Feilfri konstruksjon av System-on-Chip

moderne konstruksjons-metodikk nyttar matematiske prov for å sikre kvaliteten

Av Joakim Urdahl

Dei raske framstega i produksjonen av halvledar-teknologi er kjend for alle med interesse for elektronikk. For å kunne utnytte potensialet dette gjev har konstruksjonsmetodane for maskinvare også måtte utvikle seg mykje. Ein krets vert difor ikkje lenger teikna transistor for transistor, men beskrive på eit funksjonelt nivå ved bruk av maskinvare-beskrivande språk som VHDL eller Verilog. At avanserte digitale system i dag er å finne i nær sagt eit kvart produkt er like mykje eit resultat av nye konstruksjonsmetodar som betre halvledar-teknologi.

Utviklinga har ført nye utfordringar med seg. Korleis kan vi verifisere at avanserte digitale system fungerar som dei skal?

Oppgåva er ein stadig veksande del av konstruksjonen, både økonomisk og tidsmessig. Alt i dag må ein rekna med at verifikasjon utgjer minst halvparten av konstruksjonskostnadene. Nye og meir effektive metodar også for verifikasjon er kravd om mikroelektronikk-revolusjonen skal kunne fortsette.

Feil i maskinvare kan ikkje rettast opp ved oppdateringar og nye versjonar slik som ved programvare. Før produksjonen vert sett i gang må difor tillita til korrekt konstruksjon være svært høg. Tilstrekkeleg verifikasjon er krevjande og kostbart, men samanlikna med konsekvensane av konstruksjonsfeil som ikkje vert retta kan det verke billig.

[Ref.: Intel's 700 Millionar dollar feil, http://newsroom.intel.com/community/intel_newsroom/blog/2011/01/31/intel-identifies-chipset-design-error-implementing-solution]

Ein moderne chip er eit komplett system i seg sjølv; og betegnes gjerne System on Chip (SoC). I ein SoC kan ein finne alle modular som tradisjonelt vert sett på som eigne komponentar, men modulane er her berre skilde på det funksjonelle nivået. Kvar modul er skreve i VHDL eller Verilog i eit abstraksjonsnivå for maskinvare kalla Register Transfer Level (RTL); modulane vert difor kalla RTL modular i det følgjande. I ein effektiv konstruksjons-metodikk er gjenbruk av RTL modular mykje nytta. Typisk vert modular henta frå tidlegare prosjekt eller kjøpt frå eksterne firma medan berre nokre få vert særleg utvikla for systemet. Dette gjev store utfordringar for verifikasjonen. Korleis kan ein verifisera at ein RTL modul fungerar i eit kvart miljø / system han skal kunne verte nytta i?

Tradisjonelt er verifisering einstydande med simulering. Simulering er å teste enkelt-scenarier og analysera kvart utfall. Sjølv for små system kan ikkje alle scenaria verte testa. Ein vel difor ut tilfella som ein håpar kan representere at heile funksjonar ved designa er korrekt. Om dette skal kunne fungere godt krev det full oversikt over funksjonalitet i systemet, og god kontroll på kva feil som kan oppstå. Moderne system er for store for at ein kan halde oversikta som krevjast. Hjørnetilfella som fort vert gløymde eller misforstått i konstruksjonen er dei same hjørnetilfella som fort vert gløymde i verifikasjonen. Eit anna grunnleggande problem er å avgjere når ein kan avslutte verifikasjonen. Eksisterande mål er mangelfulle og garanterar ikkje at viktige hjørnetilfelle vert dekka.

Arbeidet med formelle metodar for verifikasjon har gått for seg lenge i akademia, men i industriell samanheng er simulering framleis klart dominerande verifikasjonsmetode. Formell verifikasjon er å bevise eigenskapar ved systemet ved å nytte formell logikk. Mykje av grunnen til at formelle metodar ikkje er meir brukt er at dei har vert for vanskelege å nytte for reelle industrielle system og at dei har lidd under dei same problema som simulering når det gjeld dekningsgrad. Berre i sikkerheitskritiske miljø, og for "general purpose" prosessorar som vert produsert i svært store mengder har ein kunne teke seg råd til ein meir utstrakt bruk.

