima prehod določenega signala iz visokega v nizko stanje ali obratno. Modul je hkrati odgovoren tudi za proženje prekinitvenih zahtev VME gostitelju. Preko registrov je VME gostitelju na voljo popolna konfiguracija, na kakšen način bodo signali obravnavani, kot tudi omogočanje in onemogočanje obravnavanja posameznih signalov.

- *modul Master Reference Generator*

Glavna naloga modula je sinhronizacija delovnega takta celotnega časovnega krmilnika s prehodi omrežne napetosti skozi ničlo. Poleg tega pa je znotraj modula opravljeno tudi merjenje periode omrežne napetosti, ki jo uporablja kontrolni sistem za izračun popravkov sinhronizacije.

- *modul RTDL*

Poleg dogodkov, ki narekujejo delovanje pospeševalnika morajo biti sprejemnikom ves čas na voljo tudi podatki, ki zajemajo raznovrstne informacije med katerimi je najpomembnejši točen čas naslednjega pospeševalniškega cikla. Vsa informacija o času je vodena znotraj modula

RTDL, za vrednosti parametrov pa je tesno sklopljen z VME modulom preko katerega kontrolni sistem vpisuje vrednosti namenjene za razpošiljanje. RTDL modul je zasnovan tako, da so vse informacije poslane v vsakem pospeševalniškem ciklu.

- *modul EL*

Delovanje pospeševalnika je v celoti odvisno od EL modula, saj le ta s pošiljanjem dogodkov določa kdaj in katere akcije morajo izvrševati sprejemniki, ki so nameščeni po celotni pospeševalniški infrastrukturi.

- *modul VME*

Celotna komunikacija med kontrolnim sistemom in časovnim krmilnikom je izvedena preko VME modula, ki vsebuje podporo za upravljanje z VME vodilom. Poleg samih bralnih in pisalnih dostopov do registrsko preslikanega pomnilnika časovnega krmilnika je modul odgovoren tudi za izdajanje prekinitvenih zahtev VME gostitelju.

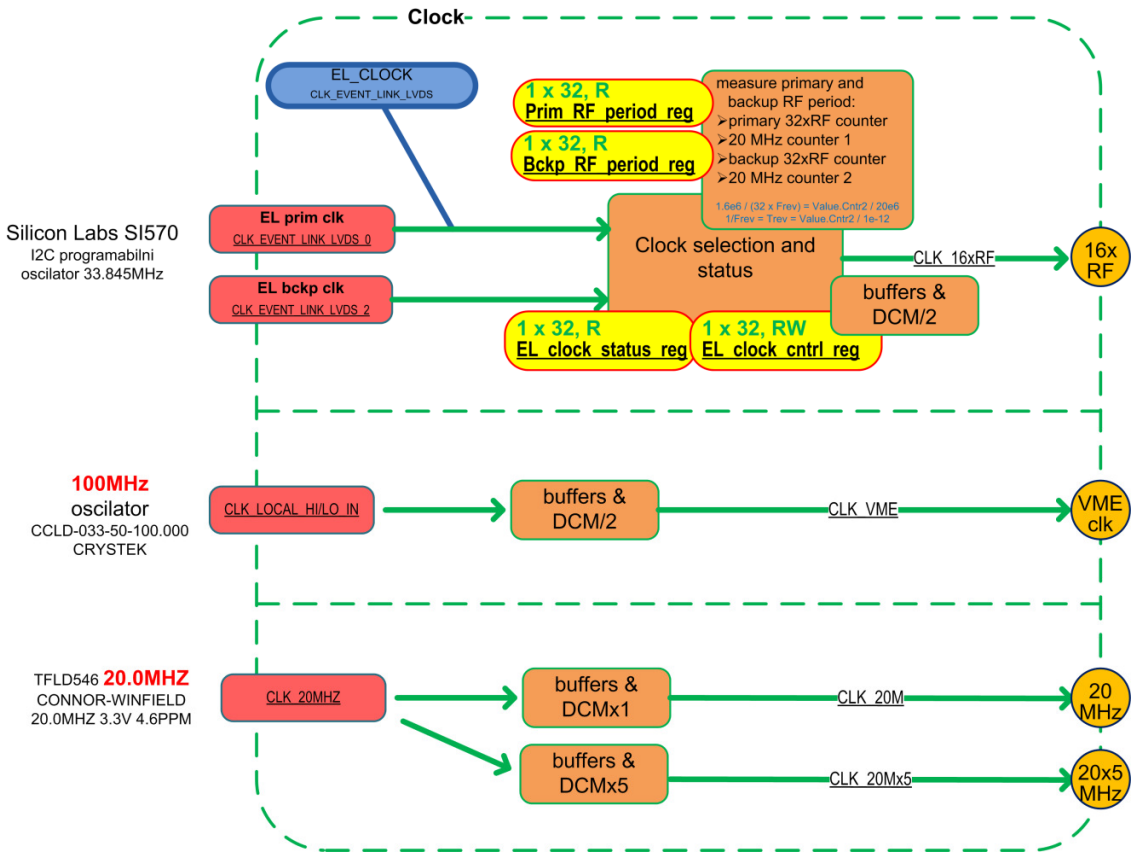
2.4 Uporabljena orodja in razvojna okolja

V panogi razvijalskih orodij namenjenih razvoju VHDL kode velja, da so orodja toga, večinoma neprilagodljiva in težka za uporabo. Pri razvoju časovnega krmilnika uporabljeno okolje ActiveHDL [**12**] ponuja uporabniku prijazen urejevalnik, kjer so hkrati vsebovane lastnosti urejevalnikov namenjenim programiranju v višjih programskih jezikih (Java, C++), kot so samodopolnitev besedila, uporaba predlog in barvno označevanje sintakse.
ActiveHDL je generično orodje za razvoj FPGA vezij, ki ni vezano na določenega proizvajalca, ter za proces sinteze in implementacije VHDL kode uporablja programe, ki jih zagotovi proizvajalec FPGA vezja, ki ga razvijalec uporablja. Pri razvoju časovnega krmilnika je bilo v povezavi z okoljem ActiveHDL uporabljeno razvojno okolje Xilinx ISE Design Suite 11 [**13**].
V sklopu ISE okolja je bil samostojno uporabljen tudi logični analizator ChipScope Pro [**14**], ki omogoča zajem trenutnega stanja signalov v FPGA vezju, ki nas zanimajo. Okolje vsebuje tudi zmogljiv simulator, ki je sklopljen s Tcl interpreterjem [**15**] in simulacijskim jedrom SystemC [**16**].

3. Implementacija

3.1 Modul Clock

Funkcija modula na sliki 3.1 je zagotavljanje vseh potrebnih urinih signalov, ki so uporabljeni v časovnem krmilniku. Signali so pridobljeni iz bodisi zunanjih izvorov ali izvorov vgrajenih na tiskanini. Za večino signalov je uporabljena tehnologija diferencialnega prenosa signalov, kjer je vsak urin signal pripeljan v FPGA integrirano vezje z dvema vodnikoma, katerih vrednosti vedno nastopajo v protifazi.



Slika 3.1 Arhitekturna zasnova Clock modula.

Poglavitno pravilo pri implementaciji narekuje, da so vsi urini signali pred dejansko uporabo sprva vodni v blažilnik IBUFGDS (ang. Differential Signaling Dedicated Input Clock Buffer and Optional Delay), ki diferencialni urin signal pretvorijo v eno-vodni signal, nato pa preko DCM

(ang. Digital Clock Manager) primitiva in izhodnega blažilnika BUFG (ang. Global Clock Buffer) nastopajo v sami matriki logičnih celic FPGA vezja.

DCM komponente so primitivni elementi kar pomeni, da so v FPGA vezju realizirane kot kompleksne komponente, ki jih lahko uporabljamo kot entitete in z njihovo podrobno sestavo nimamo opravka. Njihova naloga in zmožnost je množenje in deljenje vhodnega urinega signala, kot tudi izvajanja morebitnega nespremenljivega ali dinamičnega faznega zamika izhodnega urinega signala glede na vhodni signal. Temeljijo na uporabi PLL (ang. Phase Locked Loop) vezij. Komponenta v osnovi proizvaja naslednje izhodne urine signale:

$$2 \times f_{CLKIN}$$
$$\frac{M}{D} f_{CLKIN}$$
$$f_{CLKIN}(\emptyset = 90°)$$
$$f_{CLKIN}(\emptyset = 180°)$$
$$f_{CLKIN}(\emptyset = 270°),$$

kjer je f_{CLKIN} frekvenca vhodnega urinega signala, M (ang. Multiply) vrednost s katero želimo vhodni signal pomnožiti, D (ang. Divide) vrednost s katero želimo frekvenco urinega signala deliti, $\emptyset$ pa poda željen fazni zamik izhodnega urinega signala glede na vhodni signal.

DCM komponente poleg množenja in deljenja urinih signalov skrbijo tudi za izničevanje t.i. migetanja signala (ang. jitter), kjer uporabljajo povratno zanko za samodejno sinhronizacijo izhodnega signala, ki potuje skozi vezje in žene gradnike, z vhodnim signalom v primeru če/ko pride do faznih sprememb. Hkrati delujejo tudi kot blažilniki ob kratkotrajni izgubi vhodnega urinega signala saj lahko na izhodu nemoteno proizvajajo željen urin signal, če izgubimo do 3 periode vhodnega signala. V primeru izgube več period vhodnega signala pa je potrebno sprožiti ponovno inicializacijo komponente, ki prisili PLL zanko da se ponovno zaklene na vhodni signal. Stanje DCM koponente je vseskozi na voljo preko namenskega signala, ki označuje ali je PLL zanka zaklenjena na vhodni signal, kot tudi podrobnejše stanje delovanje preko statusnega registra.

Modul zagotavlja 4 urine signale:

CLK_16xRF	$f_{CLKIN} = 33.845MHz;\ f_{CLKOUT} = 16MHz$
CLK_VME	$f_{CLKIN} = 100MHz;\ \ f_{CLKOUT} = 50MHz$
CLK_20M	$f_{CLKIN} = 20MHz\ ;\ f_{CLKOUT} = 20MHz$
CLK_20x5	$f_{CLKIN} = 20MHz;\ \ f_{CLKOUT} = 100MHz$

Preklop med izvori urinih signalov

Časovni krmilnik potrebuje za delovanje dogodkovne serijske povezave (EL) dva vira urinega signala. Prvi in hkrati primarni izvor izvira iz zunanjega preciznega oscilatorja ter sekundarni-pomožni izvor, ki je pridobljen iz preciznega oscilatorja na tiskanini. Delovanje je načrtovano in implementirano tako, da je ob prisotnosti zunanjega signala uporabljen le-ta, v primeru izpada pa se izvede samodejni preklop na pomožni izvor s tiskanine. Preklop signala je izveden z uporabo namenskega primitiva FPGA vezja, multiplekserja urinega signala BUFGCTRL (ang. Global Clock MUX Buffer), ki glede na asinhron izbirni vhod izvaja sinhron preklop med dvema vhodnima signaloma. Povrnitev na primarni izvor urinega signala je mogoča le ob eksplicitni zahtevi VME gostitelja, katero preko VME modula programsko izvrši kontrolni sistem.

Za posebno težaven del implementacije modula se je izkazalo pravilno ponastavljanje DCM komponente, ki upravlja s primarnim urinim signalom, ki prihaja iz zunanjega vira. Ob izgubi zunanjega urinega signala se desinhronizira DCM komponenta, ki upravlja s tem signalom. DCM komponenta sicer podaja svoj status z LOCKED signalom, ki sporoča ali je interna PLL zanka zaklenjena in je urin signal na izhodu komponente veljaven, vendar ta signal ne odraža stanja hipoma temveč se vrednost pravilno nastavi šele čez čas po desinhronizaciji. Tak signal ni zanesljiv, ko potrebujemo hiter odziv in mora vezje ob izgubi primarnega urinega signala nemudoma preklopiti na pomožni izvor. Zato je poleg LOCKED signala uporabljen tudi DCM statusni register, ki bolj podrobno podaja status komponente, ki hipoma poda stanje PLL. Slabost statusnega registra pa je časovna zakasnitev, ki preteče od ponovne uspešne sinhronizacije PLL na vhodni urin signal, do uspešnega odraza stanja v stanjskem registru. Končna rešitev zajema kombinacijo obeh signalov, kjer je tako moč zaznati izgubo primarnega urinega signala in sprožiti preklop na sekundarni izvor. Ob ponovni prisotnosti zunanjega urinega signala se

periodično izvaja avtomatska ponastavitev DCM komponente, ki upravlja s tem signalom, sam preklop na ta vir pa se ne sme izvršiti, dokler tega ne zahteva kontrolni sistem.
Poleg samega urejanja urinih signalov, pa je naloga modula tudi zagotavljanje informacije o stanju urinih sinalov in izvorov, kot tudi merjenje periode urinih signalov. Ti podatki so neposredno na voljo kontrolnemu sistemu preko registrov VME modula.

