Triple Modular Redundancy -virheenkorjaustekniikan käyttö FPGA-piirien ajonaikaiseen suojaamiseen avaruuskäytössä

Mika-Petteri Kutila

TkK-tutkielma

Turun yliopisto
Informaatioteknologian laitos
1.3.2012

Linja: Tietotekniikan, elektroniikan ja tietoliikennetekniikan koulutusohjelma

Laajuus: 8 opintopistettä

Tarkastaja: Tero Säntti

Hyväksytty:

TURUN YLIOPISTO
Informaatiotekniikan laitos
Matemaattis-luonnontieteellinen tiedekunta

MIKA-PETTERI KUTILA: Triple Modular Redundancy -virheenkorjaustekniikan käyttö FPGA-piirien ajonaikaiseen suojaamiseen avaruuskäytössä

TkK-tutkielma, 27 sivua, 3 liitesivua
Tietotekniikka, elektroniikka ja tietoliikennetekniikka
Maaliskuu 2012

Avaruuteen tai muualle säteilyalttiiseen ympäristöön tarkoitetut elektroniset laitteet on suojattava säteilyltä. Nykyään säteilyalttiissa paikoissa käytetään paljon SRAM-tyyppisiä FPGA-piirejä. SRAM-solut ovat kuitenkin herkkiä vaihtamaan niihin tallennettuja arvoja säteilylle altistuessaan. Triple modular redundancy -tekniikka on yleinen tapa suojata FPGA-piirejä käytön aikana. Tässä tutkielmassa selvitetään, kuinka FPGA-piirejä suojataan säteilyltä TMR:n avulla.

TMR toimii siten, että yksittäinen piirin osa monistetaan kolmeksi rinnakkain toimivaksi osaksi. Nämä kolme lohkoa suorittavat aina keskenään samaa tehtävää. Yhden vikaantuessa kaksi kopiota antavat edelleen oikean ulostulon. Lopullinen ulostulo äänestetään kolmesta ulostuloista siten, että valitaan sellainen signaali, joita on kaksi tai enemmän.

TMR huono puoli on se, että se kolminkertaistaa piirin koon. TMR:stä onkin kehitetty erilaisia variaatioita, joilla pyritään minimoimaan piirin pinta-alaa ja tehonkulutusta, mutta kuitenkin säilyttämään TMR:n antama toimintavarmuus. Näissä tekniikoissa arvioidaan piirin eri osien vaikutuksia koko järjestelmän toimintaan, ja lisätään TMR-suojausta vain sellaisiin osiin, joiden vikaantumisesta on eniten haittaa.

TMR pystyy luotettavasti korjaamaan piirillä tapahtuvan yksittäisen vikaantumisen. Useampi samanaikainen vikaantuminen voi johtaa siihen, että yksittäiset piirin osat toimivat väärin ja näin mahdollisesti koko piiri lakkaa toimimasta. Piiri pitää pystyä korjaamaan jollain muulla menetelmällä ennen kuin vikoja kertyy liikaa. Viat korjaantuvat, kun koko piirin toiminta ohjelmoidaan uudelleen. Uudelleenohjelmointia käytetäänkin virheidenkorjaustekniikkana. Tällaista uudelleenohjelmointia kutsutaan scrubbingiksi, ja sen käyttö yhdessä TMR:n kanssa on yleisin tapa suojata SRAM-FPGA-piirejä.

ASIASANAT: FPGA, kenttäohjelmoitava logiikkapiiri, SRAM, staattinen käyttömuisti, SEU, single event upset, TMR, triple modular redundancy, säteily, avaruus, VHDL

SISÄLTÖ

1 JOHDANTO

Säteilyä on jonkun verran kaikkialla, ja kaikki elektroniikka altistuu sille. Säteily voi aiheuttaa elektronisten laitteen toimintaan hetkellisiä tai pysyviä häiriöitä. Jos halutaan laitteiden toimivan luotettavasti, pitää ne suojata säteilyltä. Erityisen tärkeää on suojata sellaisia laitteita, joita käytetään paikoissa, joissa on paljon säteilyä, jos säteilyaltistuminen on pitkäaikaista, tai jos laitteita ei päästä helposti korjaamaan. Erityisen paljon säteilyä on esimerkiksi ydinvoimaloissa ja avaruudessa, ja avaruudessa laitteet ovat alttiina säteilylle pitkiä aikoja ja laitteiden korjaaminen avaruudessa on erittäin vaikeaa. Nykyään tällaisissa paikoissa käytetään paljon SRAM-tyyppisiä FPGA-piirejä (field-programmable gate array). Tämän työn toisessa kappaleessa kerrotaan, miksi näin on, ja kuinka piirejä suojataan säteilyltä.

TMR (triple modular redundancy) lisää FPGA-piirin toimintavarmuutta säteily-ympäristössä. Se toimii siten, että samoja toimintoja suoritetaan rinnakkain useassa lohkossa. Näin yksittäisen lohkon vikaantuminen ei vaikuta piirin toimintaan. Kolmannessa kappaleessa tarkastellaan TMR:n toimintaa ja sen erilaisia toteutustapoja.

Neljännessä kappaleessa käydään läpi itsetekemäni TMR-toteutus. Siinä suunniteltiin VHDL-kielellä satunnaisgeneraattorin suojaaminen TMR:llä. Testijärjestelyssä satunnaisgeneraattoreihin syötettiin keinotekoisesti virheitä, jotka mallinsivat tietynlaisia säteilyn aiheuttamia virheitä. Tämä järjestely ohjelmoitiin FPGA-piirille ja ajettiin läpi. Ulostuloihin asti päässeet virheet laskettiin.

2 SÄTEILY-YMPÄRISTÖ

Avaruudessa oleva säteily aiheutuu auringosta ja muista tähdistä. Säteilyä on kahdenlaista. Ensinnäkin se voi koostua varatuista hiukkasista, joista yleisimpinä ovat protonit. Muita varattuja hiukkasia ovat elektronit, heliumytimet ja raskaat ionit. Toiseksi säteily voi olla elektromagneettista säteilyä, jolloin on kyse fotonien aiheuttamasta röntgen- ja gammasäteilystä. Kun yksittäinen säteilyhiukkanen iskeytyy elektroniikkapiirille, voi se aiheuttaa osumakohdassa hetkellisen virtapulssin. Sopivassa kohdassa piiriä tapahtuva virtapulssi saattaa vaihtaa muistisoluihin tallennettuja arvoja. Tällaisia muistiarvojen vaihtumisia kutsutaan SEU-ilmiöksi (single event upset). SEU-ilmiöitä aiheuttavat avaruudessa lähinnä protonit. (Xin 2010: 1.) Maanpinnalla yleisimpiä SEU-ilmiöiden aiheuttajia ovat neutronit (Normand 1996: 2746).

SRAM-tyyppisissä (static random-access memory) FPGA-piireissä toimintalogiikka on toteutettu LUT-soluilla (lookup table). LUT-solujen toimintaa ohjaavat SRAM-muistisoluihin tallennetut arvot. SRAM-solut ohjaavat lisäksi FPGA-piirien muita toimintoja, kuten sisäisten siirtolinjojen ohjausta sekä konfiguraatiomuistia. Muutokset niissä SRAM-soluissa, jotka osallistuvat piirin toimintaan, saavat piirin toimimaan väärin. SRAM-FPGA-piirit ovat siis monelta osin herkkiä SEU-virheille. (Samudrala, Ramos & Katkoori 2004: 2957.)

Vaikka SEU ei aiheuttaisikaan vikaantumista itse piirin toimintaan, voi se muuttaa piirin käsittelemää ja piirille tallennettua tietoa. Piiri toimii tällöin aivan oikein, mutta se on väärässä tilassa tai sen käsittelemä tieto on virheellistä. Tällaista kutsutaan soft error -tyyppiseksi vikaantumiseksi. (Almukhaizim & Makris 2008: 23.)

Moderni piiritekniikka on lisäksi mitoiltaan niin pientä, että yksittäinen säteilyhiukkasosuma voi aiheuttaa häiriöitä useammassa kohdassa piiriä. Näin ollen yksittäinen SEU voi muuttaa useampien piirillä lähekkäin sijaitsevien muistisolujen arvoja. (Ramaswamy, Rockett, Patel, Danziger, Manohar, Kelly, Holt, Ekanayake & Elftmann 2009: 5.) On myös mahdollista, että useampi säteilyhiukkanen osuu piirille samanaikaisesti, ja myös silloin useamman muistisolun arvot voivat muuttua samanaikaisesti.

