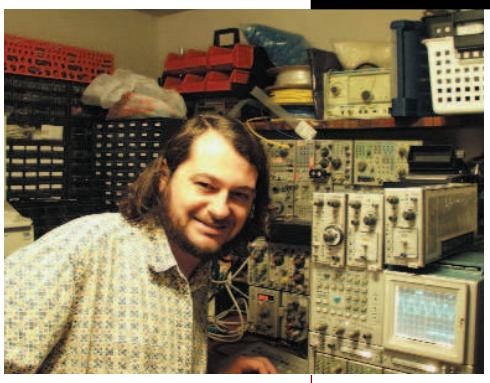
Icarus – Tervezzünk áramköröket!



„Ebben a finomítási folyamatban a kód makrókra emlékeztet, és a tervezők egyszerűsíthetik is a feladatukat, ha RTL logikai könyvtárakat használnak.”

– Michael Baxter

A nyílt forrású programok egyre inkább túlmutatnak az operációs rendszerek, az internetes és az asztali alkalmazások, valamint a felhasználói felületek és parancsnyelvek világán. Két ilyen kevéssé ismert terület az elektronikus tervezéstámogatás (Electronic Design Automation – EDA) és az alkatrészleíró nyelvek (Hardware Description Language – HDL). A két leggyakrabban használt alkatrészleíró nyelv a VHDL és a Verilog. Ez utóbbit széles körben alkalmazzák logikai áramkörök tervezésére és utánzására a félvezetőiparban és más területeken.

A HDL-nyelvek az alkatrészek és gépek utánzására, összehasonlítására készültek, ez az adott felhasználástól függően különböző elvonatkoztatási szinteken történhet. E szintek tanulmányozásával megérthetjük, hogy miben különböznek a HDL-nyelvek a C, C++, Java és más, „hagyományos” programozási nyelvektől.

Egy „mi történik akkor, ha” típusú utánzás során a gép minden egyes részelemének működését megpróbáljuk a HDL eszközeivel jellemezni, tehát a feladatára vagyunk kíváncsiak, és nem arra, hogy ezt hány ellenállás vagy tranzisztor valósítja meg. Ez a fajta HDL-kód leginkább egy hagyományos számítógépprogramra emlékeztet.

A HDL-nyelvekben használt középső szint a Register Transfer Level (RTL). Ez a kód a regiszterek szintjén megvalósított szerkezeti felépítést írja le. A regisztereket flip-flopok vagy más logikai szerkezetek (például latchek – kallantyúk) alkotják, ezek különböző logikai tulajdonságokkal bírhatnak. A logikát leíró kód lehet viselkedésközpontú, de foglalkozhat azzal is, hogy mi kerül majd a tényleges áramkörre. Az RTL-t először a rendszer főbb elemeinek fölvázolásához, majd a végleges változat részleteinek kidolgozásához használják. Ebben a finomítási folyamatban a kód makrókra emlékeztet, és a tervezők egyszerűsíthetik is a dolgukat, ha RTL logikai könyvtárakat használnak. A logikai áramköröket a létező legalacsonyabb szinten is megtervezhetjük, így a kódban az áramkör pontos szerkezetét határozzuk meg. Ezt szerkezeti tervezésnek hívják, és leginkább a gépi kódú nyelvekre emlékeztet.

Egyetlen programnyelvvel minden szinten dolgozhatunk, és ezeket keverhetjük is egymással. A HSL nyelvek és a programnyelvek között van még egy nagy különbség: az idő. Bizonyos szempontból a HDL nyelvek valósítják meg tökéletesen a „programszámláló” elvét, hiszen ezeknél alapvető fontosságú az idő modellezése azért, hogy a logikai áramkörök viselkedését pontosan elemezhessük. Ebből következik, hogy a HDL-típusú és a hagyományos programnyelvek jelrendszere között óriási különbségek vannak. A HDL-nyelvek a párhuzamos rendszerműveleteket is tökéletesen kezelik. A teljesítmény növeléséért a HDL-fordítókat általában C vagy C++ nyelven írják meg. A fordító azonban a HDL-nyelv olyan elemeinek használatát is lehetővé teszi, melyek magvát a párhuzamos műveletvégzés alkotja, hiszen ez a jellemző igazán a gépre.