Merjenje periode vhodnih signalov

Meritve periode se opravljajo za signal CLK_16xRF in sicer za oba vira signala. Osnovane so na števcu, ki je voden z urinim signalom frekvence 20MHz za časovno obdobje v katerem doseže števec voden z urinim signalom CLK_16xRF (16MHz) vrednost 800000. Vrednost prvega števca (counter(1)) predstavlja RF (ang. Revolution Period) periodo v pikosekundah. Vsaka meritev poteka približno 47 ms, kar ustreza trem ciklom pospeševalnika.

$$\frac{800 \times 10^3}{32 \times F_{REVOLUTION}} = \frac{counter}{20 \times 10^6} \quad (1)$$

$$\frac{1}{F_{REVOLUTION}} = T_{REVOLUTION} = \frac{counter}{1 \times 10^{-12}} \quad (2)$$

Signal za ponastavitev naprave

Modul poleg urinih signalov priskrbi tudi signal za ponastavitev celotne naprave in sicer z uporabo DCM komponente, ki upravlja z 20 MHz urinim signalom, ki izvira iz oscilatorja na tiskanini. Uporaba te DCM komponente za generiranje signala za ponastavitev je veljavna, ker je izvor urinega signala, ki ga uporablja popolnoma stabilen in vedno prisoten. Zaradi tega je pravilno delovanje DCM komponente zagotovljeno, saj ne obstaja možnost, da bi vhodni urin signal izgubili.
Uporabljen je LOCKED signal DCM komponente, ki zavzema nizko logično vrednost, vse dokler se PLL zanka DCM komponente uspešno ne zaklene na vhodni urin signal, oziroma ostane nizek, dokler ni izhodni urin signal komponente popolnoma stabilen in veljaven za nadaljno uporabo v FPGA vezju. Ker pa čas, v katerem LOCKED signal zavzame visoko vrednost ni popolnoma deterministično, je ta signal uporabljen zgolj za krmiljenje preprostega vezja za ponastavitev, ki sestoji iz števca, ki začne z odštevanjem proti ničli, takoj ko DCM

komponenta zaključi z inicializacjio in je LOCKED signal postavljen visoko, hkrati pa se signal za ponastavitev postavi visoko. Števec (slika 3.2) je implementiran tako, da v trenutku ko doseže ničlo, preneha z odštevanjem, signal za ponastavitev pa zavzame nizko vrednost in tako omogoči delovanje naprave. Na takšen način je zagotovljena vedno enaka dolžina ponastavitvenega signala.

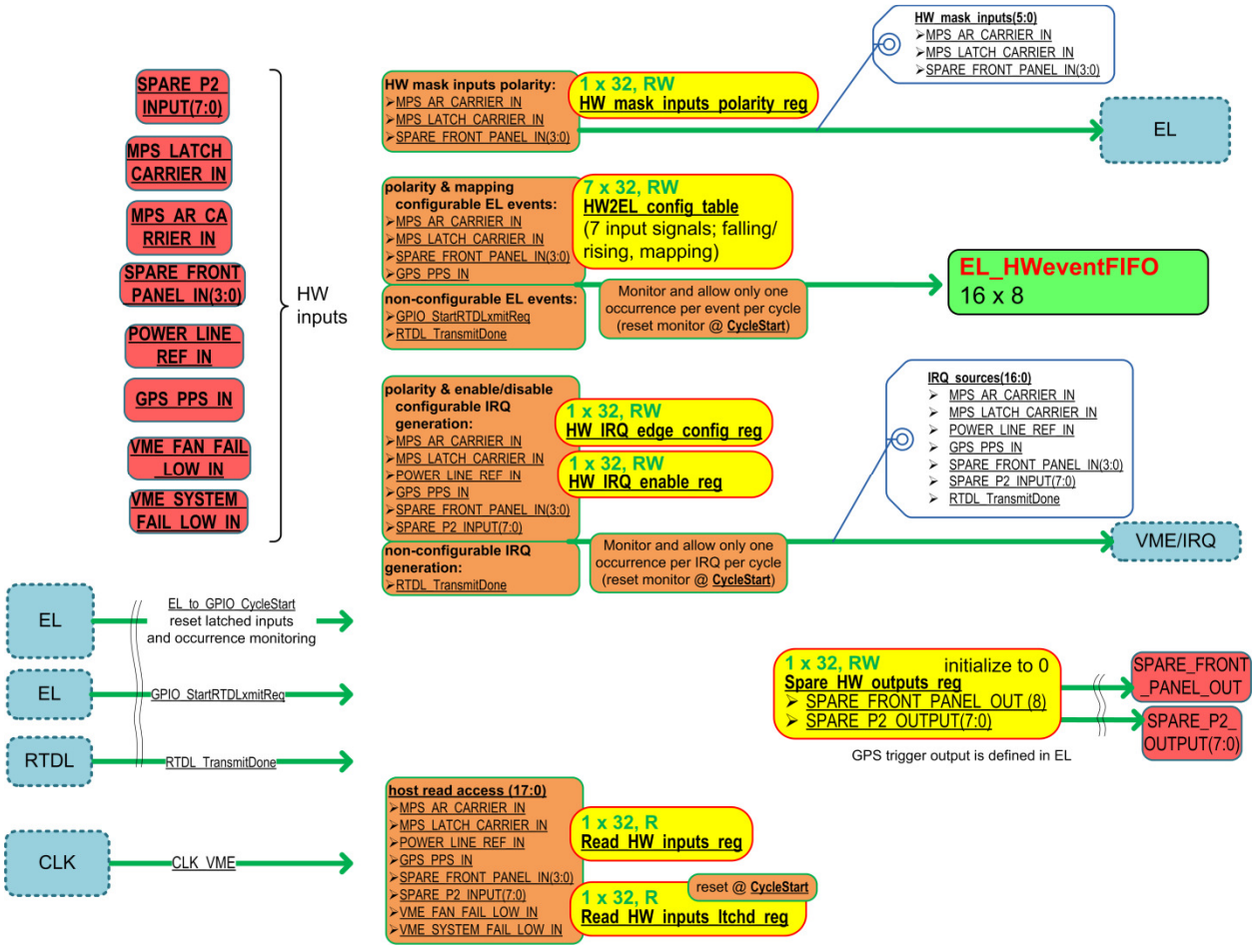
```
entity reset_circ is
        port(
                CLK_20M : in STD_LOGIC;
                DCMLckd_i : in STD_LOGIC;
                Reset : out STD_LOGIC
          );
end reset_circ;

architecture RTL of reset_circ is
        signal s_cntr : std_logic_vector(15 downto 0);
begin
        p_syncCntr : process(CLK_20M, DCMLckd_i)
        begin
                if DCMLckd_i='0' then
                        s_cntr <= (others=>'1');
                        Reset <= '1';
                elsif rising_edge(CLK_20M) then
                        if s_cntr=0 then
                                Reset <= '0';
                        else
                                s_cntr <= s_cntr - 1;
                                Reset <= '1';
                        end if;
                end if;
        end process p_syncCntr;
end RTL;
```

Slika 3.2 Implementacija števca za ponastavitev vezja.

3.2 Modul GPIO

Modul GPIO (ang. General Purpose Input Output), ki je predstavljen na sliki 3.3, upravlja z vsemi vhodnimi in izhodnimi signali, ki jih uporablja časovni krmilnik. Poleg tega določa na kakšen način se generirajo prekinitvene zahteve VME gostitelju in kako se generirajo dogodki namenjeni razpošiljanju po EL povezavi. VME gostitelju nudi možnost nastavljanja polaritete določenih signalov (proženje na pozitivno ali negativno fronto), kateri so nato posredovani bodisi VME gostitelju kot prekinitve, ali pa uporabljeni kot maska, ki določa kateri dogodki se lahko pošljejo po EL povezavi.



Slika 3.3 Arhitekturna zasnova GPIO modula.

Upravljanje in dostopanje do signalov

VME gostitelj lahko dostopa samo do 32-bitnega bralnega registra, v katerem so vsebovane trenutne vrednosti 18 signalov (*Spare_P2_Input(7:0)*, *Power_Line_Ref_In*, *GPS_PPS_IN*, ...), ki so na voljo napravi. Poleg trenutnih vrednosti signalov, pa so na voljo tudi pomnjene vrednosti signalov, saj se odvisno od konfiguracije, ob sprejemu pozitivne/negativne fronte signala, le ta postavi v visoko stanje in se ponastavi šele z začetkom vsakega pospeševalniškega cikla (pozitivna fronta signala Cycle Start dobljenega iz EL modula). Obravnavanje vseh signalov, ki so vodeni v registre ali pa so uporabljeni kot prožilci so pred njihovo dejansko uporabo vodeni skozi komponento za večkratno vzorčenje, da se izniči možnost pojavitve metastabilnih stanj. Prav tako so enako obravnavani vsi signali, ki prihajajo direktno v FPGA vezje iz zunanjih integriranih vezij, saj je njihovo obnašanje popolnoma asinhrono z delovanjem notranjih komponent v FPGA vezju in ne smejo biti neposredno uporabljeni. Večkratno vzorčenje pa v primeru branja vrednosti registrov v smeri VME modula proti GPIO modulu ni potrebno, saj oba modula delujeta v enaki urini domeni in nevarnost metastabilnih stanj ne obstaja.

Vhodno/izhodne prekinitvene zahteve

Časovni krmilnik vsebuje 16 prekinitvenih signalov, izvorov. Vsi izmed njih imajo popolnoma nastavljiv način proženja, na pozitivni ali negativni prehod. Vzorčenje posameznih signalov pa je lahko omogočeno ali onemogočeno. S temi nastavitvami v celoti upravlja VME gostitelj preko namenskih registrov. Poleg nastavljivih signalov je implementiran tudi signal (*RTDL_TransmitDone*), ki je postavljen vsakič, ko je končano pošiljanje podatkov po RTDL povezavi.

Zasnova zagotavlja, da se vsak izmed 17-ih signalov lahko proži le enkrat v pospeševalniškemu ciklu, ne glede na to kolikokrat je bil določen signal dejansko prožen. Ponastavitev sproženih – izdanih prekinitvenih zahtev se izvede ob vsakem začetku pospeševalniškega cikla. Vsi signali, ki služijo kot prekinitvene zahteve, so vodeni v VME modul, kjer so ob njihovem sproženju preslikani v unikatne enumeracije in vstavljeni v FIFO element.

GPIO modul opravi le začetno maskiranje signalov, ki služijo kot prekinitvene zahteve, vse nadaljnje nastavitve pa so opravljene s strani VME gostitelja oz. kontrolnega sistema, ki teče na njem.

Generiranje dogodkov za modul EL

Modul GPIO upošteva 7 vhodnih signalov kot izvore za generiranje EL dogodkov. Podobno, kot signali uporabljeni za generiranje prekinitvenih zahtev, je tudi obravnavanje teh signalov popolnoma nastavljivo, kjer je določljiva fronta ob kateri bo signal upoštevan. Tudi tu obstajata dva nekonfigurabilna signala (*GPIO_StartRTDLxmitReg* in *RTDL_TransmitDone*), ki sta vedno upoštevana ob njuni pozitivni fronti. Omenjena signala označujeta zahtevo za razpošiljanje dogodka po EL povezavi, ki najavi prihajajoči podatkovni tok po RTDL povezavi, ter zahtevo za razpošiljanje dogodka po EL povezavi, ki označuje konec prenosa podatkov po RTDL povezavi.
Za vseh 9 signalov je zagotovljeno, da vsak izmed njih lahko sproži zahtevo za razpošiljanje dogodka po EL povezavi le enkrat v pospeševalniškem ciklu, ne glede na to kolikokrat se dejansko proži. Ponastavitev sproženih – izdanih zahtev se izvede ob vsakem začetku pospeševalniškega cikla, ob pozitivni fronti prehoda signala *EL_to_GPIO_CycleStart*.
Vsak izmed teh signalov je ustrezno preslikan v unikatno enumeracijo EL dogodka, ki je implementirana z majhno ROM tabelo, katere vsebina je statična, nato pa so vstavljeni v 16 x 8 bit FIFO element, kjer dogodki čakajo v vrsti, da so prebrani in upoštevani pri pošiljanju po EL povezavi. Dogodki so zaporedno vstavljeni v FIFO element, četudi se pojavijo ob enakem času. Sam FIFO element je implementiran v GPIO modulu, kot tudi vezje za vstavljanje dogodkov vanj. Zajem dogodkov, oziroma branje pa je realizirano v EL modulu, do kamor je vodeno podatkovno vodilo in kontrolni signali bralne strani FIFO elementa.