2.1 FPGA-piirien käyttö

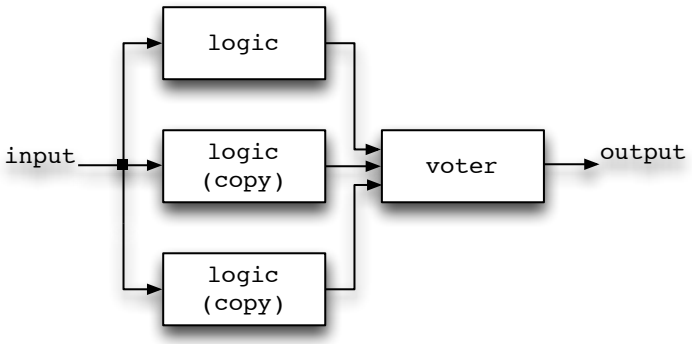
Erikseen tiettyyn käyttöön valmistettavien ASIC-piirien (application specific integrated circuit) hyvänä puolena on se, että ne ovat heti käynnistämisen jälkeen toiminta-kykyisiä. Niiden toimintalogiikka myös kestää SEU-ilmiöitä hyvin. ASIC-piirejä ei kuitenkaan voida ohjelmoida uudelleen, eikä niitä pystytä päivittämään eikä korjaamaan millään tavalla. (Carmichael, Fuller, Blain & Caffrey 1999: 1.)

FPGA-piirien suurin hyöty ASIC-piireihin verrattuna on niiden halpuus. ASIC-piirin tekeminen maksaa noin 250 tuhatta dollaria. SRAM-soluilla toteutettujen FPGA-piirien hyötynä on myös mahdollisuus ohjelmoida niitä uudelleen suunnittelun aikana muuttuvien tarpeiden mukaan. Piiriin voidaan tehdä helposti muutoksia missä tahansa suunnittelun vaiheessa, ja piiri voidaan ohjelmoida uudelleen myös käytön aikana. Useissa NASA:n (National Aeronautics and Space Administration) järjestelmissä on käytetty antifuse-tekniikalla toteutettuja FPGA-piirejä. Ne ovat kuitenkin vain kertaalleen ohjelmoitavia, eikä niille mahdu yhtä paljon logiikkakomponentteja kuin SRAM-tyyppisille FPGA-piireille. Sen vuoksi SRAM-FPGA-piirien käyttö avaruus-teknologiassa on yleistymässä. (Adell & Allen 2008: 1.) Myös tuotekehitykseen kuluva aika on lyhyt, kun piiriä ei tarvitse erikseen valmistaa tai valmistuttaa, vaan piirejä on valmiina myynnissä. SRAM-FPGA-piirit ovat kuitenkin hyvin herkkiä SEU-ilmiöille, joten ne pitää suojata hyvin säteilyltä (Samudrala et al. 2004: 2957).

Avaruuteen lähetettävät ASIC-tyyppiset elektroniikkapiirit suunnitellaan käyttämällä komponenttikirjastojen säteilynkestäviä komponentteja. Niissä kunkin loogisen portin toteutuksessa käytetään useita ylimääräisiä rinnakkain toimivia transistoreita. Tällainen säteilynsuojaustapa ei kuitenkaan onnistu FPGA-piirien kohdalla, koska niiden rakenne on valmiiksi tietynlainen, ja sen kanssa on tultava toimeen. Vastaavanlainen vika-sietoisuus pitää saada aikaiseksi muilla tavoilla, kuten monistamalla logiikkamoduuleja ja suorittamalla samoja operaatioita useassa lohkossa rinnakkain. (Samudrala et al. 2004: 2957.)

2.2 Säteilyltä suojaaminen TMR-tekniikalla

TMR-tekniikan esitti ensimmäisenä Von Neumann vuonna 1956. Siinä piirilohkon toimintalogiikka monistetaan kolmeksi rinnakkain toimivaksi kopioksi, ja näiden lohkojen ulostuloista äänestetään lopullinen ulostulo. Tämä toimintaperiaate on kuvattu kuvassa 1. Yhden logiikkalohkon vikaantuminen ei tässä vaikuta ulostuloon. Kuvasta 1 nähdään, että piirin pinta-ala kasvaa yli kolminkertaiseksi alkuperäiseen logiikkaan verrattuna. TMR:n huono puoli onkin sen aiheuttama piirin pinta-alan kasvu. (Samudrala et al. 2004: 2957.)



KUVA 1. TMR:n periaate

FPGA:n suojaamiseksi on esitetty muitakin tekniikoita. Niitä on valmistuksenaikaisia, suunnittelunaikaisia ja käytön aikana itsekorjaavia tekniikoita. FPGA-piirejä suojattaessa SEU-häiriöiltä suunnittelunaikainen TMR on nykyään yleisimmin käytetty tekniikka. Tämä johtuu TMR:n suoraviivaisesta toteutustavasta ja luotettavuudesta. TMR:n aiheuttama piirin pinta-alan merkittävä kasvu on kuitenkin saanut suunnittelijat kehittämään erilaisia tekniikoita lisäpinta-alan pienentämiseksi. (Xin 2010: 1.)

FPGA-piirit koostuvat monenlaisista piirinsisäisistä komponenteista, joista monet ovat herkkiä SEU-virheille. Esimerkiksi Xilinxin FPGA-piireillä SEU-herkkiä lohkoja ovat esimerkiksi erilliset Block SelectRAM -muistipiirit, kellopiirit ja sisään- ja ulostulopiirit. Myös näitä pitää suojata säteilyltä. Konfiguraatiobittejä on kuitenkin piirillä niin paljon, että niiden suojaaminen on ensisijaista avaruuteen lähetettävien piirien kohdalla. (Adell & Allen 2008: 7.) Esimerkiksi Xilinx Virtex XQVR1000 -piirillä SEU-herkkiä osia on piirin pinta-alaan suhteutettuna taulukon 1 mukaisesti (Caffrey, Graham, Johnson, Wirthlin, Rollins & Carmichael 2002: 1). Taulukon 1 mukaan pelkät konfigu-

raatio- ja LUT-bitit muodostavat 97,7 prosenttia koko piirin SEU-herkästä alueesta.

TAULUKKO 1. Virtex XQVR1000 -piirin osien osuudet koko piirin SEU-herkästä pinta-alasta (Caffrey et al., 2002, s. 1)

piirin osa	osuus piiristä	suhteellinen osuus piiristä
User flip-flops	26 112	0,4 %
LUT bits	393 216	6,4 %
Block SelectRAM	131 072	2,1 %
Configuration	5 603 456	91,0 %
Single event functional interrupts	?	< 0,0021 %
Transients	?	?
Half-latches	?	?

Jatkuva elektroniikkapiirien pieneneminen, matalampien käyttöjännitteiden käyttö ja signaalitaajuuksien kasvu aiheuttavat sen, että kehityksen edetessä kaikki laitteet alkavat olla herkempiä säteilylle. Koska kaikkialla säteilee jonkin verran, pitää tulevaisuudessa alkaa suojaamaan sellaistakin elektroniikkaa, jota käytetään maanpinnalla normaaliolosuhteissa. (Kastensmidt, Neuberger, Hentschke, Carro & Reis 2004: 552.)

TMR:n tärkein tehtävä on taata järjestelmän oikea toiminta yksittäisen SEU-virheen ilmaantuessa. TMR ei kuitenkaan voi taata oikeaa toimintaa, jos vikoja on enemmän kuin yksi. Piiriä pitää ohjelmoida uudelleen aika-ajoin vikojen kertymisen estämiseksi. Adellin ja Allenin (2008: 29) mukaan tehokkain suojaus saadaan aikaiseksi käyttämällä TMR:ää ja uudelleenohjelmointia yhdessä.

2.3 Scrubbing

Kun säteily aiheuttaa vikoja SRAM-FPGA-piirille, estää TMR niiden vaikutuksia piirin toimintaan. Näin piirin toiminta pysyy oikeanlaisena yksittäisistä vioista huolimatta. Altistumisajan jatkuessa piiri kuitenkin vikaantuu jatkuvasti lisää. Lopulta vikoja kertyy niin paljon, ettei TMR enää riitä korjaamaan kaikkea. Tästä syystä TMR ei yksinään riitä suojelemaan piiriä pitkäaikaisen säteilyn vaikutuksilta. Kertyneet viat voidaan kuitenkin korjata ohjelmoimalla koko piiri uudelleen. Piirin suojaamista aika-ajoin uudelleenohjelmoimalla kutsutaan nimellä scrubbing. Scrubbingia voidaan käyttää SRAM-FPGA-piireissä, koska ne ovat uudelleenohjelmoitavia. Scrubbing mahdollistaa

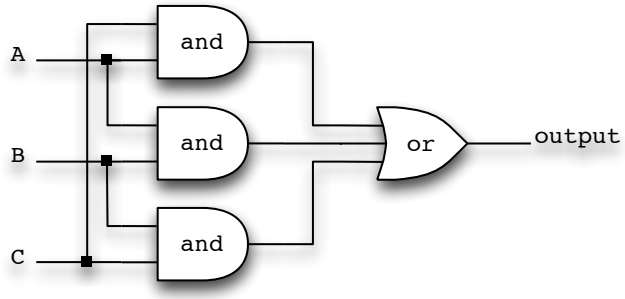
piirin pitkäaikaisen toiminnan säteilylle alttiissa ympäristössä. TMR:n pitäessä piirin toiminnan oikeellisena scrubbingin suorittamisten välillä piirin luotettavuus paranee. (Adell & Allen 2008: 20.)