Mindezek eredményeképpen a Verilogot használó EDA eszközök a hagyományos programból megvalósított eszközöknél sokkal gazdagabb lehetőségeket kínálnak. Hogy mindezt még érthetőbbé tegyük, az alábbiakban közlünk egy beszélgetést *Stephen Williamsszel,* a GPL elvei szerint terjeszthető Icarus Verilog nevű EDA eszköz fejlesztőjével.

Michael: *Mi az Icarus Verilog fordító és hogyan működik?*

Stephen: Az Icarus Verilog az IEEE 1364-es szabványú Verilog alkatrészleíró nyelvhez készült fordító. Az EDA szakterületen jártas felhasználók nyilván felfedezik a hasonlóságot az Icarus és a VCS között, hiszen mindkettő széles körben elfogadott nyelv fordítója.

Működésének lényege, hogy a Verilog formanyelvét egy megjegyzésekkel kiegészített feldolgozási fába olvassa be, melyet azután egy tervezőgrafikonba „olvaszt be”.

E folyamat során történik a modulok példányosítása, a szimbolikus állandók kiértékelése és ismertetése, majd a rendszer ellenőrzi az eszközök csatlakoztatását. A folyamat végén egy kipróbált, működő rendszert kapunk.

A tervezőgrafikonból az optimalizálók és logikai összerendező eljárások egy új ábrát készítenek, mely a célnak jobban megfelel. Ezt végül a kódelőállító nézi át.

A folyamat végén a kódelőállító átvizsgálja a tervet és a kívánt formában menti a kimenetet. Az utánzás céljára C++ kód jön létre, ez az Icarus Veriloghoz tartozó különleges osztálykönyvtárra építkezik. A csomag egy XNF kódelőállítót is tartalmaz, ezzel a kész terveket más FPGA eszközökhöz továbbíthatjuk. Mostanában egy betölthető célkód-előállítón dolgozom, ezzel számos más célformátum támogatása is megvalósulhatna.

Michael: *Mikor kezdted az Icarus Verilog fejlesztését?*

Stephen: Jegyzeteim alapján 1998. novemberében került be a CVS-be a terv, de már legalább két évvel előtte megpróbáltam a munka beindítását. Ha emlékezetem nem csal, akkor már egy évvel a CVS-be kerülést megelőzően megtaláltam a követendő utat.

Michael: Mi ösztönzött arra, hogy az Icarus Verilog fejlesztésébe vágd a fejszédet?

Stephen: A szónoki válasz az lenne, hogy Linux/Alpha-rendszerem van. A legtöbb ember számára ez nem sokat mond. A lényeg, hogy az EDA-fejlesztők nem árasztják el programokkal a Linux/Alpha-rendszert használókat, de még a Linux/Intel felület sem kap megfelelő figyelmet. Egyszerűen nem tudom elképzelni, hogy az a sok EDA-fejlesztő miért szereti annyira a Microsoft termékeit.

A másik ok már valamivel összetettebb. Azért kezdtem neki, mert tudtam, hogy képes leszek rá, és jó munkát fogok végezni. Képességeim elegendőnek tűntek egy ilyen feladat végrehajtásához, ráadásul tovább bővültek az ismereteim. Úgy kezdtem bele, hogy sokkal többet tudtam a lapka- és az FPGA-tervezésről, mint a programmérnökök legtöbbje.

Michael: *Miért ragaszkodsz a nyílt forrású fejlesztéshez?*

Stephen: Szeretnék hozzáférni saját szellemi tulajdonomhoz. Régebben rengeteg nagyobb fejlesztési munkában vettem részt, ezek eredményeit azonban munkahelyvál-

tás vagy különleges megállapodások miatt nem használhattam föl. A jelenlegi munkáltatóm azonban semmit sem kötött ki az Icarus Verilog fejlesztésével kapcsolatban, tehát elfogadja, hogy ez az én munkám, amit a saját időbeosztásomban végezhetek. Ezt a szerzői jogi rendelkezések is nyilvánvalóvá teszik, és eddig úgy tűnik, hogy ez így minden résztvevő számára biztonságosabb.
A munkaadóm üzleti terveinek mellőzésével az egészet készíthetném zárt forrású, személyes munkaként is, de mi lenne abban az élvezet? És azt se felejtsük el, hogy a nyílt forráskódnak köszönhetően egyre több hasznos hibabejelentést kapok a programmal kapcsolatban. Néha még egész programrészletek is érkeznek, melyek új lehetőségeket valósítanak meg.