Maska za razpošiljanje El dogodkov

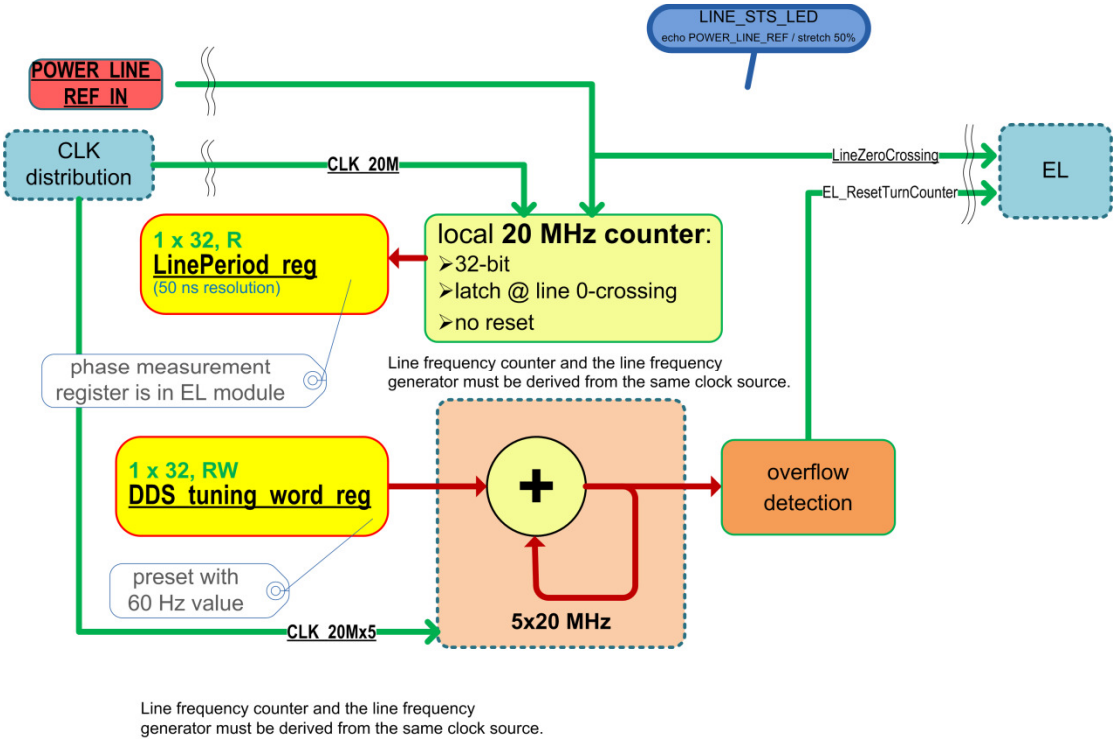
Modul EL pred razpošiljanjem dogodkov z masko preveri izpolnjenost pogojev, ki določajo ali je lahko dogodek razposlan. Del maske module EL sestavlja 6 signalov, ki izvirajo iz GPIO modula. Tem signalom lahko določamo polariteto oziroma način proženja (pozitivna ali negativna fronta prehoda) preden so signali dejansko uporabljeni v EL modulu.

Splošno namenski izhodi

Modul GPIO upravlja tudi s splošno namenskimi izhodi, ki so dostopni VME gostitelju. Vrednosti, ki jih vpisuje VME gostitelj, se neposredno, brez modificiranja odražajo na izhodih. Izhodi so iz FPGA vezja preko zunanjih integriranih vezij vodeni neposredno na čelno ploščo tiskanine, kot tudi na P2 konektor VME vodila.

3.3 Modul Master Reference Generator

Glavna naloga referenčnega generatorja (ang. Master Reference Generator) na sliki 3.4 je priskrbeti signale potrebne za sinhronizacijo celotnega sistema z omrežno napetostjo. Takšna funkcionalnost je podana z zahtevo, da mora prehod omrežne napetosti skozi ničlo nastopiti točno takrat, ko vrednost števca obhodov (ang. Turn Counter) zavzema vrednost 5050.



Slika 3.4 Arhitekturna zasnova Master Reference Generator modula.

Mehanizem, za doseganje zahtevane funkcionalnosti je izvajanje ponastavitve števca obhodov na vrednost nič ob takem času, da bo v naslednji iteraciji omrežna napetost prečkala ničlo, ko bo vrednost števca natanko 5050. Odvisno od energije žarka in nihanja frekvence omrežne napetosti, se to tipično zgodi, ko števec obhodov zavzema vrednosti med 17,022 in 18,412.

Sprva je zasnova predvidevala, da mora ves izračun popravkov – časov kdaj je potrebno ponastaviti števec obhodov, opraviti VME gostitelj, oziroma kontrolni sistem. Vendar je bilo po končani implementaciji, kot dodatek določeno, da se popravki izračunavajo strojno, popolnoma avtonomno, brez nepotrebne interakcije s kontrolnim sistemom.

VME gostitelju mora biti tako na voljo informacija o fazi, ki pa je dejansko vrednost števca obhodov registrirana ob prehodu omrežne napetosti skozi ničlo. Kontrolni sistem mora voditi zgodovino izmerjenih vrednosti, saj dobi podatek o razliki faze šele, ko od trenutne vrednosti števca obhodov odšteje predhodno vrednost. Kot že omenjeno, se vrednost števca obhodov za to meritev zabeleži, ko omrežna napetost prečka ničlo. Razlika v vrednostih dejansko prikaže fazno razliko med omrežno napetostjo ter signalom, ki ga generira modul za ponastavljanje števca obhodov. Sistem strmi k temu, da je ta fazna razlika čim manjša in se ne spreminja oziroma so razlike čim manjše.

Merjenje periode omrežne napetosti

Znotraj modula se nahaja sinhroni števec, ki deluje z urinim signalom frekvence 20 MHz, njegova vrednost pa je zapisana v register vsakič, ko omrežna napetost prečka ničlo. Ta informacija je zapisana v namenskem registru in je na voljo VME gostitelju. Da kontrolni sistem na VME gostitelju pridobi informacijo o periodi omrežne napetosti, mora zadnjo prebrano vrednost odšteti od trenutne prebrane vrednosti:

$$T_{LINE} = (LinePeriod\_reg_{CURRENT} - LinePeriod\_reg_{PREVIOUS}) \times 50ns \quad (3)$$

Direktna digitalna sinteza signala

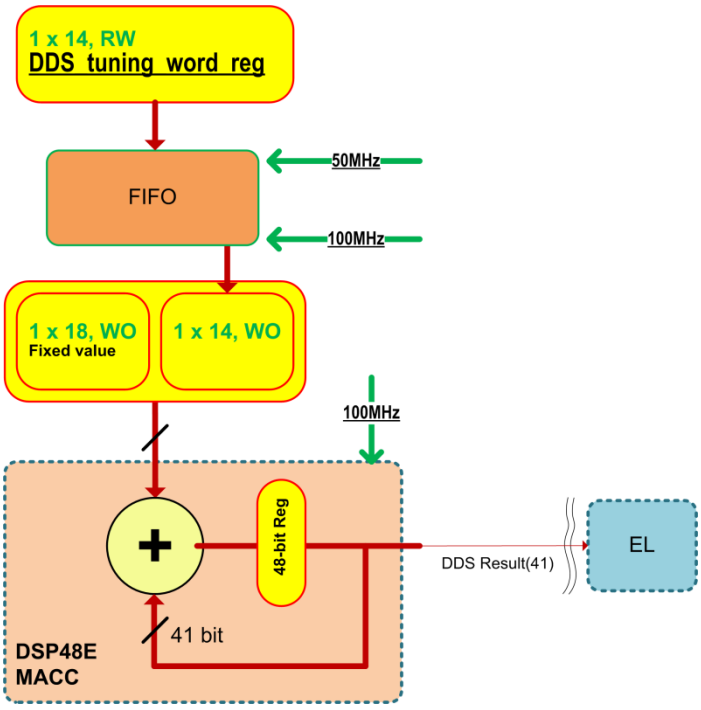
VME gostitelj je odgovoren za potreben izračun sledenja frekvenci in fazi omrežne napetosti. Iz informacije o periodi in fazi mora izračunati ciljno frekvenco signala, ki ga generira Master Reference Geneator modul in je uporabljen za ponastavitev števca obhodov.

Signal za ponastavitev je sintetiziran z uporabo DDS [**17**] (ang. Direct Digital Synthesis) koncepta. Uporabljen je 48-bitni sinhroni seštevalnik, kjer se ob vsaki iteraciji pri frekvenci urinega signala 100 MHz trenutni vsoti prišteje vrednost krmilne besede (4) (DDS_tunning_word_reg). Ker je krmilna beseda 41-bitna, znaša resolucija nastavljanja frekvence tako proizvedenega signala 45µHz.

$$\text{EL\_ResetTurnCounter} = (\sum_{0}^{\infty} \text{DDS\_tunning\_word\_reg})[41] \quad (4)$$

Celotna funkcionalnost DDS dela modula (slika 3.5) je zasnovana tako, da omogoča le delno nastavljanje krmilne besede in sicer le spodnjih 14 bitov medtem, ko so preostali višji biti nespremenljivi in določeni v sami implementaciji. Tako je doseženo, da kontrolni sistem lahko spreminja frekvenco izhodnega signala le v območju med 59.8 - 60.2 Hz. Tako največja možna razlika znaša le 0.4 Hz, ki pa je še sprejemljiv frekvenčni skok za sprejemnike in naprave, ki so z njimi krmiljeni.

Končni signal, ki je uporabljen za ponastavitev števca obhodov, je bit 41 registra seštevalnika, ki vsebuje trenutno vsoto.



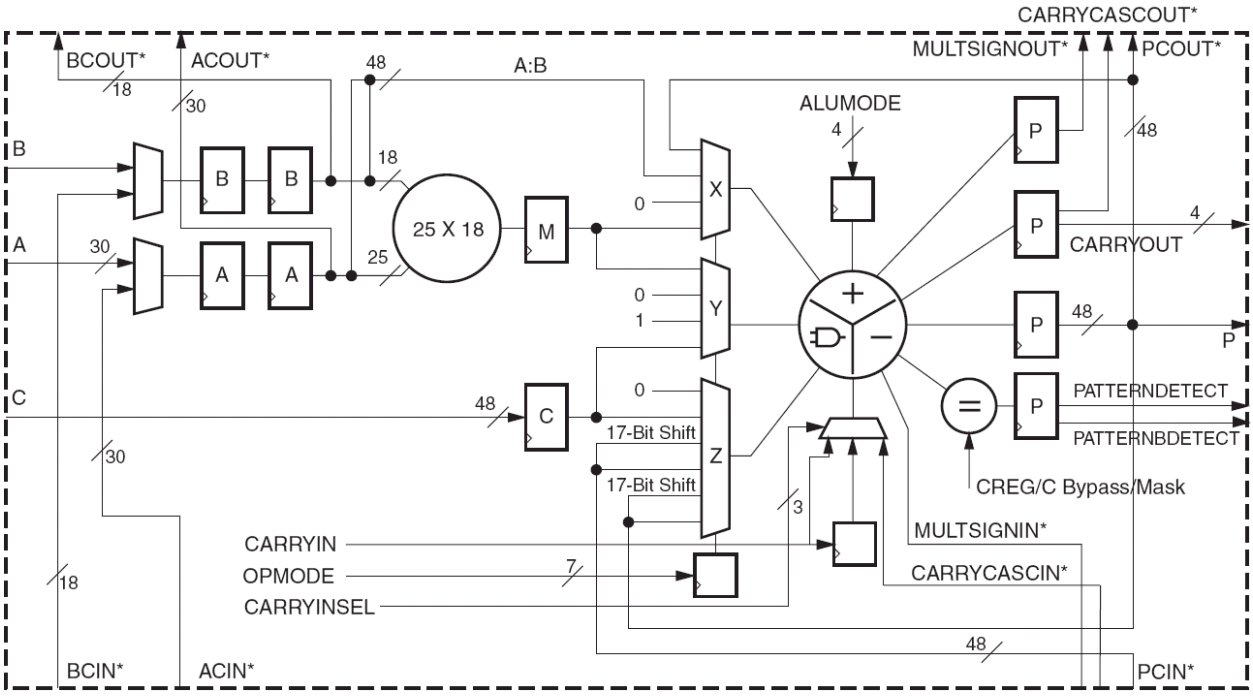
Slika 3.5 Zasnova DDS komponente.

Delovanje DDS sklopa (slika 3.5) se je med samo implementacijo izkazalo kot dokaj problematično, saj je zahtevano, da mora komponenta delovati pri frekvenci urinega signala 100 MHz. Realizirati je bilo potrebno preprost 48-bitni seštevalnik, vključujoč tudi enako širok register za hranjene trenutne vrednosti vsote ter komponento za večkratno vzorčenje krmilne besede. Če je bil proces sinteze izvršen samo nad tem sklopom brez ostalih komponent časovnega krmilnika, je sklop teoretično lahko dosegel zelo dobro časovno performanco, saj je frekvenca urinega signala lahko znašala preko 200 MHz. Pri sintezi celotnega časovnega krmilnika pa sklop ni bil zmožen delovati pri željeni frekvenci.

Največji problem pri procesu sinteze in implementacije VHDL kode je predstavljalo večkratno vzorčenje krmilne besede namenjeno za odpravo možnosti pojavitve metastabilnih stanj, saj pisanje vrednosti krmilne besede poteka v drugi urini domeni – s strani VME modula.

Namesto običajnega načina za večkratno vzorčenje vodil je bil tukaj uporabljen FIFO pomnilni element, ki je zelo učinkovita rešitev za prečkanje različnih urinih domen, ko uporabljamo široka vodila. Poleg samega FIFO elementa, je bilo potrebno na pisalno in bralno stran dodati tudi manjša vezja za krmiljenje samega FIFO elementa, ki poskrbijo, da je upravljanje s FIFO elementom popolnoma transparentno in ga komponente obravnavajo kot običajni register.

Seštevalnik je bil realiziran z uporabo DSP48 rezin (ang. MACC – Multiply And Accumulate), ki so strojno realizirane aritmetično logične enote v FPGA vezju. Značilnost teh elementov je, da so zmožni opravljati preproste, kot tudi zahtevnejše matematične operacije pri zelo visokih frekvencah urinega signala. Elementi omogočajo operacije nad vodili do širine 48 bitov, izvajanje zahtevnejših operacij, kot je množenje pa je izvedeno v največ dveh urinih periodah, kar je vsekakor dobrodošla lastnost. Uporabljena MACC enota je izkoriščena samo kot seštevalnik, kjer je množenje neuporabljeno, oziroma se za faktor uporablja konstantna vrednost 1.