Joakim Urdahl, artikkelforfatter og vinner av Mikroelektronikkprisen 2010.

Algoritmisk kompleksitet er største utfordring ved formelle metodar. Kompleksiteten aukar eksponensielt med talet på tilstands-bit (for ein krets er dette signal og register). Model checking er grunnlaget for dei fleste brukte formelle metodar, men dagens industrielle design er typisk svært mykje for store for å nytte model checking direkte

[Ref: E. M. Clarke, O. Grumberg og D. A. Peled, "Model Checking" MIT Press, 1999].

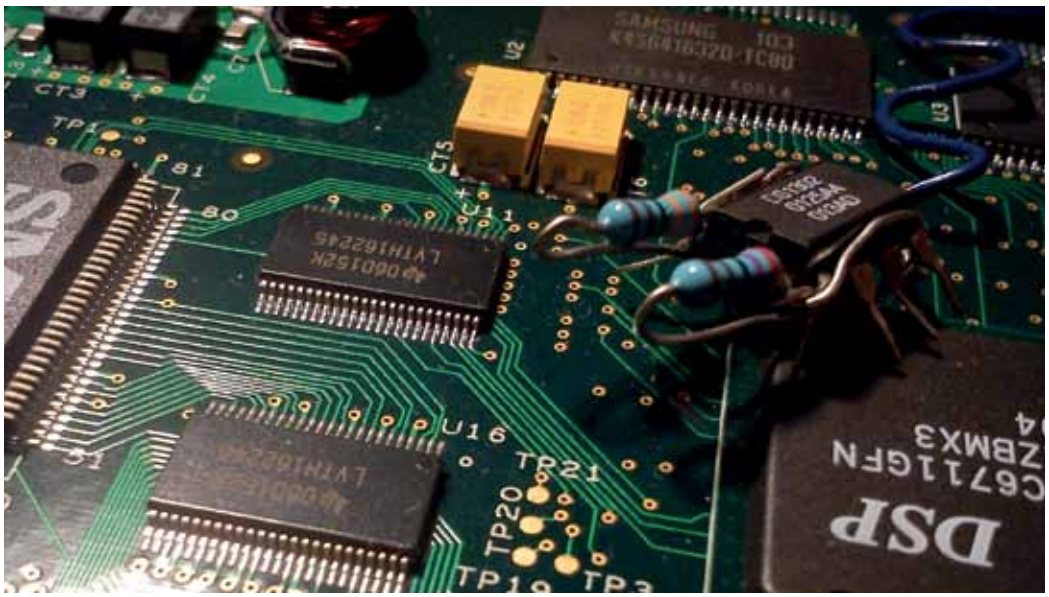
I oppgåva mi har eg studert ein formell metode kalla Interval Property Checking (IPC). Metoden vart oppfunne ved Siemens alt på midten av 90talet; sidan den gong er metoden utvikla mykje, og er no kommersielt tilgjengeleg ved OneSpin Solutions [Ref: URL: http://www.onespin.com]. IPC skalerar for reelle industrielle design og er intuitivt å nytte for ingeniørar med bakgrunn frå kretskonstruksjon.

Eigenskapar som skal provast med IPC må være temporære logiske uttrykk av typen "safety property" formulert over faste tidsintervall. Eit enkelt døme på ei slikt uttrykk er korrekt addisjon.

t representerar eit kva som helst tidspunkt i kretsen, medan t+1 er ein takt etter t. Operation, Output, op1 og op2 er signal i kretsen.

```
assume:
  at t: Operation = ADD_C;
prove:
  at t+1 Output = prev(op1) +
prev(op2)
```

Ideen bak IPC er å utnytte eigenskapar som er typisk for maskinvare til å forenkle den matematiske oppgåva. Mykje av



kompleksiteten i model checking er å finne kva tilstandar som kan nåast i kretsen. IPC har ein radikalt ulik tilnærming. I IPC ynskjer ein at eigenskapane som skal verte bevist er forma som operasjonar av kretsen. Tanken er at utføringa av slike operasjonar er ganske uavhengige av kva tilstand kretsen var i då operasjonen vart starta. For å syne prinsippet, addisjon er eit naturleg døme på ein operasjon for ein ALU krets. Resultatet av addisjonen er avhengig utelukkande av verdiane til operandane, ikkje verdiar av eventuelle interne variablar.