Michael: *Kik segítettek neked a legtöbbet?*

Stephen: A csomagban található README fájlból mindez kiderül, de talán a próbaüzem felállításában segédkező *Steve Wilson* segítsége nélkül lett volna a legnehezebb. Sokszor mondták, hogy a kipróbálás a fordítók fejlesztésének legfontosabb része, és eddigi tapasztalataim alapján igazat kell adnom mindenkinek.
Ismétlem, a felhasználóktól kapott hibabejelentések is óriási segítséget jelentettek. Ezek mindig jó minőségűek, részletesek, és a legtöbb esetben időszerűek. Nagyon-nagyon ritkán fordult elő, hogy egy felhasználó által beküldött hibabejelentés kivizsgálásakor az derült ki, hogy a hiba mégsem az Icarus Verilog „készülékében" van. Ha mégis, akkor általában valami egyszerű félreértésről volt szó.
A hibabejelentések és a változtatási kérelmek, javaslatok alapján döntöm el azt is, hogy mi lesz a fejlesztés következő lépése.

Michael: *Melyek az Icarus jellegzetes felhasználási területei?*

Stephen: Nos, erre nehéz válaszolnom, hiszen nem áll a hátam mögött egy piackutató részleg, mely ennek kiderítésével foglalkozik. Legfőbb forrásom e tekintetben is a felhasználóktól – általában hibabejelentések formájában – érkező visszajelzések.
Úgy tűnik számomra, hogy a nagy intézményekben dolgozó felhasználók nem férnek hozzá a méregdrága HDL-fejlesztőkörnyezethez, így az Icarus Verilog segítségével otthoni linuxos gépükön dolgoznak különböző könyvtárakkal és alrendszerekkel.
Számos olyan esetről hallottam, amikor valaki az Icarus megjelenése következtében anyagi vagy elvi okokból abbahagyta a díjkötelezett eszközökkel végzett munkát. A szerényebb igényű HDL-felhasználók azért szeretik az Icarus Verilogot, mert céljaiknak tökéletesen megfelel, az árversenyben pedig verhetetlen. A csomag azok kezébe adja a HDL-es tervezés lehetőségét, akik máskülönben biztosan nem engedhetnék meg magunknak.

Michael: *Összehasonlítható-e az Icarus egy „fizetős" EDA-eszközzel?*

Stephen: Az Icarus Verilog nem jelent veszélyt a minőségi, márkás eszközökre, hiszen ezek fejlesztői nálam sokkal hamarabb kezdték a munkát. Nemrég még azt gondoltam, hogy én a kis Icarusommal föl sem vehetem velük a versenyt. Manapság azonban egyre több olyan fizetős eszközt találtam, amelyeknek piaci jelenlétét szerintem az égvilágon semmi sem teszi szükségessé. A különbségek a nyelv elemeinek megvalósításában, az utánzás teljesítményében és az összerendezés (synthesis) minőségében jelentkeznek. Ami a nyelv megvalósítását illeti, az Icarus Verilog aránylag jól áll a versenyben, és a fejlődés e téren folyamatos. A jelenlegi szint pedig átlagosnak mondható.
Az utánzás teljesítményét nagymértékben rontja a g++ fordító, amely az előállított utánzást fordítja le. Attól tartok, hogy hiba volt a C++-t választanom közvetítő nyelvként, szóval hamarosan sima C-re vagy valami másra cserélem le. A terv a fordítás után azonban meglepően jól és gyorsan működik.
Az Icarus Verilogban megvalósított összerendezés még nem üzleti minőségű, bár jónéhányan eredményesen használják. Az Icarus Verilog összerendezőjének határain belül maradva akár Xilinx-típusú terveket is készíthetünk. Tudok olyan esetről is, hogy valaki az Icarusszal helyettesítette az Abelt a tervezési folyamatban.

„A nyílt forrás hatásaként rengeteg értékes hibabejelentést kapok. Rendkívül örülök a felhasználóktól érkező hibabejelentéseknek."

– Stephen Williams

Michael: *Létezik-e olyan feladat, amit az üzleti EDA-eszközökhöz képest az Icarusszal nehezebb megvalósítani?*

Stephen: Először is, az Icarus Verilog nem képes mindenre, amire egy EDA-felhasználónak szüksége lehet. Természetesen képes XNF-ben menteni, de a használni kívánt rész feltérképezéséhez és optimalizálásához a gyártótól függően más és más eszközöket kell igénybe vennünk. Nagyon nehéz beszerezni a gyártóktól a megfelelő adatokat, ezért a Netlist formátumokat kell használnom, ez pedig igencsak kiábrándító. Az is bosszant, hogy Linuxra nem léteznek mélyebb rétegekkel foglalkozó eszközök.