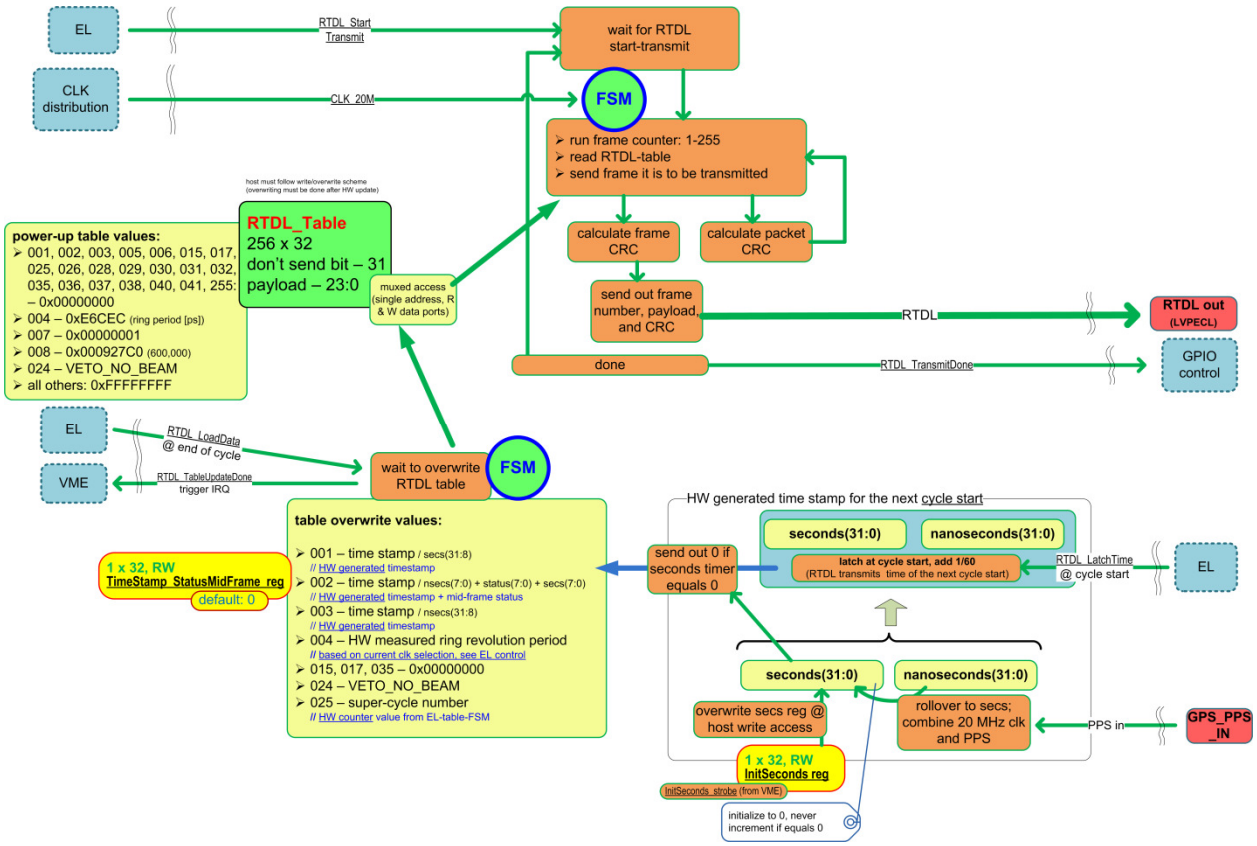


Slika 3.6 Struktura MACC elementa.

Slika 3.6 prikazuje shemo MACC primitivnega elementa DSP48 rezine, ki je uporabljena za implementacijo DDS sklopa. Element je konfigurian kot seštevalnik, izkoriščena je notranja povratna zanka, saj je namesto da bi bil rezultat seštevalnika voden preko »tkanine« FPGA vezja nazaj na vhod MACC elementa, celotna povezava opravljena znotraj DSP48 rezine. Tak način je performančno mnogo boljši, saj je interno povezovanje vodil v DSP48 rezini implementirano optimalno in ga ni moč spreminjati. Prav tako pa so uporabljeni interni registri (zatiči) v MACC enoti s čimer ni bilo potrebe po potratnem registriranju signalov med izhodom FIFO elementa in vhodom seštevalnika.

3.4 Modul RTDL

Naloga modula RTDL (slika 3.7) je oskrbovanje sprejemnikov z informacijo o točnem času, informacijo o statusu in vrsti pospeševalniških ciklov ter ostalimi ključnimi parametri. Podatki se pošiljajo periodično, vendar čas oziroma perioda pošiljanja ni strogo deterministična, ker je pošiljanje RTDL podatkov nekritično opravilo z nižjo prioriteto. Zasnova kljub temu zagotavlja, da bodo podatki poslani po vsakem pospeševalniškem ciklu – po vsakem dogodku *CycleStart*.



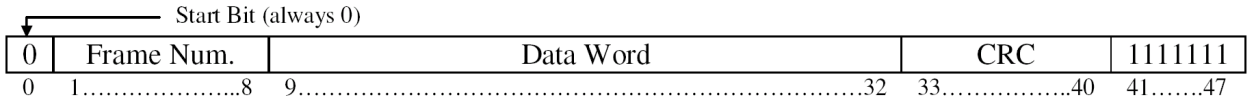
Slika 3.7 Arhitekturna zasnova RTDL modula.

Zapis in osveževanje RTDL podatkov

Zapis podatkov v tabelo se lahko vrši interno v RTDL modulu, poleg tega pa lahko do nje dostopa tudi VME modul, preko katerega kontrolni sistem zapisuje poljubne vrednosti. Dostop je časovno multipleksiran, kjer je interno pisanje v tabelo dovoljeno samo določen čas in je predhodno zahtevano s strani modula EL.

Sestava in pošiljanje podatkov

Podatki oziroma parametri so hranjeni v eni sami pomnilniški tabeli, velikosti 256 32-bitnih lokacij, kjer je za podatke uporabljenih le 24 bitov.



Slika 3.8 Sestava RTDL podatkovnega okvirja.

Kot prikazuje slika 3.8, je vsaka vrednost tabele predstavljena s svojim 48-bitnim okvirom, kjer je vsebovan začetni bit (bit 0), 8-bitna zaporedna številka paketa, 32-bitna vrednost (podatki zavzemajo le 24 bitov), 8-bitna CRC (ang. Cyclic Redundancy Check) vrednost, ter 7-bitni zaključek.

Pošiljanje podatkovnih okvirjev je implementirano s končnim avtomatom, ki v nedejavnem stanju čaka na zahtevo *RTDL_StartTransmit*, s strani EL modula. Posredno lahko zahtevo za pošiljanje izda tudi VME gostitelj, oziroma kontrolni sistem, preko dogodka, ki je vstavljen v *EL_SoftEventFIFO* pomnilni element, ter je s strani EL modula obravnavan takoj, ko je to mogoče. Če VME gostitelj pošiljanja ne zahteva, se eventuelno izteče največji dovoljeni čas, ki lahko preteče med posameznimi pošiljanji RTDL podatkov in se le to izvrši avtomatsko. Zahteva za pošiljanje pride s strani EL zato, ker mora biti pošiljanje RTDL okvirjev predhodno najavljeno s posebnim dogodkom poslanim po EL povezavi. Po prejemu zahteve, avtomat prebere vsebino vsake izmed 256 lokacij tabele, ter vrednost pošlje če MSB bit v 32-bitni vrednosti (Data Word) ni postavljen. Po končanem pošiljanju avtomat postavi signal *RTDL_TransmitDone*, ki je voden v

GPIO modul in služi kot povratna zanka med moduli. GPIO modul ob sprejemu potrditve o končanem pošiljanju v *EL_HWeventFIFO* vstavi dogodek, s katerim so sprejemniki po EL povezavi obveščeni o koncu prenosa po RTDL povezavi.

Izračun vrednosti CRC

Zadnji prenešeni okvir je vedno okvir 255, ne glede na to koliko okvirjev je bilo dejansko poslanih. Ta okvir kot podatek vsebuje CRC vrednost vseh vrednosti, ki so bile poslane. Čeprav vsak paket prav tako vsebuje CRC vrednost, je namen tega okvirja zagotoviti možnost dodatnga preverjanja veljavnosti vseh poslanih okvirjev.

Izračun časa naslednjega cikla

Prvi trije poslani RTDL okvirji vsebujejo vrednosti časa ob katerem se bo zgodil naslednji cikel, kjer je ta informacija podana s sekundami in nanosekundami. Čas začetka naslednjega cikla je pridobljen z registriranjem vrednosti sekundnega in nanosekundnega števca vsakič, ko v EL modulu števec obhodov zavzame vrednost 0, kateremu se prišteje konstantna vrednost 1/60.
Zasnova naprave zahteva, da je bila implementacija sekundnega števca narejena tako, da se ne sme povečevati vse dokler ni inicializiran na neničelno vrednost, kar pa mora storiti kontrolni sistem. Po uspešni inicializaciji se 32-bitni števec sekund poveča ob vsakem prelivu nanosekundnega števca, ki prosto teče s frekvenco urinega signala 20MHz.
Poleg samega štetja sekund in nanosekund je implementirana tudi sinhronizacija z zunanjim GPS (ang. Global Positioning System) sprejemnikom, ki periodično, točno vsako sekundo časovnemu krmilniku pošlje kratek električni pulz.
V modulu je implementirano samodejno popravljanje vrednosti nanosekundnega števca, kjer se izdajajo popravki vrednosti glede na vrednost ugotovljene napake. Le ta je izračunana kot absolutna razlika med pričakovano (konstantno) vrednostjo nanosekundnega števca, ki bi jo moral zavzemati ob prejemu signala iz GPS sprejemnika (1/s) ter njegovo dejansko vrednostjo.

Izračun vrednosti popravkov je dobljen po sledečih pravilih:

$$\text{napaka } < 2ms \Rightarrow \text{nanosekunde } (+/-)\ \frac{razlika}{2} \tag{5}$$

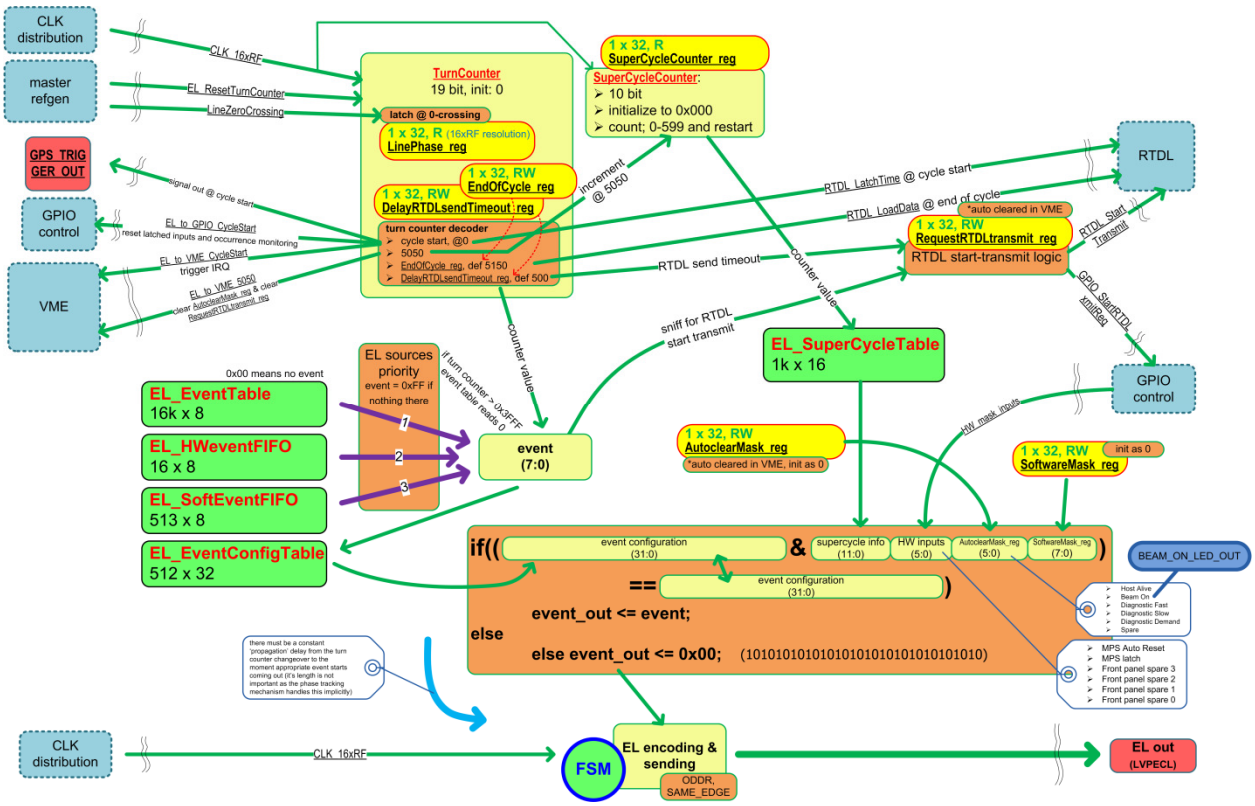
$$\text{napaka } > 2ms \Rightarrow \text{nanosekunde } (+/-)\ 1ms \tag{6}$$

$$\text{napaka } > 60ms \Rightarrow \text{nanosekunde} \leftarrow\!\!\!\rightarrow 0;\ \lceil \text{sekunde} \rceil \tag{7}$$

Implementacija zajema tudi možnost izgube GPS signala, kjer izvajanje popravkov dlje časa ni mogoče.V takem primeru celotna funkcionalnost nemoteno deluje, kjer večjih odstopanj ni pričakovati. Da bi se izognili napačnemu delovanju, kjer bi popravki lahko zavzemali velike vrednosti in bi tako povzročili preskoke nanosekundnega števca, je to preprečeno s pravilom 3, ki določa največji možni enkratni popravek 1 ms.