Adell ja Allen (2008: 20) suosittelevat scrubbingia suoritettavaksi vähintään kymmenen kertaa useammin kuin pahin mahdollinen arvioitu SEU-virheiden esiintymistaajuus. Scrubbingin suoritustaajuuteen vaikuttaa säteilyhiukkasten virran suuruus ja piirin pinta-ala. Edmonds (2009) päättelee, että SEU-virheiden esiintymistaajuutta ei voi arvioida tarkasti yksinomaan laskennallisesti. Hän altistaakin piiriä säteilylle hiukkaskiihdyttimessä. Näin saadusta kokeellisesta testidatasta voidaan tehdä oikeellisempia arvioita SEU-taajuuksista.

TMR:n käyttäminen yhdessä scrubbingin kanssa on yleisin tapa suojata FPGA-piirejä korkean tason, eli esimerkiksi VHDL- tai Verilog-kielellä tehtävässä suunnittelussa (Adell & Allen 2008: 17).

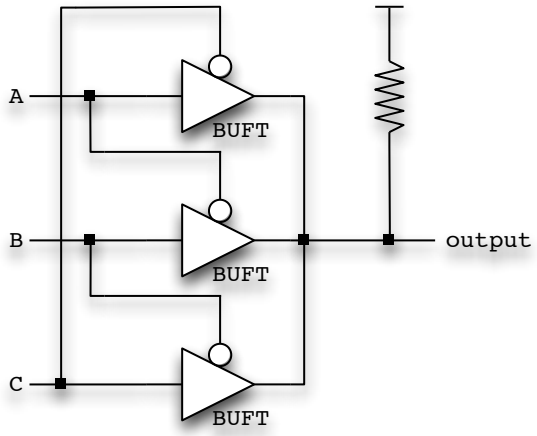
3 TRIPLE MODULE REDUNDANCY

TMR:n avulla suojattavan piirin yksittäinen osa kopioidaan kolmeksi samanlaiseksi lohkoksi, jotka suorittavat samaa toimintaa rinnakkain (kuva 1, sivu 4). Näiden kolmen lohkon ulostulot liitetään äänestyslogiikkaan, jonka toiminta on kuvan 2 mukainen. Äänestyslogiikka valitsee kolmesta sisääntulosignaalista (kuvassa 2 A, B ja C) sen arvon, joita on enemmistö, eli kaksi tai enemmän. Jos siis yksi kolmesta piiristä hajoaa ja antaa väärän tuloksen, on ulostulo kuitenkin oikea. Kuvan 2 signaalit ovat yksibittisiä. (Hentschke, Marques, Lima, Carro, Susin & Reis 2002: 3.) TMR:ää voidaan soveltaa joko rekisteripiireihin tai logiikkapiireihin tai molempiin.



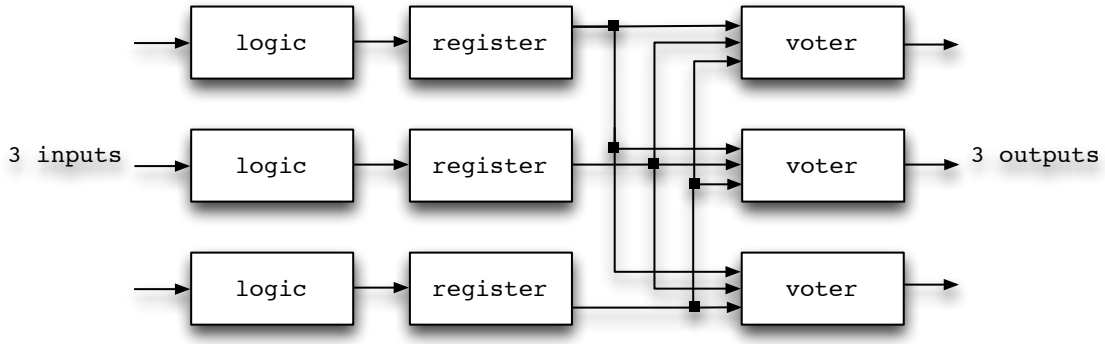
KUVA 2. Äänestyslogiikka

Äänestyslogiikka toteutetaan yleensä FPGA-piirille tulkitsemalla äänestyslogiikka LUT-lohkon toiminnaksi. LUT-lohkon konfiguraatiobitit vastaavat äänestyslogiikan totuustaulua. Carmichael (2001: 3–6) esittelee toisenlaisen tavan toteuttaa äänestyslogiikka kokoamalla se LUT-lohkojen sijasta Xilinxin Virtex-piirillä olevista tri-state buffereista (komponentti BUFT). Hänen mukaansa tällaisella ratkaisulla saadaan säästettyä FPGA-piirin logiikkalohkoja muuhun käyttöön, ja näin piirille saadaan suunniteltua enemmän toimintoja. Kuvassa 3 on kuvattu tällainen toteutus Xilinxin Virtex-piirin BUFT-buffereilla. Samudralan työryhmän (2004: 2958–2959) mukaan tällainen kestää SEU-ilmiöitä paremmin kuin SRAM-toteutus, koska yksittäinen häiriö aiheuttaa vain siirtolinjan kytkeytymisen irti väliaikaisesti, eikä äänestyslogiikasta ulostuleva arvo muutu.



KUVA 3. Tri-state buffereista koottu äänestyslogiikka

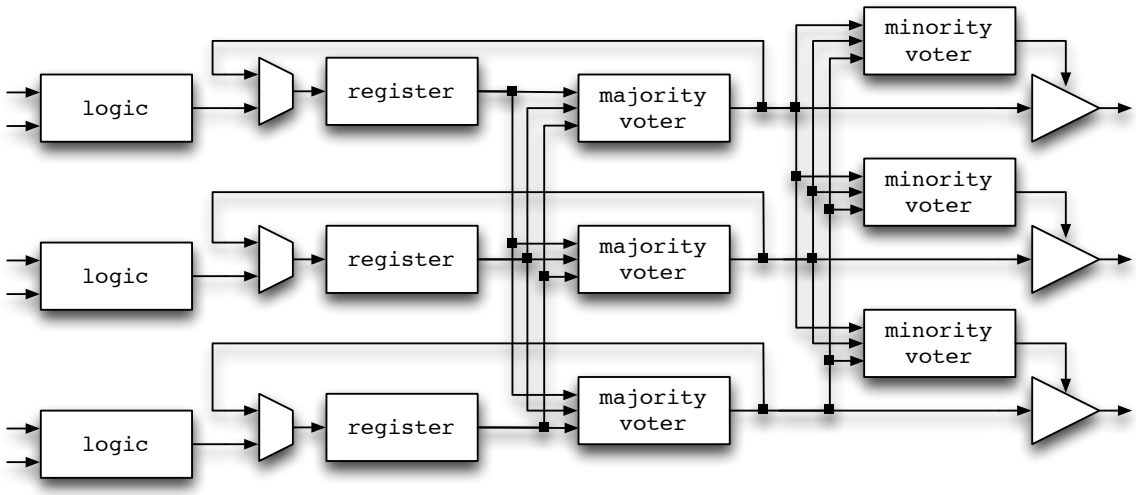
Koska SEU:t voivat osua piirillä mihin kohtaan vain, parempi suojaus saavutetaan suorittamalla kaikkia toimintoja, mukaan lukien äänestystä, kolmessa rinnakkaisessa moduulissa. Tällaista toteutusta havainnollistaa kuva 4. Siinä minkään yksittäisen lohkon vikaantuminen ei vaikuta ulostulojen oikeellisuuteen. Myös sisääntulo- ja ulostulosignaalit ovat kolmena kappaleena, joten yksittäiset viat siirtolinjoissakin korjaantuvat. Kuvassa 4 näkyvään piiriin piiriin voidaan lisätä vielä äänestyslogiikka logiikkakomponenttien ja rekistereiden väliin, jolloin saadaan piiriin vielä lisää toimintavarmuutta. Kaavio tämäntyyppisestä toteutuksesta on nähtävissä kuvassa 9 sivulla 18.