Michael: *Az Icarus Verilog nyílt forrású eszmeisége jelent-e valamiféle előnyt az üzleti eszközökkel szemben?*

Stephen: Az Icarus legnyilvánvalóbb előnye, hogy rugalmas munkakörnyezetet teremt. Ha egy terv működik az Icarusszal, akkor egy új számítógép vásárlása esetén is biztosak lehetünk abban, hogy azon is működni fog. Megfigyeltem, hogy az EDA-fejlesztők korlátlan időre szóló szerződéseket hirdetnek, a programot futtató operációs rendszerre és a számítógépre azonban ez nem igaz. Ráadásul X gyártó nem támogatja Y termék 1.0-s változatát a frissen vásárolt gépünkön. Ebből következik, hogy új gép vásárlásakor frissítéseket is kell vennünk, ez nemcsak méregdrága mulatság, hanem még kockázatos is, mondjuk egy 95 százalékban kész XC4013XL tervezés esetén. Láttam már olyat, hogy egy eszköz újabb változata tönkretette a majdnem kész tervet.

Michael: *Az üzleti EDA-fejlesztők lassan közelednek a Linuxhoz, bár néhányan már használják. Azonban kivétel nélkül viszolyognak a nyílt forrású fejlesztések befogadásától vagy indításától. Vajon miért?*

Stephen: Nos, azt hiszem, a cégek számára elég rázós lenne egy százezer dolláros program kifejlesztése után

„Egyszerűen nem tudom elképzelni, hogy számos EDA-fejlesztő miért szereti annyira a Microsoft termékeit."

– Stephen Williams

a nyílt forrású fejlesztésről ódákat zengeni, egyébként meg nem is tudom. Minden munkatársam nap mint nap panaszkodik, hogy Microsoft operációs rendszerekkel kell dolgozniuk, mert nincsen más választásuk. Az FPGA-gyártók kifejezetten mogorvák e tekintetben. Talán, ha majd ott is beköszönt a nyílt forrású fejlesztés tavasza...

Michael: *Mennyi kód van az Icarusban?*

Stephen: 50 000 C++ nyelvű sor, egy kis C-vel, lexszel és yacc-cal megtűzdelve. Úgy tűnik, hogy jó néhány hónapja nem hajlandó ennél nagyobbra nőni... De komolyra fordítva a szót: eljutottam arra a szintre, hogy legalább annyi kódot távolítok el belőle, mint amennyit hozzáteszek. Van egy kis próbalabor is, ahol 300 kisebb Verilog-ellenőrzés folyik, ez összesen 16 000 Verilog-sort érint. Ezeket egy kis Perl vezérli.

Michael: *Van valami ötleted, javaslatod a nyílt forrású programfejlesztők számára, mely elősegíthetné az Icarushoz hasonló munkák elindítását?*

Stephen: Nos, jó lenne, ha a g++ mondjuk, tízszer ilyen gyorsan fordítana. Nem is tudom, miért panaszkodom, hiszen például az MSVC++ egyszerűen képtelen lefordítani a fordítómat. A Linux/Alpha és a Cygwin32 binutils esetében a szimbólumtáblákkal is akadtak gondok.

Michael: *Az Icarus fejlesztésének alapját a Linux képezi. Nehéz feladat volt más operációs rendszerekre átültetni?*

Stephen: Segítőim az utolsó működő változat előre fordított bináris fájljait átültették Solarisra, NetBSD-re és MacOS-re is. Nemrégiben egy windowsos változat is elkészült, ezt a Cygwin32 Net kiadásának köszönhetjük. Az átalakítások legnehezebb részét mindig a dinamikus kapcsolások képezték – a HP/UX különösen makacs természetűnek bizonyult e téren –. Azért akinek van egy korszerű C++ fordítója, meg egy make programja, annak nem sok baja lesz az Icarus Verilog fordításával. A FAQ oldalon igyekeztem felsorolni a gyakoribb hibákat és azok kiküszöbölésének módját.