3.5 Modul EL

EL modul na sliki 3.9 predstavlja srce časovnega krmilnika. Je najbolj pomembna in kritična komponenta, ki s pošiljanjem dogodkov narekuje delovanje celotnega pospeševalnika. Poglavitna naloga modula je zagotoviti neprekinjen tok podatkov po dogodkovni povezavi, po kateri so zaporedoma razpošiljani posamezni dogodki ob začetku vsakega novega obhoda delcev v akumulacijskem obroču pospeševalnika. Pošiljanje posameznega dogodka traja 1µs, kjer mora biti v tem času poslana 32-bitna vrednost dogodka. Frekvenca pošiljanja dogodkov tako znaša 1MHz in odraža frekvenco obhodov delcev v akumulacijskem obroču pospeševalnika.

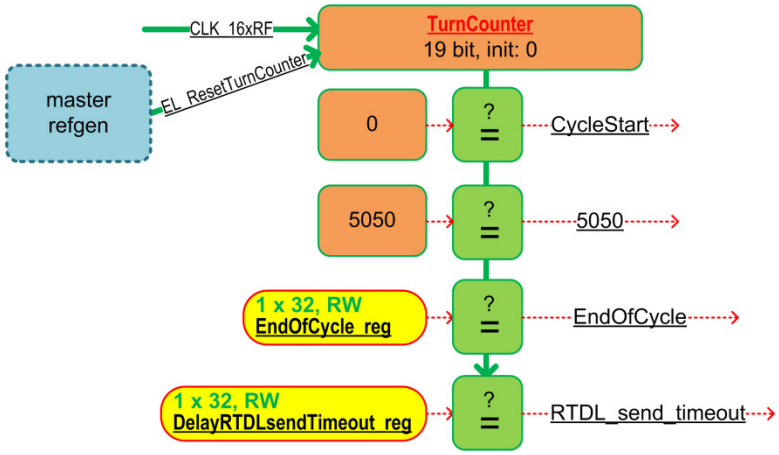


Slika 3.9 Arhitekturna zasnova EL modula.

Števec obhodov - TurnCounter

Jedro modula predstavlja števec obhodov (ang. Turn Counter), ki je implementiran kot 19-bitni inkrementalni števec, katerega vrednost je povečana ob vsaki pozitivni fronti CLK_16xRF urinega signala (16 MHz) in predstavlja obhode delcev v akumulacijskem obroču. Zgornjih 15 bitov (18:4) predstavlja cele obhode, spodnji štirje biti (3:0) pa predstavljajo vmesne deleže posameznega obhoda v korakih po 1/16 obhoda. Števec je periodično, sinhrono in neodvisno ponastavljen s strani *Master Reference Generator* modula - ponastavitev števca mora sovpadati s prehodom ničle omrežne napetosti. Neposredno je uporabljen za naslavljanje tabele EL_EventTable, ki vsebuje zaporedje dogodkov, ki bodo razposlani po EL povezavi.

Glede na vrednost števca obhodov so generirani sledeči signali:



Slika 3.10 Generiranje ključnih signalov.

- *CycleStart* – pozitivna fronta signala označuje začetek cikla, ob katerem se VME gostitelju izda prekinitvena zahteva, v GPIO modulu pa se ponastavijo zatiči, ki pomnijo stanje vhodno/izhodnih signalov, ki so bili sproženi v predhodnem ciklu. RTDL modul takrat shrani trenutni čas, ki bo uporabljen za razpošiljanje po RTDL povezavi. Prav tako pa je generiran tudi 100 µs širok signal potreben za proženeje zunanjega GPS sprejemnika, ki je uporabljen za časovno sinhronizacijo.
- *5050* - izvrši inkrement števca super ciklov (*SuperCycleCounter).* Hkrati je VME modulu izdana zahteva za ponastavitev registra, ki določa masko pri evaluaciji pogojev, ki določa pogoj ali sme biti trenutno izbrani dogodek poslan.
- *EndOfCycle* – RTDL modulu je izdana zahteva za osvežitev podatkov v RTDL tabeli.
- *RTDL_send_timeout* – zahteva za začetek prenosa po RTDL povezavi, če tega do sedaj ni zahteval že VME gostitelj.

Števec super ciklov

Števec super ciklov *(ang. Super Cycle Counter)* je 10-bitni števec, inkrementiran vsakič, ko števec obhodov (*TurnCounter)* doseže vrednost 5050. Deluje v modulo 600 načinu, kjer se vrednost ponastavi na 0 vsakič, ko doseže vrednost 599.

Števec je neposredno uporabljen za naslavljanje tabele *EL_SuperCycleTable*, katera priskrbi 12-bitno vrednost, ki je uporabljena kot del pogoja pri preverjanju ali sme biti določen dogodek poslan.
Z uporabo števca super ciklov je pridobljena raznolikost ciklov, kjer je vsak posamezen cikel (sestavljen iz 600 obhodov) vnaprej določen z vrednostmi mask v tabeli.

Izbiranje dogodkov

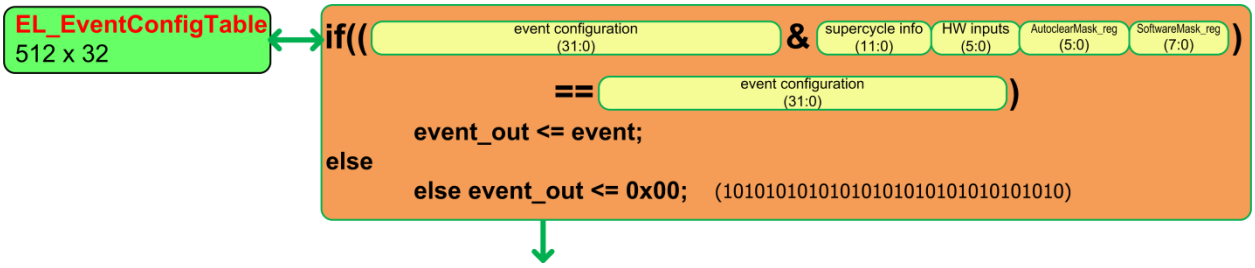
Razpošiljanje dogodkov po EL povezavi je v celoti doseženo s števcem obhodov, saj se ob vsakem inkrementu števca zagotovi nov dogodek, ki je lahko pridobljen iz sledečih virov:

- *EL_EventTable* pomnilna tabela, ima najvišjo prioriteto med viri. Dogodek pridobljen iz tabele je veljaven in uporabljen, če vsebuje neničelno vrednost. Ker je tabela manjša, kot naslovni prostor, ki ga lahko naslovi števec obhodov, je pri naslavljanju tabele vpeljana omejitev. Za vse naslove, ki so večji kot naslov zadnje lokacije tabele (0x3FFF), je vrnjena vrednost vedno enaka 0.
- *EL_HWEventFIFO* – FIFO pomnilni element z izvorom dogodkov v GPIO modulu. Vrednosti so pridobljene sekvenčno, kjer je upoštevano načelo, da je dogodek upoštevan le, če vsebuje neničelno vrednost. Nov dogodek je iz FIFO pomnilnega elementa vzet le v primeru če je bil ustrezno obravnavan predhodno prebran dogodek. Tako je zagotovljeno, da se dogodki iz FIFO pomnilnih elementov ne izgubljajo.
- *EL*_SoftEventFIFO – FIFO pomnilni element z izvorom dogodkov v VME modulu (najnižja prioriteta). Delovanje je enako kot pri *EL_HWEventFIFO* viru.

Komponenta, ki določa iz katerega od virov bo uporabljen dogodek je implementirana kot končni avtomat, ki ob vsakem inkrementu števca obhodov, sekvenčno preveri vse tri vire in z upoštevanjem prioritet pridobi ustrezno vrednost.
V primeru, ko nobeden od virov ne vsebuje veljavnega dogodka - *EL_EventTable* na trenutno naslovljeni lokaciji vsebuje vrednost 0, *EL_HWEventFIFO in EL_SoftEventFIFO* pa sta prazna, je uporabljena privzeta vrednost dogodka 0x00.

Validacija dogodkov

Vsakemu dogodku pripada tudi 32 bitna konfiguracijska vrednost, ki se nahaja v tabeli *EL_EventConfigTable* in določa pogoje pri katerih je določen dogodek lahko razposlan. Za naslavljanje tabele je uporabljena kar vrednost dogodka samega.



Slika 3.11 Validacija dogodkov.

Konfiguracijska vrednost je uporabljena pri maskiranju z vektorjem (slika 3.11), ki je sestavljen iz sledečih vrednosti:

- *SuperCycle info* – 12-bitna vrednost, pridobljena iz *EL_SuperCycleTable* tabele, kjer je za naslavljanje uporabljen *EL_SuperCycleCounter* števec.

- *HW inputs* – 6-bitna vrednost, ki vsebuje trenutno stanje vhodnih priključkov na tiskanini. Vrednost je pridobljena preko GPIO modula. Poglavitni uporabljeni priključki so povezani z MPS (ang. Machine Protection System) sistemom.

- *AutoclearMask_reg* – 6-bitni register implementiran v VME modulu. Vrednost, ki jo vanj vpiše kontrolni sistem se brezpogojno ponastavi na 0 vsakič, ko števec obhodov doseže vrednost 5050.

- *SoftwareMask_reg* – 8-bitni register implementiran v VME modulu. Maskirni register, ki je v celoti kontroliran s strani kontrolnega sistema.

Proces validacije kot rezultat nudi predhodno izbran dogodek, če konfiguracijska vrednost izpolne logično IN operacijo med konfiguracijsko vrednostjo dogodka in maskirnim vektorjem. Če ta pogoj ni izpolnjen, je rezultat privzeta vrednost dogodka 0x00, ki po končnem kodiranju na EL povezavi predstavlja signal z največjo frekvenco izmenjevanja med visokim in nizkim stanjem signala (0b10101010101010101010101010101010).

Kodiranje dogodkov

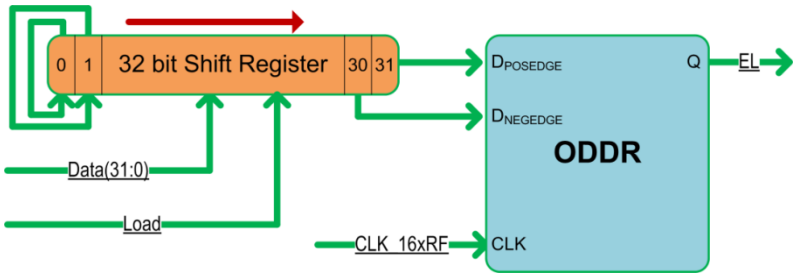
Po končanem izbiranju in validaciji dogodka, mora biti vrednost bi-fazno kodirana in po EL povezavi poslana z dvakratnikom frekvence urinega signala CLK_16xRF.
Kodiranje dogodkov je namesto s sprotnim izračunavanjem, izvedeno z uporabo LUT tabele (ang. Lookup Table), kjer so zapisane vse končne vrednosti dogodkov, za naslavljanje LUT tabele pa se uporablja kar vrednost dogodka, ki je bil izbran. Tabela kodiranih vrednosti je implementirana kot ROM pomnilnik.

Pošiljanje dogodkov

Implementacija pošiljanja dogodkov po EL povezavi zahteva, da so posamezni dogodki v neprestanem toku podatkov pravilno konkatenirani, kjer se mora začetek trenutnega dogodka fazno ujemati s koncem predhodno poslanega dogodka.
Na sliki 3.12 je prikazana komponenta, ki pošilja kodirano vrednost dogodka po EL povezavi. Vsebovan je ODDR zatič (ang. Output Double Data Rate Flip Flop), ki je realiziran kot samostojni primitivni element v FPGA vezju. Zatič ima dva podatkovna eno bitna vhoda, katerih vrednost se preslika na izhod ob pozitivni in negativni fronti urinega signala. Z uporabo takega principa je dosežena prej omenjena zahtevana dvakratna frekvenca pošiljanja vrednosti dogodkov, kjer je 32-bitna vrednost preslikana na izhod v šestnajstih urinih periodah.
Komponenta poleg posebnega zatiča vsebuje tudi 32-bitni pomikalni register, ki hrani vrednost namenjeno pošiljanju in jo ob vsaki pozitivni fronti urinega signala zamakne za dve mesti. Kot varovalka za ohranjanje pravilnega nemotenega toka vrednosti na izhodu zatiča je pomikalni register implementiran tako, da se bita 0 in 1 vedno preslikujeta sama vase (slika 3.12). S tem je doseženo, da po vpisu vrednosti privzetega dogodka v pomikalni register, komponenta na izhodu vedno proizvede izmenljiv vzorec logičnih vrednosti 1 in 0. Tak primer nastopi kadar validacija

dogodka ni uspela in je poslana privzeta vrednost dogodka enaka nizu vrednosti: 0b10101010101010101010101010101010.

Za pravilno razpošiljanje vrednosti preko zatiča, mora nadrejena komponenta novo vrednost v pomikalni register vpisati vsakih 16 urinih period.



Slika 3.12 Pošiljanje vrednosti z uporabo DDR zatiča.