KUVA 4. TMR, jossa logiikka, rekisterit ja äänestyslogiikka on monistettu

Kuvan 4 kaltaista toteutusta TMR:stä käyttää ainakin Xilinxin kehittämä ohjelma TMRTool. Se automatisoi TMR:n lisäämisen piirille. Perinteiseen TMR-tekniikkaan verrattuna TMRTool kolmentaa kaiken piirillä olevan. Se monistaa logiikkalohkojen, rekistereiden ja äänestyslogiikkojen lisäksi kaikki sisääntulot, kellot ja ohjauslogiikan

(*throughput logic*). Piirin vikaannuttua scrubbingilla saadaan korjattua piirin toiminta, mutta piirin käsittelemä tietoon se ei vaikuta. Rekistereihin tallennetun tiedon vikaantuminen saadaan korjattua TMRToolin tekemän takaisinkytkennän avulla. Siinä äänestyslogiikan tulos on kytketty takaisin rekisterin sisääntuloon (kuva 5). TMRTool-ohjelman tekemä TMR-järjestelmä korjaa rekisterien arvot virkistämällä ne seuraavalla kellonlyömällä oikeiksi. Äänestyslogiikan vikaantuminen havaitaan toisella äänestyslogiikalla, joka havaitsee vähemmistön (minority voter) ja kyseinen ulostulo poistetaan käytöstä tristate bufferilla (kuva 5). Näin ulostulosignaaleihin ei syötetä samanaikaisesti loogista ykköstä ja loogista nollaa. (Xilinx 2009.)



KUVA 5. TMRtool -ohjelman tuottaman TMR:n toimintaperiaate

Myös FPGA-piirin sisään- ja ulostulot on hyvä suojata. Esimerkiksi kuvan 5 tyyppisellä tekniikalla voidaan suojata ulostuloja. Tällöin kuvan 5 kolme ulostuloa liitetään FPGA-piirin kolmeen ulostulopinniin, ja nämä kolme pinniä liitetään yhteen vasta piirin ulkopuolella. Vähemmistöäänestyslogiikka (minority voter) estää konfliktit piirin ulostuloissa. Tällainen tekniikka kuitenkin käyttää piiriltä kolme ulostulopinniä yhtä ulostuloa varten. (Adell & Allen 2008: 19.)

3.1 TMR:n aiheuttaman lisäpinta-alan hillitseminen

Jotta FPGA-piirien resurssit saataisiin hyödynnettyä paremmin, TMR:stä on pyritty kehittämään tehokkaampia muunnelmia. Niiden tarkoituksena on pienentää käytettyä

pinta-alaa, ja samalla saada piireille mahtumaan enemmän logiikkaa. Näissä TMR:n muunnelmissa ei monisteta sokeasti kaikkea mitä piirillä on. Eri tekniikoilla arvioidaan, mitkä osat piirillä ovat herkimpiä virheille, ja missä tapahtuvat virheet johtavat piirin väärään käyttäytymisen. Arvioinnin perusteella lisätään rinnakkaisuutta vain piirin herkimpiin osiin, ja jätetään vähemmän herkät osat toimimaan alkuperäisen suunnittelun mukaisesti ilman suojausta. Yksi esimerkki tällaisesta tekniikasta on Samudralan työryhmän (2004) esittelemä Selective TMR (STMR). Siinä lasketaan suunnitteluvaiheen jälkeen piiristä signaalien muuttumistodennäköisyyksiä, ja lisätään TMR-tekniikkaa vain niihin piirin osiin, joissa vikaantuminen aiheuttaa haittaa suurimmalla todennäköisyydellä. Reduced TMR (RTMR), jonka esittelee Kamakotin työryhmä (Kamakoti, Chandrasekhar, Mahammad, Vijaykrishnan & Kamakoti 2005), arvioi vasta technology mapping -vaiheen jälkeistä LUT-solujen verkkoa ja lisää TMR-tekniikkaa herkimmille LUT-soluille. Almukhaizim ja Makris (2008) esittelevät tekniikan, jossa monistetaan siirtolinjoja (*wires*), ja näin pyritään estämään SEU-ilmiöiden aiheuttamien virheiden pääsemistä muistisoluihin tai ulostuloihin. (Xin 2010: 1.)

3.1.1 Selective TMR

STMR-tekniikassa lähtökohtana ovat piiriin sisääntulevien signaalien arvojen todennäköisyydet. Arvot riippuvat piirin käyttötarkoituksesta ja käyttöympäristöstä. Niiden avulla saadaan määriteltyä myös piirin sisäisten signaalien todennäköisyydet. Piirin kunkin logiikkaportin sisääntulosignaalit määritellään herkiksi ja ei-herkiksi sen mukaan, vaikuttaako sisääntulosignaalin arvon vaihtuminen portin ulostuloon. Sisääntulosignaalin määrittely herkäksi johtaa myös kyseisen portin määrittelyyn herkäksi. STMR:ssä käytetään lisäksi threshold-arvoa, jota säätämällä tulkintaa herkistä ja ei-herkistä signaaleista muuttuu. (Samudrala et al. 2004: 2961–2962.)

Piiristä monistetaan TMR-tekniikalla vain herkiksi määritetyt osat - ei siis kaikkea kuten normaalisti TMR-tekniikkaa käytettäessä. Threshold-arvoa säätämällä TMR-suojauksen määrä piirillä kasvaa tai vähenee. Samudralan työryhmä (2004: 2957) tuo ilmi, että vaikka TMR:ään verrattuna STMR vähentää hieman koko piirin SEU-sietoisuutta, sillä saadaan pienennettyä suojatun piirin pinta-alaa merkittävästi. He käyttävät esimerkkinä Xilinxin Virtex-piiriä, ja päättelevät, että kun siihen käytetään STMR:n

lisäksi muita virheenkorjausmenetelmiä, kuten Virtexin readback and reconfiguration -ominaisuutta, saadaan piiristä hyvin SEU:ita kestävä.

Perinteiseen TMR:ään verrattava säästö määräytyy piiriin tulevien sisääntulojen ominaisuuksista ja muutenkin piirin ominaisuuksista. Pahimmassa tapauksessa STMR tuottaa kuitenkin aivan samanlaisen piirin kuin perinteinen TMR:kin, jolloin etua ei synny. (Samudrala et al. 2004: 2969.)

3.1.2 Reduced TMR

RTMR (Reduced TMR, Selective TMR which operates on LUTs) muokkaa LUT-verkkoa, joka on saatu aikaiseksi FPGA-piirin technology mapping -suoritusvaiheen seurauksena. Piiriltä arvioidaan signaalien muuttumistodennäköisyyksiä (signal probability propagation), ja niiden perusteella jaetaan LUT-solut kolmeen luokkaan: *sensitive*, *internally insensitive* ja *last-level insensitive*. Vain *sensitive*- ja *last-level sensitive* -luokituksen saaneisiin LUT-soluihin sovelletaan TMR-tyyppistä kolmentamista, ja näin saavutetaan säästöä perinteiseen TMR:ään verrattuna. Lisäksi tietyissä tilanteissa *last-level insensitive* -lohkoille riittää kahdentaminen, jolloin säästyy vielä lisää LUT-lohkoja. Äänestyslogiikka suoritetaan sivulla 8 olevan kuvan 3 tyyppisillä tri-state buffereilla. (Kamakoti et al. 2005: 1.)

Kamakotin työryhmän testien perusteella RTMR tuottaa 99,61 prosentin lisäyksen alkuperäiseen LUT-solujen lukumäärään, kun perinteinen TMR tuottaa 200 prosentin lisäyksen. Kamakotin työryhmä toteaa testien perusteella, että vaikka pinta-alaa säästyy paljon, kestää RTMR-piiri silti hyvin SEU-ilmiöitä. (Kamakoti et al. 2005: 1.)

RTMR:n ero STMR:ään verrattuna on se, että STMR muokkaa piiriä porttitasolla, kun taas RTMR LUT-tasolla. Useimmat FPGA-CAD-työkalujen optimointitekniikat muokkaavat piirejä ennen LUT-verkon luomista STMR:n tapaan. (Kamakoti et al. 2005: 1–2.)

3.1.3 Selective addition of redundant wires

Almukhaizim ja Makris (2008) esittelevät tekniikan, joka perustuu ylimääräisiin siirtolinjoihin. Periaatteena on lisätä piirillä oleville siirtolinjoille rinnakkaisia linjoja, joissa kaikissa kulkee sama signaali. Jos yhden siirtolinjan signaali muuttuu virheelliseksi, pystyvät rinnakkaiset siirtolinjat estämään väärän arvon pääsemisen seuraavaan ulostuloon tai rekisteriin asti. Näin saadaan estettyä soft error -tyyppisten virheiden syntymistä. Lisäsiirtolinjat kuitenkin kasvattavat piirin pinta-alaa, tehonkulutusta ja viipeitä, ja lisäksi ne itsessään tuovat piirille lisää SEU-herkkiä elementtejä. Tätä tekniikkaa sovelletaan jo logiikkasuunnitteluvaiheessa.