Michael: *Megemlítenél néhány olyan nyílt forrású EDA-eszközt, amelyek érdekelhetik az Icarus felhasználóit?*

Stephen: Ott van például a gEDA csomag, ennek honlapján ➲ http://www.geda.seul.org/, fellelhetünk számos érdekes EDA-eszköz oldalaihoz vezető hivatkozást. A GTKWave egy szintén értékes hullámforma-megjelenítő. Jól használható az Icarus VCD kimenetének tanulmányozására. Az Electronic VLSI Design System ➲ http://www.staticfreesoft.com/ is nagyon érdekes, és ami a legszebb, hogy Linux/Alpha alatt is működik. Egy mutatós EDA-lista található az Open Collector weboldalán ➲ http://www.opencollector.org/. Aki itt szétnéz egy kicsit, az rengeteg aranyos dolgot találhat.

Michael: *Tervezed-e, hogy az Icarust más HDL-nyelvek támogatásával is kiegészíted?*

Stephen: Egyelőre nem. A Verilog szabványosítási folyamattal kapcsolatos új adatok beépítése is éppen elég feladatot ró egyetlen emberre. Ha esetleg valamilyen véletlen folytán mégis „utolérném" a nyelvet, a fordításban olyan új nehézségek adódhatnak, melyek egy jó darabig ellátnak munkával.

Michael: *Hogyan képzeled el az Icarus jövőjét?*

Stephen: Ugyanazt szeretném látni, ami a gcc esetében is történt. Ha valaki idejön, és azt mondja, hogy sok pénzt fizet, ha folytatom az Icarus Veriloggal elkezdett munkát, annak nagyon fogok örülni...

Michael: *A munka mely részterületein van szükséged segítségre?*

Stephen: Kódelőállítókat kellene készíteni! Nagyon sok fáradságot jelent a kódelőállítók által használt API-k csiszolgatása. Legmerészebb álmom, hogy készítek néhány alapvető kódelőállítót, majd a segítőim az összes létező Netlist formátumhoz és utánzómotorhoz megírják a megfelelő kódelőállítót. A gcc is körülbelül így működik. Kipróbálók kellenek! Hiányukat leggyakrabban a hibabejelentések elemzésével hidalom át, de sokszor célszerűbb, hogy külön próbakörnyezetbe helyezzek egy-egy frissen kifejlesztett elemet. De nyilván én nehezebben találok meg egy hibát a saját kódomban.

Michael: *Véleményed szerint a nyílt forrású eszközök milyen szerepet játszhatnak az EDA-eszközök fejlődésében?*

Stephen: Nem igazán tudom, hiszen nem vagyok jós. Annyi bizonyosnak látszik, hogy hatásukra emelkedni fog az üzleti csomagok színvonala. Úgy vélem, a nyílt forrású eszközökben megvan a lehetőség arra, hogy a régi tervekkel is dolgozhassunk. A terv forrását tárolhatjuk, de a régi eszközök tárolásának semmi értelmét nem látom.

Michael: *Ikarus élete gyászos véget ért. Számodra van valami jelentősége a névválasztásnak?*

Stephen: Meglepődnél azon, hogy mennyire kevesen ismerik a történetet. „Icarus, hogyan kell leírni?" – általában ez az első kérdés. A név inkább apró utalás, nem a jelentése a lényeg. Én eredetileg programmérnök vagyok, nem géptervező. Na jó, tudom kezelni az oszcilloszkópot meg a logikaelemzőt, és 160 pontos csatlakozók tűire is forrasztgattam huzalokat, ennek ellenére én csak egy programmérnök vagyok, aki túl közel repül a Naphoz. Sokszor mondták, hogy egy Verilog-fordító elkészítése nagyobb munka, mint ahogy én azt képzelem. Ezt általában géptervező mérnökök mondogatták. De mostanában már nem hallok ilyeneket. Tudom, hogy korábban milyen tudásbéli hiányosságokkal kellett szembenéznem, de mára azért változott a helyzet.

Michael: *Mesélnél valamit a logóról?*

Stephen: Természetesen. Steve Wilson lényegesen részletesebben tudna róla mesélni, de röviden összefoglalva Steve nagybátyja, *Charles Wilson* nyugdíjas grafikus rajzolta. Művét ingyen felajánlotta az Icarus Verilog számára, és ezért én igen hálás vagyok neki. Soha nem használtunk más jelet. Ez újabb bizonyíték arra, hogy a nyílt forrású mozgalom a számítógépprogramok körén kívüli területeket is meghódíthatja.

Michael Baxter 1969-ben látta a 2001. Űrodüsszeia filmváltozatát, és azóta, azaz kilencéves korától foglalkozik számítógépekkel. Tapasztalt számítógépes mérnök: rendszer-, kártya- és FPGA logika-tervező. Michaelnek tíz bejegyzett szabadalma van.