Uttrykk vert prova i IPC om negasjonen av uttrykket ikkje kan oppfyllast uansett verdi på tilstandsvariablar (om dei ikkje er del av uttrykket som skal verte bevist). Dette er ei overestimering av start-tilstandar (kva samtidige verdiar som er mogleg er avgrensa). Om negasjonen av uttrykket kan oppfyllast kan det være to orsakar. Uttrykket er ikkje gyldig for kretsen (kalla "true negative"). Negasjonen av uttrykket kan oppfyllast berre på grunn av overestimeringa av start-tilstandar ("false negative"). Merk at overestimeringa ikkje kan føre til det motsette, at falske uttrykk feilaktig vert prova. Om uttrykk vert forma som operasjonar av kretsen vert problem med "false negatives" minimert. I nokre tilfelle kan problema likevel inntreffe, ein må da eksplisitt prova at start-tilstandane som fører til at uttrykket ikkje held ikkje er moglege. For ein ingeniør med kunnskap om kretsen er det ei oversynleg oppgåve løyst ved å analysere dømet som oppfyll negasjonen av uttrykket.

Eit gjennombrot for metoden kom nyleg og er ei utviding som gjer at ein kan prova at heile åtferda til kretsen er dekka av IPC uttrykk /operasjonar **[Ref: Jörg Bormann. Vollständige funktionale Verifikation. PhD thesis, University of Kaiserslautern, 2009.].**

Også ein metodikk for å oppnå full dekning er utvikla. Verifikasjon i hendhald til denne metodikken ligg nært måten for konstruksjon. Ved avslutta verifikasjon resulterar eit sett av operasjonar som beskriv heile kretsen. For best mogleg verifikasjon skal desse operasjonane ligge so nært opp mot abstraksjonsnivået til spesifikasjonen som mogleg.

Dei resulterande operasjonane kan sjåast på som ei ny, fullstendig og korrekt modell av kretsen. Dette gjev ein ny innfallsvinkel å forstå kretsen utifrå, eit ortogonalt syn på designet som passar godt med korleis ein naturleg abstraherar kretsen. Ved universitetet i Kaiserslautern har vi undersøkt om og korleis denne nye modellen kan verte nytta til å gjere formell verifikasjon av heile system mogleg. For å kunne mogleggjere dette har vi formalisert ein tidsunøyaktig abstraksjon basert på operasjonar [Ref: J. Urdahl, D. Stoffel, J. Bormann, M. Wedler og W. Kunz, "Path Predicate Abstraction by Complete Interval Property Checking" i Proc. International Conference on Formal Methods in Computer-Aided Design (FMCAD). IEEE Computer Society, 2010, pp. 207–215.].

I arbeidet med master-oppgåva mi har eg synt at det er mogleg å utarbeide slike abstraksjonar. For Infineons FPI bus klarte vi å utvikle abstraksjonar som gjorde at vi kunne bevise eigenskapar ved heile det samansette systemet. Merk at dette er eit oppsiktsvekkande resultat fordi bevisa er gyldige ikkje berre for ein modell av systemet, men for den reelle implementerte kretsen. I pågåande arbeid undersøker vi korleis desse abstraksjonsmetodane kan verte gjort generiske. Om arbeidet lykkast vert dette ein klar framgangsmåte for korleis slike abstraksjonar kan vert utvikla for ein kvar modul på ein slik måte som garanterar at eit prov for system av abstraksjonar også er eit prov for det verkelege systemet.