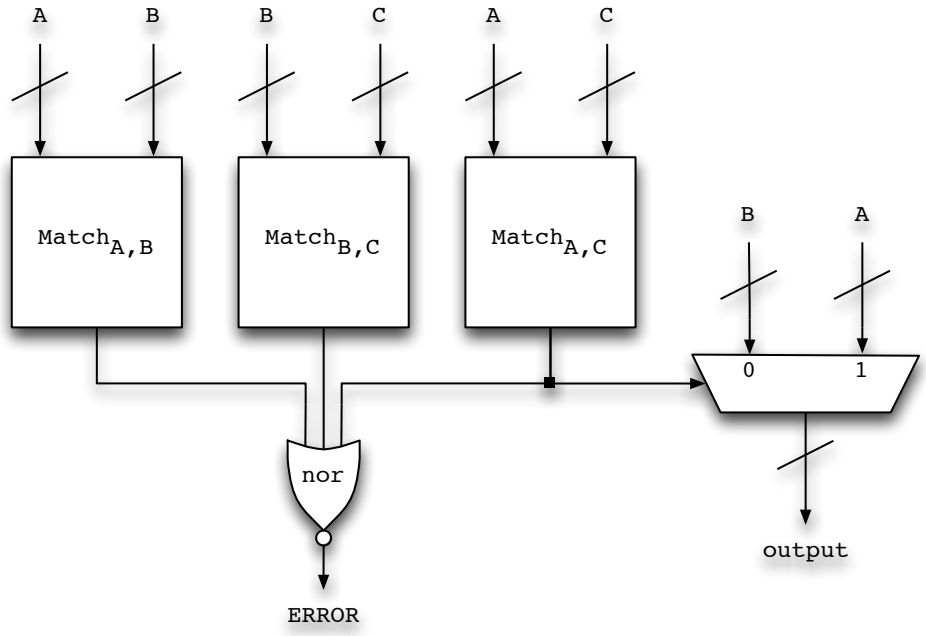
3.2 Useampibittinen äänestyslogiikka (Word-Voter)

Useamman bitin pituinen signaali voidaan suojata samalla tavalla kuin useampi yksibittinen signaali, eli bitti bitiltä. Tällainen suojaus rakennetaan käyttämällä sivulla 7 olevan kuvan 2 kaltaista yksibittistä äänestyslogiikkaa sanan kullekin bitille. On kehitetty myös kerralla koko sanan tarkistavia TMR-tekniikoita. Mitran ja McCluskeyn (2000) esittelemä Word-Voter-tekniikka on eräs tällainen. Heidän mukaansa Word-Voter on käytännöllinen erityisesti, kun tapahtuu useampia samanaikaisia virheitä. Bitti bitiltä -tyyppinen tarkastaminen on kuitenkin yleisemmin käytössä (Mitra & McCluskey 2000: 2).

Kokonaista sanaa tarkistettaessa eteen voi tulla sellainen tilanne, että kaikki kolme sisään tulevaa sanaa ovat erilaisia. Tällaista voi tapahtua, jos kahdessa sanassa tapahtuu yhtä aikaa vikaantuminen eri kohdissa sanaa. Word-Voter havaitsee tällaisen virhetilanteen, ja ilmaisee ylimääräisellä ERROR-ulostulobitillä, että enemmistöä ei pystytty määrittämään, ja että signaalissa on virhe. (Mitra & McCluskey 2000: 466–468.)

Word-Voter-tekniikan toimintaperiaate on tunnistaa kolmesta sisään tulevasta sanasta pareittain, ovatko mitkään kaksi niistä samoja. Tämä suoritetaan kolmella vertailijalla (kuvassa 6 Match). Jos kaikki kolme sanaa ovat erilaiset, ei oikeaa ulostuloa pystytä äänestämään, ja tilanne tulkitaan virhetilanteeksi. Virheen ilmaisemiseksi ERROR-bitti muutetaan ykköseksi. Tällöin muilla ulostuloilla ei ole väliä, kun oikeasta ulostulosta ei

ole varmuutta. Kun ei olla virhetilassa, ainakin kaksi sisääntuloa ovat samoja. Silloin Word-Voter toimii samoin kokonaisilla sanoilla, kuin perinteinen TMR:n äänestys yksittäisillä biteillä. Esimerkiksi kuvassa 6 sisääntulosignaalit ovat sanat A, B ja C. Jos sanat A ja C ovat samat, päästetään ulos sana A. Jos sanat A ja C eivät ole samat, päästetään ulostuloon sana B. (Mitra & McCluskey 2000: 467.)



KUVA 6. Word-Voter, sisääntuloina sanat A, B ja C, ulostuloina ERROR-bitti ja output-sana

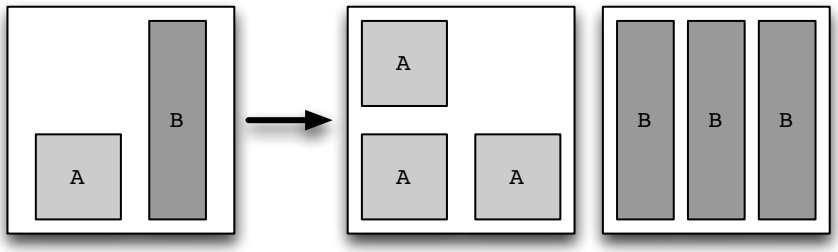
Word-Voter-tekniikassa hyvänä puolena on se, että järjestelmän vikaantuessa siitä saadaan ilmoitus. Näin voidaan suorittaa järjestelmän korjaus, esimerkiksi scrubbing, juuri oikealla hetkellä. Word-Voter-tekniikalla ensimmäinenkään virheellinen arvo ei ehdi päästä ulos. Perinteinen bitti bitiltä tehty TMR ei ilmaise mitenkään, onko ulostulossa virhettä, ja korjaamisen ajoitus pitää määrittää jollain muulla tavalla.

Jos vika on sopivasti kahdessa sanassa eri kohdissa sanaa, bitti bitiltä -tyyppinen tarkistus olisi voinut korjata sen, vaikka Word-Voter antaisikin virheilmoituksen. Tällaisessa tilanteessa Word-Voter antaa virheilmoituksen liian aikaisin, kun korjaustoimet eivät ole vielä aivan välttämättömiä. Word-Voter kuitenkin havaitsee sellaiset virheet, jotka aiheuttavat useita samanaikaisia muutoksia eri kohtiin eri sanoja, ja jonkalaisia

bitti bitiltä -tyyppinen tarkistus olisi päästänyt virheellisenä ulostuloon asti.

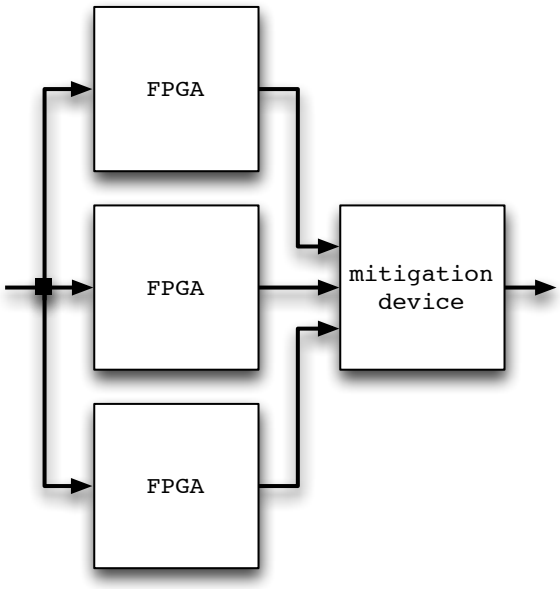
3.3 Toiminnan jakaminen useammalle FPGA-piirille

Jos käytetään perinteistä TMR:ää, toimintalogiikka joudutaan kolmentamaan. Jos alkuperäinen logiikka vie yli kolmanneksen piirin resursseista, ei kolmennettu systeemi mahdu yhdelle piirille. Tällöin voidaan toiminta jakaa useampaan moduuliin, ja jakaa moduulit useammalle piirille. Tällainen toteutus on esitetty kuvassa 7. Siinä moduuli A on kolmennettu yhdelle piirille ja moduuli B toiselle. Logiikan ulostulot ja moduulien väliset signaalit pitää yhdistää oikein, ja se voi olla hankalampaa kuin yhden piirin tapauksessa. Järjestely on hyvä suunnitella siten, ettei aikakriittisiä signaaleja kulje fyysisten piirien välillä, koska silloin koko systeemin toimintakyky huononee. (Carmichael et al. 1999: 8–9.)