Pomembna zahteva celotne obdelave dogodkov, od pridobitve do samega pošiljanja je, da mora biti zakasnitev (ang. counter to event propagation delay) od inkrementa števca obhodov do samega pošiljanja vedno enaka, neodvisno od tega iz katerega vira je bil dogodek pridobljen.

Izjema pri pošiljanju začetnega dogodka

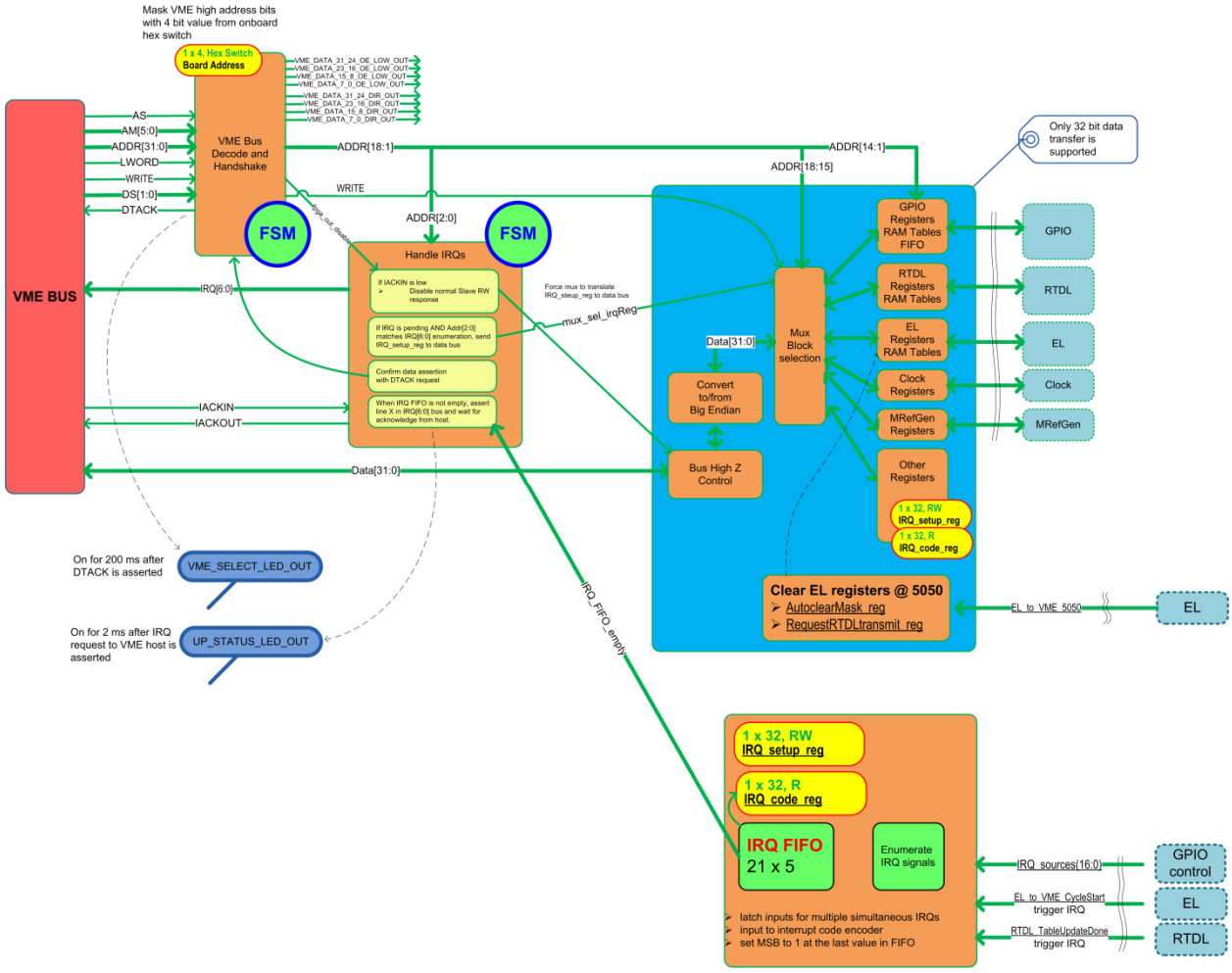
Za razliko od vseh ostalih dogodkov je začetni dogodek *CycleStart*, ki označuje začetek vsakega novega obhoda delcev, poseben in obravnavan z izjemo. Vzrok je obstoječa implementacija celotne infrastrukture pospeševalnika, ki zahteva, da je ta dogodek poslan 1/16 obhoda vnaprej. Rešitev je bila dosežena tako, da so vsi ostali dogodki z ustreznim kodiranjem zakasnjeni tako, da sta prva dva bita sicer 32-bitne vrednosti dogodka neuporabljena in imata pri vseh dogodkih enako privzeto vrednost (vsak bit vrednosti predstavlja 1/32 obhoda), ki predstavlja razširjen start bit. Ker sta pri *CycleStart* dogodku uporabljena tudi začetna bita je tako dosežena efektivna prednost pred ostalimi dogodki in je sam dogodek poslan 1/16 obhoda prej, kot ostali dogodki.

Merjenje faze

Kot že opisano, celoten sistem teži k temu, da se prehod omrežne napetosti skozi ničlo ujema s trenutkom, ko števec obhodov zavzema vrednost 5050. To je doseženo z uglasitvijo modula *Master Reference Generator*, ki periodično generira signal za ponastavitev števca obhodov. Za popolno sinhronizacijo, mora imeti kontrolni sistem za izdajanje popravkov faze signala, ki ga generira modul *Master Reference Generator*, na voljo informacijo o vrednosti števca obhodov v trenutku, ko omrežna napetost prečka ničlo. Vrednost števca obhodov je pomnjena ob vsakem prehodu omrežne napetosti skozi ničlo (prehod je zaznan v modulu *Master Reference Generator*), ter je preko VME modula na voljo kontrolnemu sistemu, kjer je nato izračunan popravek glede na odstopanje med dejasnsko vrednostjo in vrednostjo 5050.

3.6 Modul VME

Modul VME (ang. VERSA-module Europe) na sliki 3.13 predstavlja edini komunikacijski vmesnik preko katerega lahko kontrolni sistem upravlja s časovnim krmilnikom. Glavna naloga je nuditi popolno podporo standardiziranemu protokolu VME, ki je uporabljen v sistemu VME. Gostitelj VME preko modula upravlja s časovnim krmilnikom kot pomnilniškim prostorom, ki sestoji iz množice registrov in pomnilniških tabel.



Slika 3.13 Arhitekturna zasnova VME modula.

VME sistem

Arhitekturni koncept VME sistema, oziroma VME vodil temelji na VERSA vodilu [**5**], ki ga je leta 1981 v sodelovanju z drugimi podjetji razvilo podjetje Motorola. Vodilo je bilo sprva zasnovano kot razširitveno vodilo, ki so ga uporabljali mikrokontrolerji družine Motorola 68000, kasneje pa je bilo izpopolnjeno in je zaradi preproste zasnove in robustnosti kmalu postalo industrijski standard, ki je vsesplošno uporabljen.

Uporabljen je princip skupnih vodil, ki ga sestavljajo 32-bitno naslovno vodilo, 32-bitno dvosmerno podatkovno vodilo, 7-bitno prekinitveno vodilo in 9-bitno kontrolno vodilo. Celoten sistem pa običajno sestavlja CPU gostiteljska kartica (ang. VME Host) ter več namenskih kartic (ang. VME slave), s katerimi CPU kartica komunicira preko sistema skupnih vodil. Taka zasnova ni pravilo, saj so možne tudi konfiguracije sistema, ki vsebuje več gostiteljskih CPU kartic ali pa samo namenske kartice, kjer se CPU gostitelj nahaja v ločenem VME ohišju. Poleg gostiteljske CPU in namenskih kartic, mora sistem obvezno vsebovati tudi arbitra, ki dejansko upravlja s skupnimi vodili, ter določa katera izmed kartic ima v določenem trenutku pravico do njihove uporabe. Arbitrova naloga je imeti popolen nadzor nad vodili, dodeljevanje in sproščanje vodil in upravljanje z vodili med postopkom izdajanja in strežbe prekinitev. Poseganje arbitra v delovanje sistema je za posamezne kartice popolnoma transparentno in neopazno, saj je vsaka njegova akcija oziroma ukrep karticam videm kot ukrep, ki ga izdaja CPU gostiteljska kartica. Med najbolj opaznimi akcijami velja omeniti izdajanje čakalnih ciklov, ki jih lahko zahteva namenska kartica za obdobje, ko še ni pripravljena na komunikacijo. Arbiter je prav tako odgovoren za pravilno obravnavanje napak, ki nastopijo pri komunikaciji preko vodil.

Protokol VME vodila

Specifikacijo komunikacijskega protokola uporabljenega v VME sistemih podaja organizacija VITA z določilom ANSI/VITA 1-1994 (R2002) .

Vsak prenos preko vodil se izvaja z rokovanjem, kjer iniciator prenosa sprva zaprosi arbitra za vodilo z držanjem ene od štirih BR (ang. Bus Request Line) linije v nizkem stanju. Na to zahtevo se odzove arbiter, ki ciklično vzorči vse 4 BR linije z namenom, da določi katera kartica je zaprosila prevzem vodila in v primeru da je teh več, odloči katera izmed njih bo prevladala. Arbiter pri vzorčenju in odločitvi kateremu iniciatorju prenosa bo dodeljeno vodilo, uporablja

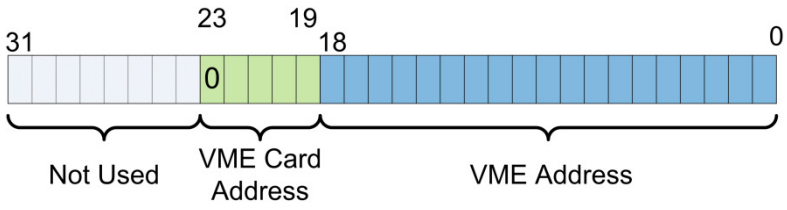
princip prioritet, kjer ima linija BR3 največjo težo. Po končani arbitraži za vodilo, iniciator, ki mu je uspelo pridobiti vodilo, le to oznani s potrditvijo tako, da postavi BBSY (ang. Bus Busy) signal v nizko stanje.

V primeru, ko hoče iniciator, trenutni gospodar, izvršiti pisalni dostop do določene lokacije, postavi željene vrednosti na naslovno in podatkovno vodilo, hkrati pa ustrezno nastavi tudi signale na vodilu, ki označuje naslovne modifikatorje (ang. Address Modifier Bus). Ti določajo kakšne vrste prenos se bo izvršil:

- Blokovni prenos v priviligiranem načinu
- Enkratni podatkovni prenos v priviligiranem načinu
- Enkratni podatkovni prenos v navadnem načinu
- ...

Sledi postavitev signala AS (ang. Address Strobe) v nizko stanje kar označuje veljavno vrednost na podatkovnem vodilu. V naslednjem trenutku pa sta v nizko stanje postavljena tudi DS (ang. Data Strobe) signala, ki označujeta veljavno vrednost na podatkovnem vodilu. Širino podatkovnega prenosa (8, 16, 24 ali 32-bitni prenos) označuje kombinacija DS signalov (2 bita) ter LWORD signala. Pri najbolj pogosto uporabljenem 32-bitnem prenosu so vsi trije signali postavljeni nizko. Ta korak označuje konec iniciatorjevega začetnega cikla, kjer le ta sedaj čaka na odgovor namenske kartice.

Ker celotni VME sistem uporablja sistem skupnih vodil, vse kartice vidijo enake vrednosti na vseh vodilih (slika 3.14), zato morajo ločiti kateremu naslovnemu prostoru pripadajo. Najpogosteje uporabljena rešitev je implementacija šestnajstiško kodiranega stikala na tiskanini namenske kartice, ki preko štirih izhodov podaja nastavljivo šestnajstiško vrednost, ki je uporabljena pri dekodiranju naslova.

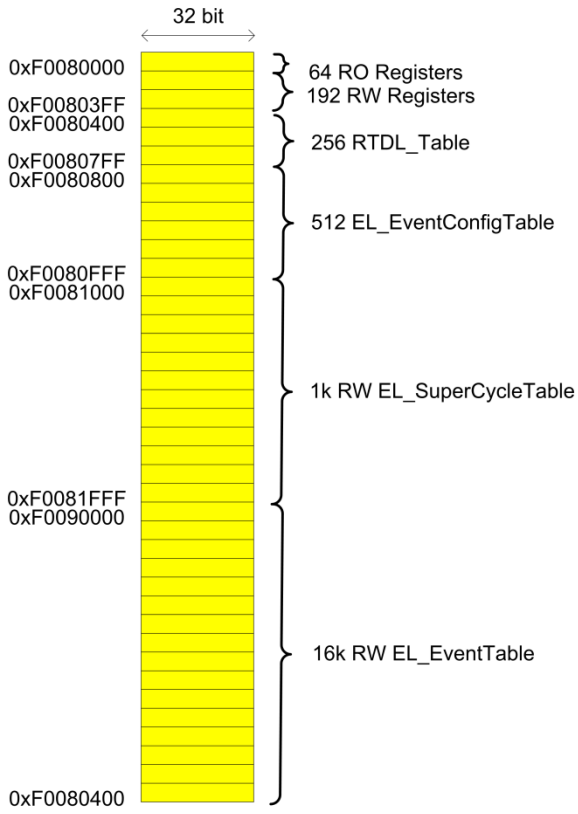


Slika 3.14 Delitev naslovnega vodila.