KUVA 7. Moduulit A ja B kolmennettuna ja jaettuna kahdelle erilliselle piirille

Triple device redundancy on suojaustapa, jossa kuvan 8 tavalla koko logiikka kolmennetaan kolmelle erilliselle FPGA-piirille. Lisäksi ulostulojen yhdistämiseen tarvitaan erillinen piiri tai useita piirejä, jos ulostuloja on paljon. Erillinen piiri voi olla TMR:llä suojattu FPGA-piiri, tai säteilynkestävä ASIC-piiri. Triple device redundancy kestää hyvin yksittäisen tai useamman samanaikaisen SEU:n tai minkä tahansa yksittäisessä piirissä tapahtuvan toiminnallisen häiriön tai vaikka koko piirin hajoamisen. Tällainen toteutus on kuitenkin kallis, eikä kuitenkaan anna merkittävästi parempaa suojausta kuin muut suojaustavat. (Carmichael et al. 1999: 10.)



KUVA 8. Triple device redundancy

Eräs vaihtoehto on käyttää erillistä prosessoria hallinnoimaan monistettuja erillisiä piirejä. Tällöin yhden piirin vikaannuttua sitä voidaan alkaa korjaamaan, ja samalla toimintaa voidaan jatkaa seuraavalla ehjällä piirillä. Näin toimintaan ei tule keskeytyksiä, eivätkä vikaantuneen piirin korjaustoimet aiheuta viivytyksiä. (Carmichael et al. 1999: 10.)

3.4 Virheellistä toimintaa aiheuttava vikaantuminen

TMR:ssä kolmennetaan yksittäinen piirin lohko, jotta vikaantumisen tapahtuessa olisi vielä kaksi oikein toimivaa kopiota samasta lohkosta. TMR korjaa siis ulostulon oikeaksi, jos kolmesta samanlaisesta lohkosta yksi antaa virheellisen ulostulon. Jos kahdessa lohkossa tapahtuu samanaikainen vikaantuminen siten, että signaaleissa on virheet sanan eri kohdissa, korjaa bitti bitiltä toteutettu TMR silloinkin ulostulon oikeaksi. Tällainen tilanne on nähtävissä taulukossa 2, kun korjattavista signaaleista B ja C sisältävät kumpikin virheen, mutta eri kohdissa sanaa. Kuitenkin, jos tapahtuu vikaantumista siten, että kahden sanan samassa kohdassa olevat bitit ovat virheellisiä, pääsee ulostuloksi virheellinen sana sekä perinteisestä TMR:stä sekä Word-Voterista. Taulukosta 2 nähdään myös Word-Voter-äänestyksen ERROR-bitillä ilmaisema virhetilanne, jolloin voidaan aloittaa korjaustoimet ennen kuin ulostuloon mahdollisesti

vaikuttavia virhe ilmaantuu.

TAULUKKO 2. Kaksi samanaikaista virhettä kaksibittisessä signaalissa, virheetön ulostulo "00"

		TMR:ään sisääntulevien signaalien tapauksia		
		Ei virheitä	Virhe B:n ja C:n eri kohdissa	Virhe B:n ja C:n samassa kohdassa
Sisääntulot	A	00	00	00
	B	00	0**1**	**1**0
	C	00	**1**0	**1**0
Ulostulot	Bitti bitiltä TMR-korjattu ulostulo	00	00	**1**0 (virhe ulostulossa)
	Word-Voter-korjattu ulostulo	00	ERROR	**1**0 (virhe ulostulossa)

Yksittäisen SEU-ilmiön aiheuttaman vikaantumisen vaikutusta ulostuloihin voidaan pienentää suunnittelemalla piiri siten, ettei saman lohkon rinnakkaisia kopioita sijoiteta piirillä fyysisesti vierekkäin. Näin yksittäisen SEU:n aiheuttama vikaantuminen ei todennäköisesti tapahdu ainakaan samanaikaisesti saman lohkon kopioissa samassa kohdassa.

Huomioitavaa on, että vikaantumista tapahtuu tasaisesti joka kohtaan FPGA-piiriä. Osa vikaantumisista tapahtuu piirillä sellaisissa kohdissa, joissa ei ole mitään toimintaa ohjelmoituna. Osa ilmaantuu piirin sellaisessa osassa, joka on toiminnassa mukana, mutta juuri sen hetkisiä arvoja ei tarvita. Tällaiset virheet eivät vaikuta piirin toimintaan.

Jos vikaantuminen tapahtuu vain piirissä kulkevan signaalin arvoille, jatkaa piiri seuraavalla kellojaksolla toimintaansa oikein. Jos taas vikaantuminen tapahtuu konfiguraatiobiteissä, piirin toiminta jatkuu viallisena lopullisesti, tai kunnes suoritetaan korjaustoimenpiteenä esimerkiksi scrubbing seuraavan kerran.

Virheen ilmaantuminen piirille voidaan pyrkiä havaitsemaan, jotta voitaisiin ajoittaa piirin uudelleenohjelmointi juuri sille hetkelle, kun ensimmäinen virhe ei ole vielä aiheuttanut virheellisiä ulostuloja. Virheiden havainnointilogiikka vie kuitenkin omalta osaltaan vielä lisää piirin resursseja. Jos vikaantumista ei kuitenkaan havainnoida mitenkään, pitää scrubbing tai muut korjaustoimet ajastaa sokeasti todennäköisten vikaantumisaikojen perusteella. Tällöin käy niin, että piiri ohjelmoidaan uudelleen

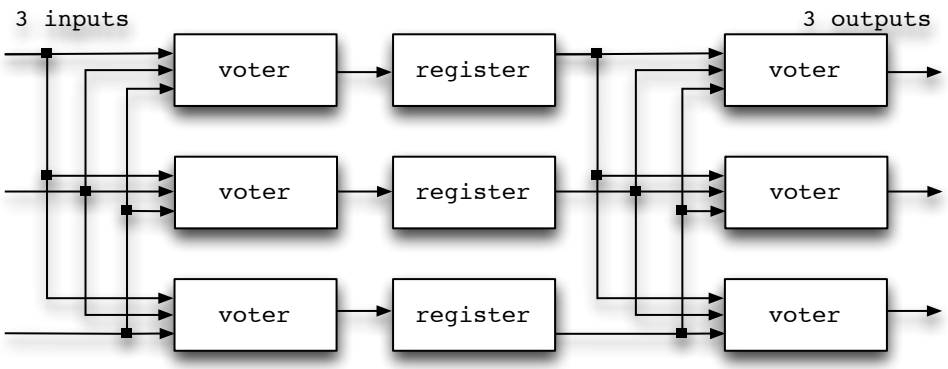
usein, vaikkei se olisi vikaantunutkaan, ja näin menetetään piirin hyödyllistä normaalia toiminta-aikaa.

4 TMR-TOTEUTUS VHDL-KIELELLÄ (TAPAUSTUTKIMUS)

Suunnittelin järjestelyn, jossa suojataan piiri TMR:llä. Näin saadaan tuntumaa siitä, mitä TMR:n toteuttaminen vaatii ja miten se toimii käytännössä. Suojattavana toimintalogiikkana on pseudosatunnaislogiikka, joka monistetaan kolmeksi ja suojataan erillisellä geneeriseksi suunnitellulla TMR:moduulilla. Erillisen ja monen pituisia signaaleja käsittelevän TMR-moduulin perusajatuksena on sen mahdollinen uudelleenkäyttö. Satunnaislogiikat on suunniteltu siten, että niiden ulostuloihin voidaan syöttää keinotekoisia virheitä. Virheet kuvaavat satunnaislogiikoissa tapahtuvia SEU-virheitä. TMR-suojauksen ulostuloa verrataan virheettömän satunnaislogiikan ulostuloon. Eroavaisuudet näissä kuvaavat ulostuloon asti päässeitä virheitä.

4.1 TMR-moduuli

Kolme identtistä satunnaisgeneraattoria suojataan liittämällä ne TMR-moduulin (kuva 9) kolmeen sisääntuloon. Moduuli sisältää rekisterin, jota ennen ja jälkeen on äänestyslogiikka. Äänestyslogiikat ja rekisterit on kolmennettu, joten moduuli toimii oikein yksittäisen äänestyslogiikkaosan tai rekisterin ollessa hajonneena. Moduuli kestää useammankin samanaikaisen vikaantumisen, jos ne tapahtuvat sopivasti, Esimerkiksi jos yksi sisääntuloista on vikaantunut, ja lisäksi TMR-moduulin (kuva 9) osista jonkin vaakatason osista kaikki kolme vikaantuvat, pysyy ulostulo oikeana. Ulostulo tulkitaan oikeaksi, jos kolmesta ulostulosta on enemmistö oikein.

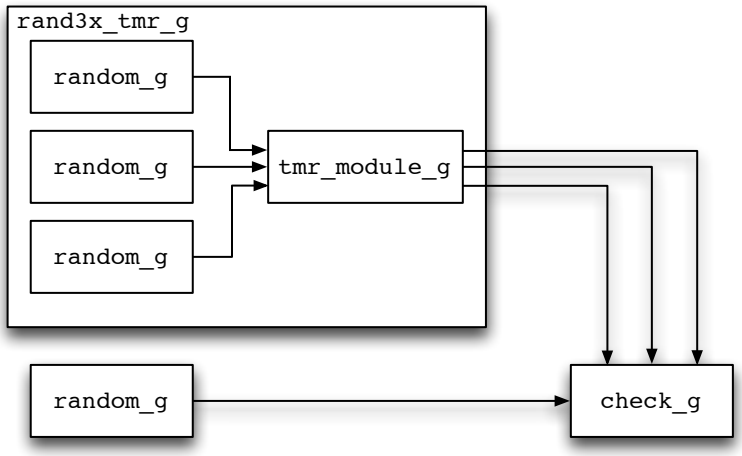


KUVA 9. TMR-moduuli (tmr_module_g)

Pelkkä kolmennettu äänestyslogiikka on suunniteltu erillisenä moduulina (liite 1). Ensimmäinen äänestyslogiikka korjaa moduuliin saamien kolmen sisääntulon virheitä, ja toinen äänestyslogiikka korjaa rekistereissä tapahtuvia virheitä. Kun tämä kaikki on yhdessä moduulissa (kuva 9 ja liite 2), on TMR helpompi lisätä halutun logiikkapiirin perään. Signaalit ovat vektoreita, joiden pituus määritellään generics-muuttujalla, joten moduulia voidaan käyttää eri pituisten signaalien kanssa. Yli yksibittisten signaalien kanssa TMR-suojaus suoritetaan samalla tavalla kullekin bitille, eli TMR-moduuli sisältää kuvan 9 piirin kutakin bittiä kohden.

4.2 Testausjärjestely

TMR:n toiminnan testausjärjestely koostuu neljästä samanlaisesta satunnaisluku-generaattorista, joista kolmen ulostulot kulkevat TMR:n kautta ja neljännen ulostulo toimii vertailusyötteenä. TMR-moduuliin syötettyihin signaaleihin generoidaan virheitä, ja TMR:n ulostuloja verrataan vertailusyötteeseen. Sekä generoitujen virheiden että ulostuloihin asti päässeiden virheiden määrät lasketaan check_g-moduulissa. kuva 10



KUVA 10. TMR:n testausjärjestely. Virheitä generoidaan kolmen ylimmän random_g-moduulin ulostuloihin.

Virheitä generoidaan kahteen alimpaan bittiin aina yhtäaikaisesti siten, että ne asetetaan ykkösiksi. Tällaista virhettä ei kuitenkaan voida havaita, jos alimpien bittien pitäisikin olla ykkösiä. Havaitseminen on mahdollista vain 75 prosentissa tapauksista. Tämä siksi, että kaksi alinta bittiä ykkösinä (1, 1) tuottavat virheellisen ulostulon vain silloin, kun alimpien bittien pitää olla jotain muuta, eli (0, 0), (0, 1) tai (1, 0). Tämä kahden bitin

virhe generoidaan samanaikaisesti kahteen signaaliin, joten virheen pitäisi näkyä TMR-suojauksesta huolimatta ulostulossa.

4.3 Tulokset

Virheitä generoitiin 1023 kertaa, ja ulostuloista oli virheellisiä 787. Virheen todennäköisyys oli tässä testissä 0,769. Eroavaisuus 0,75:stä selittyy sillä, että satunnaisgeneraattorit eivät käy systemaattisesti läpi kaikkia mahdollisia signaalien arvoja, ja otosmäärä on liian pieni.

4.4 Pohdintaa

Vaikka virheiden generointi vain 1023 kertaa ei anna tarkkaa tulosta, on tulos 0,769 kuitenkin lähellä odotettua tulosta 0,75. Satunnaislukugeneraattorin toteuttaminen enemmän satunnaisena ja niiden ajaminen pidemmän ajan tuottaisi tilastollisesti paremman tuloksen. Kuitenkin vaikuttaa siltä, että kaikki tässä kokeilussa generoidut virheet pääsivät läpi suojauksesta huolimatta.

Ajoin testijärjestelyn läpi myös siten, että virheitä generoitiin vain yhteen sisääntulosignaaliin. Silloin TMR-suojatussa ulostulossa ei ollut yhtään virhettä. Tämä vastaa TMR:n oikeaa toimintaa, eli yksittäisen virheen tapauksessa TMR pystyy korjaamaan ulostulon oikeaksi.

Vaikka TMR:n lisääminen piirille on suoraviivaista, tekee se piiristä suuremman ja monimutkaisemman. Tässä suojattava piiri oli varsin yksinkertainen, ja suojausta käytettiin vain yhden kerran. Jos TMR:llä suojataan monimutkaisempaa piiriä, hankaloituu piirin hallinnointi. Automaattinen TMR-työkalu helpottaa työtä varsinkin, jos halutaan toimintalohkojen lisäksi suojata muita FPGA-piirin osia, kuten kellosignaaleja tai sisään- ja ulostuloja.

5 YHTEENVETO

TMR-tekniikalla suojataan elektronisia piirejä säteilyn aiheuttamilta vikaantumisilta. SRAM-muistisolut ovat erityisesti herkkiä vaihtamaan niihin tallennettujen bittien arvoja säteilyn vaikutuksesta. SRAM-tyyppisten FPGA-piirien toiminta perustuu SRAM-soluihin tallennettuun tietoon, joten säteilyalttiissa paikoissa käytettävät FPGA-piirit pitää suojata säteilyn haitoilta. Yleisin tapa suojata niitä on käyttää TMR-tekniikkaa ja scrubbing-uudelleenohjelmointia yhdessä.

TMR-tekniikan ansiosta piiri toimii oikein, vaikka jokin sen yksittäinen osa vikaantuu. Scrubbing taas korjaa vikaantumiset ja palauttaa piirin alkuperäiseen tilaansa. Scrubbing vie aikaa, ja sen ajan piiri on poissa käytöstä. Siksi scrubbingia ei voida suorittaa koko ajan, vaan se suoritetaan yleensä tietyin väliajoin. TMR:n ansiosta piirin toimii scrubbingien välillä oikein.

TMR-tekniikassa lisätään yksittäisen piirin osalle kaksi kopiota toimimaan alkuperäisen osan rinnalla samanaikaisesti. Näiden kolmen osan ulostuloista äänestetään lopulliseksi ulostuloksi se, joita on kaksi tai enemmän. Jos siis yksi ulostuloista eroaa muista, eli yksi kolmesta osasta on vikaantunut, on lopullinen ulostulo oikea.

TMR-tekniikkaa voidaan soveltaa toimintalogiikkaan monella eri tavalla. Voidaan monistaa vain käytetyt rekisterit ja äänestää ulostulo niistä. Voidaan monistaa lisäksi toimintalogiikka, jolloin saadaan lisää toimintavarmuutta. Koska mikä tahansa osa piirillä voi vikaantua, myös äänestyslogiikka voidaan monistaa, jolloin saadaan vielä lisää toimintavarmuutta. Äänestyslogiikka voidaan toteuttaa usealla eri tavalla. Se voidaan rakentaa FPGA-piirin normaaleista logiikkalohkoista tai esimerkiksi tri-state buffereista, ja se voidaan sijoittaa järjestelmään vain rekistereiden perään, tai muuallekin logiikan sekaan. TMR-tekniikkaa voidaan toteuttaa myös useampaa fyysistä piiriä käyttäen.

TMR:ää on käytetty jo pitkään. Perinteinen TMR aiheuttaa piirin pinta-alan kasvun kolminkertaiseksi, ja sen takia on kehitetty muunnelmia TMR:stä, joiden tarkoituksena on lähinnä pienentää suojatun piirin pinta-alaa. Esimerkkejä tällaisista tekniikoista ovat

Selective TMR ja Reduced TMR, joissa piirin toimintaa analysoidaan ja eritellään piiriltä sellaisia osia, joiden vikaantuminen aiheuttaa eniten haittaa piirin toiminnalle. TMR-suojausta lisätään vain näihin piirin osiin. Näin piirin kokonaispinta-ala saadaan pienemmäksi, kuin TMR:n lisäämisellä sokeasti jokaiseen piirin osaan.