Časovni krmilnik je načrtovan za delovanje v 24-bitnem naslovnem prostoru (A24), kar pomeni da je od 32 bitov naslovnega vodila upoštevanih le 24. Zgornjih 5 bitov je uporabljenih za določitev naslova kartice, preostalih 19 bitov pa predstavlja dejanski naslovni prostor kartice.
Naloga VME modula časovnega krmilnika je neprestano vzorčiti naslovno vodilo in primerjati vrednost nastavljeno na šestnajstiško kodiranem stikalu z biti 23:19 naslovnega vodila. Ko se vrednosti ujemajo, se kartica odzove glede na vrsto prenosa z ustreznim krmiljenjem kontrolnih signalov.

Potek nadaljevanja VME komunikacije določa, da mora naslovljena kartica oddati potrditev s postavitvijo skupnega potrditvenega signala DTACK (ang. Data Transfer Acknowledge) v nizko stanje. S tem je iniciator prenosa, ki ves čas čaka na negativno fronto DTACK signala, obveščen da je naslovil veljavno pomnilniško lokacijo in lahko zaključi trenutni cikel prenosa. Zaključek je izveden s ponastavitvijo signalov AS, DS, LWORD kontrolnega vodila v visoko stanje. Prav tako pa mora tudi naslovljena kartica ponastaviti DTACK signal v visoko stanje, s čimer je cikel prenosa v celoti zaključen.
V primeru, da je bil ves čas izvrševanja v teku pisalni dostop (iniciator zapisuje vrednost v izbrano pomnilniško lokacijo) mora naslovljena kartica preklopiti svoje podatkovno vodilo v visoko impedančno stanje – vhodni način ter shraniti vrednost, prebrano iz dvosmernega podatkovnega vodila.
VME protokol omogoča tudi blokovne prenose podatkov, kjer je z enim dostopom, zaporedno prenesen blok podatkov. Uporabe takega načina prenosa časovni krmilnik ne zahteva in podpora zanj ni implementirana.

Preslikava pomnilnega prostora

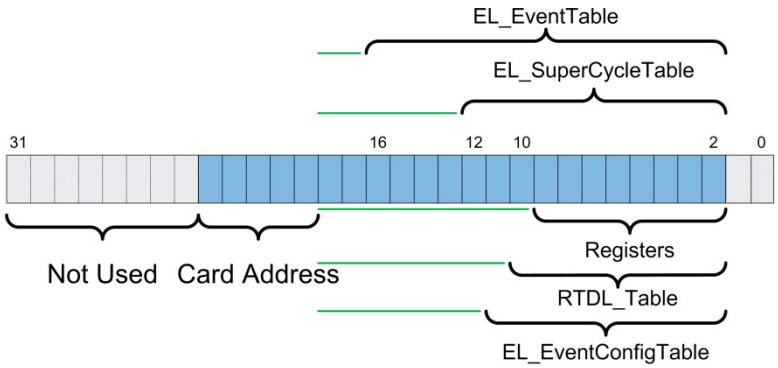


Slika 3.15 Struktura pomnilniškega prostora.

Poglavitna naloga VME modula je preslikati naslovni prostor kot ga obravnava VME gostitelj v naslovni prostor časovnega krmilnika, ki zajema posamezne registre in pomnilniške tabele časovnega krmilnika (slika 3.15).

Preslikava pomnilniškega prostora je implementirana kot samostojna komponenta, ki na vhodu prejema naslovno vodilo, izhod pa je povezan neposredno z dvosmernim podatkovnim VME vodilom. Smer komunikacije je podana s signaloma *regWrEn_i* in *regRdEn_i*, ki določata orientacijo podatkovnega vodila med visokim impedančnim stanjem med operacijo pisanja (VME gostitelj zapisuje vrednosti v registre ali tabele), ter aktivno vodenim podatkovnim vodilom (VME gostitelj bere vrednosti registrov ali tabel). Signala za orientacijo vodila sta priskrbljena iz komponente, ki upravljanja z rokovanjem vodila in je implementirana z uporabo avtomata FSM (ang. Finite State Machine).

Preslikovanje je razdeljeno na dva ločena dela in sicer preslikovanje registrov in preslikovanje tabel. Ker je arhitektura časovnega krmilnika poleg 256 registrov zahtevala tudi implementacijo štirih različno velikih tabel (256, 512, 1024 in 16384 lokacij) je to poenostavilo interno določanje pomnilniških sektorjev.



Slika 3.16 Segmentiranje pomnilniškega prostora.

V podanem izseku kode je prikazan prikazan preprost kodirnik, ki pretvarja vrednosti z naslovnega vodila kot je prikazano na sliki 3.16 v interno enumeracijo, ki je uporabljena pri segmentiranju pomnilnika.

```
s_addrDecMux <=
  "101" when vmeAddr_i(18 downto 16)="001" else -- EL_EventTable
  "100" when vmeAddr_i(18 downto 12)="0000001" else -- EL_SuperCycleTable
  "011" when vmeAddr_i(18 downto 11)="00000001" else --EL_EventConfigTable
  "010" when vmeAddr_i(18 downto 10)="000000001" else  -- RTDL_Table
  "001"; -- Registers
```

Podatkovna vodila vseh tabel in izhod multiplekserja registrov so vodena v manjši multiplekser, kateremu je na izbirni vhod pripeljana vrednost zgoraj opisanega kodirnika. Z izhodom multiplekserja neposredno upravlja samostojen proces, ki določa ali se na podatkovno vodilo zapiše izhod multiplekserja ali pa se podatkovno vodilo postavi v visoko impedančno stanje.

Implementacija registrov

Registri so predstavljeni kot polje (ang. array) 256 32-bitnih vektorjev, kjer začetnih 64 polj predstavlja samo bralne registre, preostalih 192 polj pa predstavlja bralno-pisalne registre. Bralni registri omogočajo VME gostitelju branje vrednosti, ki izvirajo iz različnih komponent časovnega krmilnika. Pred samo priključitvijo posamezne vrednosti iz drugih komponent v register je potrebno upoštevati prečkanje različne urine domene in signale večkratno vzorčiti za rabo v urini domeni, ki jo uporablja VME modul. Bralno-pisalni registri omogočajo VME gostitelju zapisovanje vrednosti v register, ki je nato priključen posamezni komponenti, hkrati pa VME gostitelju omogoča branje vrednosti, ki jo je pravkar zapisal.

Implementacija pomnilnih tabel

Vse pomnilne tabele, ki jih uporabljajo drugi moduli, so realizirane z DPRAM (ang. Dual Port RAM) primitivnimi elementi, kjer je ena stran naslovljena z naslovnim vodilom VME modula, druga stran pa je povezana z EL in RTDL moduloma od koder izvirajo druga naslovna vodila. Vsi signali so v DPRAM elemente peljani neposredno, z dodatno logiko pa so krmiljeni le signali, ki omogočajo pisanje v pomnilnik s strani VME modula.

```
s_ELEventConfigTableVMEWe <= '1' when (s_addrDecMux = "011" and s_WrEn_1 = '1') else
'0';
```

Proženje prekinitev

Mehanizem prekinitev je glavni vir obveščanja VME gostitelja o dogodkih, ki se zgodijo v časovnem krmilniku na katere se mora programsko odzvati gostitelj oziroma kontrolni sistem.
Vsaka sprožena prekinitev od CPU enote zahteva določeno rutino, ki mora biti izvedena v okviru ISR (ang. Interrupt Service Routine) rutine, kjer CPU shrani trenutni kontekst v katerem se nahaja, nato začne s strežbo sprejete prekinitvene zahteve ter po obnovitvi konteksta nadaljuje z izvajanjem kjer je bilo le to prekinjeno.

Zasnova časovnega krmilnika zahteva, da CPU kartica ob sprejeti prekinitvi, iz vnaprej znanega naslova zaporedno bere 32-bitne vrednosti, vse dokler prebrana vrednost nima visoko postavljenega MSB bita.

V časovnem krmilniku je na voljo 19 prekinitvenih signalov, od katerih jih 17 izvira iz GPIO modula. Dodatna dva (*EL_to_VME_CycleStart* in *RTDL_TableUpdateDone*) signala izvirata iz EL modula in označujeta začetek pospeševalniškega cikla in končano osveževanje podatkov v RTDL tabeli.

Vsi signali so vodeni do svoje instance komponente (sinhroni SR zatič), ki ob pozitivni fronti vhodnega signala sig_i, svoj izhod postavi v visoko stanje, le ta pa je ponastavljen šele z visokim stanjem na clr_i vhodu. Tak način omogoča beleženje le enega prehoda dokler le ta ni bil obravnavan, četudi se je na določenemu signalu v nekem časovnem obdobju pojavilo več pozitivnih front. S tem je preprečeno nepotrebno izdajanje novih prekinitvenih zahtev pred obravnavo trenutno izdane zahtve.

Vsi izhodi SR zatičev so v nadaljevanju obravnavani v preslikovalni komponenti, ki posamezne signale zatičev pretvori v enolične enumeracije. Tako je vsak signal, če je sprožen, predstavljen z unikatno 5 bitno enumeracijo. Preslikovanje signalov se odvija z uporabo prioritet, s čimer je ob hkratni postavitvi večih signalov doseženo, da se v vsakem urinem ciklu na izhodu komponente pojavi druga enumeracija. SR zatiči so po postopku enumeracije ponastavljeni in tako pripravljeni na sprejemanje novih pozitivnih front. V primeru, ko ni postavljen noben zatič, je izhod preslikovalne komponete enak 0. Izhod preslikovalne komponente je v nadaljevanju voden do FIFO pomnilnega elementa (IRQ_FIFO), v katerega je vstavljena vsaka neničelna enumeracija signalov.

Izdajanje prekinitvenih zahtev VME gostitelju se izvede ob vsaki negativni fronti Empty kontrolnega signala IRQ_FIFO pomnilnega elementa, s katerim je oznanjeno, da FIFO element vsebuje vsaj eno neprebrano vrednost.

Časovni krmilnik takrat v nizko stanje postavi sedmo linijo prekinitvenega vodila, ki CPU sporoči, da je bila izdana prekinitvena zahteva z najvišjo prioriteto. V nadaljevanju mora časovni krmilnik vzorčiti IACKIN (ang. Inetrrupt Acknowledge In) signal, kjer negativna fronta označuje sprejetje prekinitvene zahteve s strani CPU. Nato časovni krmilnik vzorči spodnje tri bite naslovnega vodila, na katerega CPU sekvenčno izdaja vrednosti od 1 do 7 ter čaka na odziv v obliki negiranega DTACK signala. Časovni krmilnik mora DTACK signal postaviti v nizko

stanje takoj, ko se na naslovnem vodilu pojavi vrednost 7, kar pomeni potrditev izdaje prekinitvene zahteve številka 7. V naslednjem trenutku CPU pričakuje vrednost na podatkovnem vodilu, ki predstavlja prekinitveni vektor za katerega želi časovni krmilnik, da ga CPU obravnava. Prekinitveni vektor je popolnoma nastavljiv in je predstavljen kot navaden bralno pisalni register katerega vrednost lahko kontrolni sistem spreminja z normalnimi dostopi. Tako je rokovanje ob prekinitveni zahtevi lahko zaključeno, naloga obeh strani je le, da ponastavita vse signale, ki so bili uporabljeni. CPU zatem začne izvajati ISR rutino, ki pripada sprejetemu prekinitvenemu vektorju in mora biti predhodno naložena v delovni pomnilnik in ustrezno registrirana v operacijskemu sistemu.

Kot že omenjeno, se v ISR rutini izvaja branje iz vnaprej znane pomnilniške lokacije v časovnem krmilniku, ki je CPU vidna kot bralni register, dejansko pa je to IRQ_FIFO pomnilni element. Branje vrednosti iz te lokacije se izvaja vse dokler 32-ti bit prebrane vrednosti ni postavljen v visoko stanje. Tako CPU prebere vse vrednosti, ki so se do takrat nabrale v FIFO elementu, ki po končanem branju ostane prazen, "*Empty*" kontrolni signal se povrne v visoko stanje in celoten cikel izdajanja prekinitvenih zahtev se lahko začne znova. V VME modulu časovnega krmilnika je notifikacija o praznem IRQ_FIFO elementu implementirana na sledeč način:

```
irqFIFODataOut_o(31)    <= s_fifoEmpty;
irqFIFODataOut_o(30 downto 5) <= (others=>'0');
irqFIFODataOut_o(4 downto 0) <= s_fifoDataOut;
```

Ker se zaradi zasnove časovnega krmilnika proženje prekinitvenih zahtev začne takoj po priklopu FPGA vezja na napajanje in končani njegovi konfiguraciji – zaradi signala *EL_to_VME_CycleStart* lahko pride do nepravilnega delovanja sistema.

Preden je VME gostitelj sposoben obravnavati prekinitvene zahteve mora opraviti osnovno konfiguracijo, kot tudi naložiti operacijski sistem ter v pomnilnik naložiti vse ISR sklope, za kar pa je potrebno ogromno časa – reda 90 sekund. V tem času časovni krmilnik že izda prekinitveno zahtevo, ki pa ni obravnavana, CPU pa hkrati vidi prekinitveno zahtevo za katero še ni naložen ustrezen ISR sklop. Da do takih slučajev ne pride, sta v implementaciji časovnega krmilnika vgrajeni dve varovalki:

- Proženje prekinitvenih zahtev je onemogočeno vse dokler VME gostitelj tega eksplicitno ne zahteva z vpisom določene vrednosti v poseben register.
- Prekinitvena zahteva je lahko izdana ponovno, če prejšnja ni bila obravnavana v časovnem obdobju največ 4 ms.