TMR pystyy takaamaan piirin oikean toiminnan vain yksittäisen vikaantumisen tapahtuessa. Useampien vikojen ilmaantuessa piiri voi alkaa toimia virheellisesti, ja scrubbing on hyvä suorittaa ennen kuin tällainen tilanne pääsee syntymään. Scrubbing ajastetaan yleensä suoritettavaksi tietyin aikavälein. Scrubbingien välinen aika pitää arvioida siten, ettei piiri ehdi vikaantua liikaa. Yksi suositus on, että scrubbingtaajuus olisi kymmenesosa arvioituun SEU-virheiden ilmenemistaajuuteen nähden. Jos halutaan suorittaa scrubbing vasta sitten, kun piiri vikaantuu, pitää piirille suunnitella ylimääräistä logiikkaa virheiden havaitsemiseen ja scrubbingin käynnistämiseen.

TMR on suoraviivainen ja toimiva tapa suojata SRAM-FPGA-piirejä yksittäisiltä säteilyn aiheuttamilta SEU-virheiltä. Kuitenkin TMR:n lisäksi tarvitaan scrubbingia estämään useampien virheiden syntymistä piirille.

LÄHTEET

Adell, P & Allen, G. (2008). Assessing and mitigating radiation effects in Xilinx FPGAs. Haettu 29.11.2011, sivulta http://hdl.handle.net/2014/40763

Almukhaizim, S & Makris, Y. (2008). Soft Error Mitigation Through Selective Addition of Functionally Redundant Wires. IEEE Transactions on Reliability, 57(1), 23 -231. doi:10.1109/TR.2008.916877

Caffrey, M, Graham, P, Johnson, E, Wirthlin, M, Rollins, N & Carmichael, C (2002). Single-Event Upsets in SRAM FPGAs. Julkaisu konferenssista Military and Aerospace Applications of Programmable Logic Devices (MAPLD).

Carmichael, C. (2001). Triple Module Redundancy Design Techniques for Virtex FPGAs. Haettu 22.11.2011, sivulta http://www.cs.york.ac.uk/rts/docs/Xilinx-datasource-2003-q1/appnotes/xapp197.pdf

Carmichael, C, Fuller, E, Blain, P & Caffrey, M (1999). SEU mitigation techniques for Virtex FPGA's in space applications. Julkaisu konferenssista Military and Aerospace Programmable Logic Devices (MAPLD) for 1999.

Edmonds, LD. (2009). Analysis of single-event upset rates in triple-modular redundancy devices. Haettu 11.11.2011, sivulta http://hdl.handle.net/2014/41123

Hentschke, R, Marques, F, Lima, F, Carro, L, Susin, A & Reis, R (2002). Analyzing area and performance penalty of protecting different digital modules with Hamming code and triple modular redundancy. Julkaisu konferenssista the 15th Symposium on Integrated Circuits and Systems Design (SBCCI'02).

Kamakoti, V, Chandrasekhar, V, Mahammad, SNM, V, Vijaykrishnan, N & Kamakoti, V (2005). Reduced Triple Modular Redundancy for Tolerating SEUs in SRAM-based FPGAs. Julkaisu konferenssista The 8th annual MAPLD International Conference.

Kastensmidt, FGdL, Neuberger, G, Hentschke, RF, Carro, L & Reis, R. (2004). Designing fault-tolerant techniques for SRAM-based FPGAs. IEEE Design Test of Computers, 21(6), 552 - 562. doi:10.1109/MDT.2004.85

Mitra, S & McCluskey, EJ (2000). Word-voter: a new voter design for triple modular redundant systems. Julkaisu konferenssista 18th IEEE VLSI Test Symposium.

Normand, E. (1996). Single event upset at ground level. Nuclear Science, IEEE Transactions on, 43(6), 2742 -22750. doi:10.1109/23.556861

Ramaswamy, S, Rockett, L, Patel, D, Danziger, S, Manohar, R, Kelly, CW, Holt, JL, Ekanayake, V & Elftmann, D (2009). A radiation hardened reconfigurable FPGA. Julkaisu konferenssista Aerospace conference, 2009 IEEE.

Samudrala, PK, Ramos, J & Katkoori, S. (2004). Selective triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs. IEEE Trans. Nucl. Sci., 51(5), 2957-2969. doi:10.1109/TNS.2004.834955

Xilinx. (2009). Xilinx TMRTool Product Brief. Haettu 4.11.2011, sivulta http://www.xilinx.com/publications/prod_mktg/XTMRTool_ssht.pdf

Xin, W (2010, november). Partitioning Triple Modular Redundancy for Single Event Upset Mitigation in FPGA. Julkaisu konferenssista 2010 International Conference on E-Product E-Service and E-Entertainment (ICEEE).

LIITTEET

LIITE 1: Äänestyslogiikka VHDL-kielellä

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY voters3x_g IS

     -- default width of the word:
     GENERIC(width: integer:=1);

     -- 3 inputs and 3 outputs:
     PORT(in1, in2, in3: IN std_logic_vector((width-1) DOWNTO 0);
          out1, out2, out3: OUT std_logic_vector((width-1) DOWNTO 0)
          );
END voters3x_g;

ARCHITECTURE beh OF voters3x_g IS
BEGIN
     PROCESS(in1, in2, in3)
     BEGIN

          -- same kind of voters for each bit of the word:
          FOR i IN (width-1) DOWNTO 0 LOOP

               -- 1st voter:
               out1(i) <= ((in1(i) AND in2(i)) OR
                           (in1(i) AND in3(i)) OR
                           (in2(i) AND in3(i)) );
               -- 2nd voter:
               out2(i) <= ((in1(i) AND in2(i)) OR
                           (in1(i) AND in3(i)) OR
                           (in2(i) AND in3(i)) );
               -- 3rd voter:
               out3(i) <= ((in1(i) AND in2(i)) OR
                           (in1(i) AND in3(i)) OR
                           (in2(i) AND in3(i)) );
          END LOOP;
     END PROCESS;
END beh;
```

LIITE 2: TMR VHDL-kielellä

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tmr_module_g IS
     -- default width of the word:
     GENERIC(width: integer:=4);
     -- 3 inputs, 3 outputs, reset and clock:
     PORT(in1, in2, in3: IN std_logic_vector((width-1) DOWNTO 0);
          out1, out2, out3: OUT std_logic_vector((width-1) DOWNTO 0);
          rst, clk: IN std_logic);
END tmr_module_g;

ARCHITECTURE beh OF tmr_module_g IS

     -- 3 parallel voters:
     COMPONENT voters3x_g IS
          GENERIC (width:    integer);
          PORT(in1, in2, in3: IN std_logic_vector((width-1) DOWNTO 0);
          out1, out2, out3: OUT std_logic_vector((width-1) DOWNTO 0)
          );
     END COMPONENT;

     -- internal signals:
     SIGNAL voter_to_reg1, voter_to_reg2, voter_to_reg3:
          std_logic_vector((width-1) DOWNTO 0);
     SIGNAL reg1_to_voter, reg2_to_voter, reg3_to_voter:
          std_logic_vector((width-1) DOWNTO 0);

BEGIN
     -- 3 parallel voters for the inputs:
     voters_log: voters3x_g  GENERIC MAP(width)
                             PORT MAP(in1, in2, in3, voter_to_reg1,
                                  voter_to_reg2, voter_to_reg3);
     -- 3 parallel voters for the outputs:
     voters_reg: voters3x_g  GENERIC MAP(width)
                             PORT MAP(reg1_to_voter, reg2_to_voter,
                                  reg3_to_voter, out1, out2, out3);
```

(jatkuu)

(jatkuu)

```
    -- 3 parallel registers in the middle:
    registers: PROCESS(clk)
    BEGIN

        -- synchronous actions only:
        IF rising_edge(clk) THEN

            -- synchronous reset (active low):
            IF (rst = '0') THEN
                FOR i IN (width-1) DOWNTO 0 LOOP
                    reg1_to_voter(i) <= '0';
                    reg2_to_voter(i) <= '0';
                    reg3_to_voter(i) <= '0';
                END LOOP;

            -- normal action:
            ELSE
                FOR i IN (width-1) DOWNTO 0 LOOP
                    reg1_to_voter(i) <= voter_to_reg1(i);
                    reg2_to_voter(i) <= voter_to_reg2(i);
                    reg3_to_voter(i) <= voter_to_reg3(i);
                END LOOP;
            END IF;
        END IF;
    END PROCESS registers;
END beh;
```