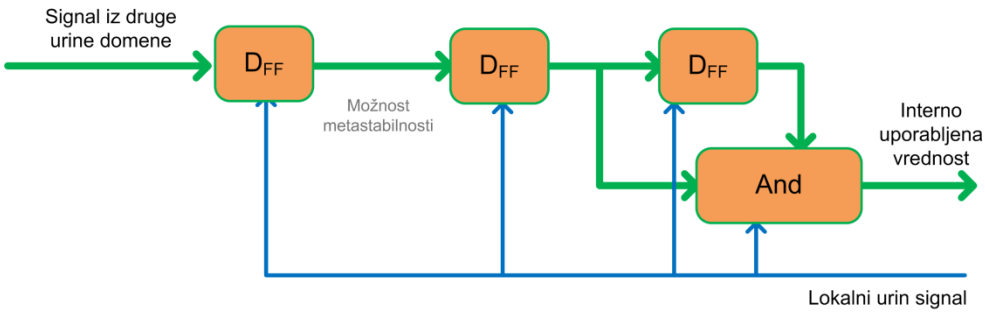
3.7 Uporabljene rešitve za odpravljanje napak in nekonsistentnosti

Sinhroni pomilni elementi t.i. zatiči, so lahko podvrženi pojavu metastabilnosti [**18**]. V normalnem delovanju, kjer so spoštovana vsa pravila za delovanje, se njihov izhod nahaja le v dveh točno določenih stanjih, 0 ali 1. V posebnih pogojih, pa lahko zatič deluje nepravilno in njegov izhod zavzema vrednosti ki ne pripadajo ne logični '0' ne logični '1'. Obdobje takega obnašanja je lahko daljše kot je trajanje enega urinega cikla.

Najpogosteje smo s takšno napako soočeni kadar kršimo časovne omejitve zatiča (ang. setup time, hold time), ki podajajo minimalne čase pred in po fronti urinega signala, kjer mora biti vhodni signal zatiča popolnoma stabilen.

V praksi so za tako vedenje najpogosteje odgovorne različne urine domene, kjer je vhod zatiča vzorčen ob fronti urinega signala, ki ima različno frekvenco ali fazo kot urin signal s katerim je voden izvor signala.

Posamezni signali ali vodila, ki so priključeni na vhod posamezne komponente ne smejo biti neposredno uporabljeni, če le ti izvirajo iz urine domene, ki je različna od trenutno uporabljene. Namesto neposredne uporabe, morajo biti sprva vodeni čez cevovod vsaj treh zaporedno vezanih zatičev, ki delujejo v lokalni urini domeni.



Slika 3.17 Večkratno vzorčenje vhodnega signala.

Slika 3.17 prikazuje sestavo komponente, ki je uporabljena pri vsakokratni uporabi signalov z izvorom v drugi urini domeni. Prvi zatič je v celoti podvržen pojavu metastabilnih vrednosti, ki bi lahko povzročile nepravilno delovanje sklopa, sledeča dva zatiča pa možnost tega pojava izničita. Kot dodatna varovalka je poleg samih zatičev realiziran tudi logični primerjalnik, kjer so primerjane izhodne vrednosti drugega in tretjega zatiča. Izhod je veljaven le v primeru, ko se izhoda zadnjih dveh zatičev ujemata. Princip je implementiran kot samostojna komponenta in je skozi celotno implementacijo množično uporabljena.

Implementacija komponente za večkratno vzorčenje signala z dodatnim primerjalnikom vrednosti je podana kot:

```
entity SigInputSampleAndRisingEdgeDetection is
port (
        sig_i: in std_logic;
        clk_i: in std_logic;
        sig_o: out std_logic
);
end SigInputSampleAndRisingEdgeDetection;

architecture RTL of SigInputSampleAndRisingEdgeDetection is
        signal s_prev: std_logic_vector(2 downto 0);
begin

p_syncSample : process(clk_i)
begin
        if rising_edge(clk_i) then
                s_prev<= s_prev(1 downto 0) & sig_i;
                sig_o <= not s_prev(2) and s_prev(1);
        end if;
end process p_syncSample;

end RTL;
```

Drugi načini reševanja prečkanja različnih urinih domen

Implementacija cevovoda za izničenje možnosti pojava metastabilnih stanj zatičev je odlična rešitev v primeru, ko imamo opravka s posameznimi signali ali ozkimi vodili.
Arhitektura naprave pa je v veliko primerih privzela uporabo širokih vodil med komponentami, kjer je implementacija cevovoda predstavljala veliko oviro pri doseganju željenih hitrosti delovanja. Eden izmed razlogov je prostorska potratnost, saj za pravilno uporabo 32-bitnega vodila potrebujemo kar 96 D zatičev. Za doseganje minimalne frekvence urinega signala s katerim delujejo posamezni sklopi in uporabljajo široka vodila, je najprimernejša rešitev uporaba strojnih FIFO pomnilnih elementov (slika 3.18).

Slika 3.18 Premostitev različnih urinih domen s FIFO elementom.

FIFO elementi so v FPGA vezju zgrajeni z uporabo dvo-stranskega blokovnega pomnilnika DPRAM (ang. Dual Port RAM), kjer vsaka stran upravlja z elementom popolnoma samostojno, v svoji urini domeni. Tako element predstavlja idealno rešitev, ko smo soočeni s širokim vodilom, saj sama implementacija elementa poleg ožičenja ne skalira enakomerno z naraščanjem širine vodila, saj je v večini primerov uporabljen enak blokovni pomnilnik, le izkoriščenost je večja ali manjša.
Poleg zgoraj opisanih načinov premoščanja različnih urinih domen, pa je uveljavljena še rešitev z uporabo rokovanja.
Podatkovno vodilo je v takem načinu vodeno od izvora neposredno na vhod komponente oziroma gradnikov kjer ga želimo uporabiti, kontrolni signali pa so vodeni skozi predhodno opisan cevovod. Oddajna stran mora najprej oddati podatke na vodilo, zatem s kontrolnimi signali sporočiti sprejemnemu delu, da so podatki pripravljeni in počakati na potrditev da so bili podatki prebrani. Pri takšnem načinu so večkratno vzorčeni signali samo signali za rokovanje, s katerimi zagotovimo pravilno delovanje in hkrati popolnoma pravilno branje samih podatkov, kajti le ti se med obdobjem rokovanja ne smejo spreminjati. Takšna rešitev je dokaj preprosta, vendar zahteva

dodatno logiko za izvedbo rokovanja in v implementaciji časovnega krmilnika ni bila uporabljena.

3.8 Testiranje

Ker med implementacijo naprave v podjetju nismo imeli na voljo dejanske strojne opreme – tiskanine časovnega krmilnika, je razvoj v celoti potekal s pomočjo simulacij VHDL kode. Posamezni moduli in komponente, so bili temeljito preizkušeni z uporabo simulacijskega okolja SystemC. Za razliko od običajnega testiranja s testnimi procedurami napisanimi v jeziku VHDL (ang. testbench) to okolje omogoča neposredno uporabo VHDL konstruktov v programskem jeziku C++, kjer so le ti predstavljeni kot običajni objekti. Takšen način je omogočil testiranje komponent in celotnega sistema z uporabo standardnega C++ jezika kar je zagotavljalo mnogo lažje in hitrejše pisanje testnih procedur, hkrati pa so se nad zahtevnejšimi vezji simulacije izvajajo mnogo hitreje.

Za preizkušanje manjših in prepostih komponent sem namesto SystemC simulacije uporabljal tudi simuliranje z uporabo skriptnega jezika Tcl. Ta ne zahteva nobenega testnega sistema in se simulacija izvaja na podlagi skript kjer so zapisani stimulatorji posameznih signalov.

Po končanem razvoju naprave, je preizkus delovanja potekal v razvojnem laboratoriju pospeševalnika SNS, kjer je bil krmilnik priklopljen na obstoječ pomožni sistem sprejemnikov, ki je bil replika pravega sistema. Zanimiv je bil rezultat prvih testov, kjer je že na začetku deloval večji del naprave. Med samim preizkušanjem je bilo odkritih nekaj hroščev in napak, ki pa so bile sproti odpravljene. Največ težav je med preizkusom povzročal RTDL modul saj diagnostični sprejemnik ni uspel razpoznati podatkov, ki so bili poslani po povezavi. Po izčrpnem pregledu in podrobni analizi se je izkazalo, da je bil vzrok napak prav v diagnostičnem RTDL sprejemniku, katerega implementacija ni popolnoma spoštovala specifikacij RTDL podatkov.

Po končanem dvotedenskem obisku laboratorija je naprava delovala v celoti, z izjemo modula VME, za katerega je bilo sprva predvideno, da ga bodo implementirali razvijalci inštituta vendar je bila kasneje sprejeta odločitev, da to v nadaljevanju naredim jaz. Podobno kot pri razvoju RTDL modula se je veliko težav pojavilo tudi pri implementaciji VME modula. Vzrok težav je bil v napaki na tiskanini krmilnika, kjer so napačno delovala vezja, ki so ločila FPGA priključke

od podatkovnega vodila VME. Vezja so večino časa ohranjala konstantno logično vrednost na izhodih neglede na stanje vhodov, zaradi česar je VME gostitelj vedno prebral napačne vrednosti registrov krmilnika. Ta napaka sicer ni onemogočila implementacije podpore protokolu, ki ga uporablja sistem VME je pa zelo otežila razvoj podpori prekinitvam. Kljub vsem težavam je bil po dolgotrajnem iskanju vzroka napak in testiranju tudi ta modul uspešno implementiran.

4. Sklepne ugotovitve

Arhitekturna zasnova in implementacija novega časovnega krmilnika je bil svojevrsten izziv, saj je to ključna komponenta pospeševalnika delcev, ki mora delovati zanesljivo in brez napak. Zato je bilo doseči zastavljeni cilj zelo zahtevno delo. Načrtovanje in implementacija vezij z uporabo FPGA tehnologije neprimerno težja naloga v primerjavi z običajnim načinom programiranja. Vsa posamezna vezja delujejo popolnoma vzporedno kar zelo otežuje testiranje in preverjanje funkcionalnosti sistema.

Pri implementaciji naprave so bile uporabljene napredne in robustne metode za reševanje problemov, kot je princip DDS, hitri prenosi podatkov z uporabo ODDR zatičev in učinkovito prečkanje različnih urinih domen med posameznimi komponentami v napravi.

Naleteli smo na napake in hrošče, ki pa so bile hitro odpravljene in niso zaustavljale razvoja celotnega projekta. Pridobljena so bila nova znanja, s stališča snovanja sistema in v povezavi z metastabilnimi stanji katerim so podvržena logična vezja. Posebno pomembno je dejstvo, da se skozi celoten razvoj zasnova časovnega krmilnika kljub popravkom ni nikoli spremenila. Pri simuliranju posameznih sklopov, kot tudi celotnega krmilnika je bilo spoznavanje in uporaba naprednih simulacijskih metod in orodij ključnega pomena.

Rezultat skoraj enoletnega projekta je naprava, ki v celoti izpolnjuje zahteve naročnika, hkrati pa je bil zaradi uspešnega zaključka pridobljen tudi nov projekt, ki vključuje dodatni razvoj naprave. Podjetje Cosylab d.d. bo tako v celoti razvilo nov časovni krmilnik za pospeševalnik delcev SNS, kar v začetku sicer ni bilo predvideno.

5. Literatura

[1] (2009) SNS. ORNL Neutron Sciences. Dostopno na: http://www.sns.gov/

[2] (2009) Virtex-5 FPGA Family. Dostopno na: http://www.xilinx.com/products/virtex5/index.htm

[3] (2009) vxWorks. Dostopno na: http://www.windriver.com/

[4] (2009) Brookhaven National Laboratory. Dostopno na: http://www.bnl.gov/world/

[5] (2009) VITA. Dostopno na: http://www.vita.com/

[6] J. Dedic: ESS - reusability of the SNS environment, interno poročilo, Cosylab d.d, 2009

[7] (2009) Precision Time Protocol. Dostopno na: http://en.wikipedia.org/wiki/Precision_Time_Protocol

[8] (2009) Clock Data Recovery. Dostopno na: http://en.wikipedia.org/wiki/Clock_recovery

[9] (2009) Frequency Shift Keying. Dostopno na: http://en.wikipedia.org/wiki/Frequency-shift_keying

[10] J. Dedic: *New Timing Master Design implementation*, interna dokumentacija, Cosylab d.d., 2009

[11] (2009) EPICS. Dostopno na: http://www.aps.anl.gov/epics/

[12] (2009) Aldec Inc. Active-HDL. Dostopno na: http://www.aldec.com/ActiveHDL/

[13] (2009) Xilinx ISE Design Suite. Dostopno na: http://www.xilinx.com/tools/webpack.htm

[14] (2009) Xilinx ChipScope Pro. Dostopno na:
http://www.xilinx.com/tools/cspro.htm

[15] (2009) Tcl Developer Xchange. Dostopno na:
http://www.tcl.tk/

[16] (2009) SystemC Simulation. Dostopno na:
http://www.systemc-simulation.com/

[17] (2009) Direct Digital Synthesis. Dostopno na:
http://en.wikipedia.org/wiki/Direct_digital_synthesis

[18] (2009) Metastability in electronics. Dostopno na:
http://en.wikipedia.org/wiki/Metastability_in_electronics