ANNA

17. veljače 2011.

1 Kostur

Kostur sačinjavaju sučelja koja se nalaze pod hr.anna.interfaces. Sučelja predstavljaju očekivanja i mogućnosti svake od komponenata.

Protokoli koje kostur omogućava su neovisni o implementaciji. Za primjer potpunosti uzmimo trojku IBusMaster, IBus i IBusUnit. IBusMaster može pristupati jedinicama na sabirnici preko sljedećeg protokola:

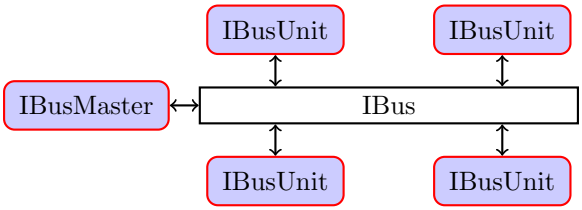
IBusMaster poziva IBus.requestWrite i preko nje postavlja zahtjev za čitanje. Sabirnica obavlja translaciju adrese i dodatne operacije potrebne za pristup i prosljeđuje zahtjev preko IBusUnit.requestWrite. Jedinica na sabirnici nakon obavljene operacije uzvraća preko IBus.busUnitWriteCallback. Sabirnica opet obavlja određene translacije i na kraju dostavlja informaciju preko IBusMaster.busWriteCallback. Protokol je ilustriran na slici 1.

1.1 Kritike i poboljšanja

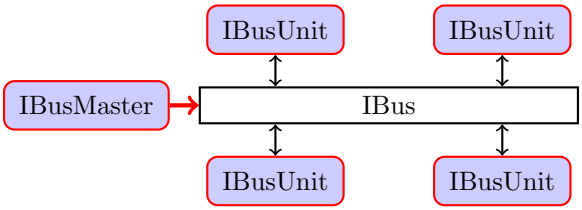
Protokol je većinom prilagođen za komunikacije na sabirnicama gdje postoji jedan ili par upravljača (mastera) i više jedinica. Pozitivna stvar ovoga je što je većina internih protokola master-slave koncepta, a i neki eksterni protokoli. Ova sučelja trenutno ne predviđaju složene arbitracije i to je jedno od područja koje se može dodatno razvijati.

Sučelja također omogućuju bridgeve među sabirnicama. Bridge jedinica je za jednu sabirnicu samo IBusUnit, a za drugu IBusMaster.

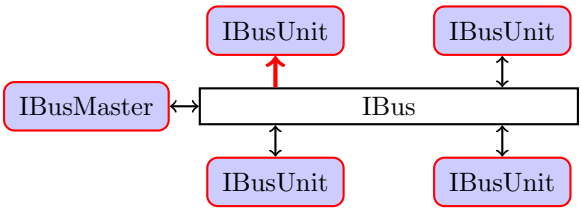
Dodati tablicu koja nudi summary protokola koje je moguće/nije moguće implementirati



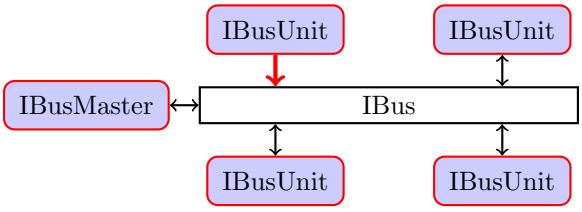
(a) Prije početka komunikacije



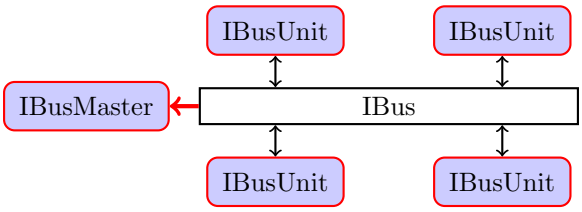
(b) IBus.requestBusWrite(IBusMaster,Address,Data)



(c) IBusUnit.requestBusUnitWrite(IBus,TranslatedAddress,Data)



(d) IBus.busUnitWriteCallback(IBusUnit,TranslatedAddress)



(e) IBusMaster.busWriteCallback(IBus,Address)

Slika 1: Protokol za sabirnicu (pisanje); slično vrijedi i za čitanje samo što se onda podaci prenose preko callback metoda

2 Word

Razred Word omogućuje prikaz riječi proizvoljne širine u bitovima. Podržane su logičke operacije, operacije zbrajanja i oduzimanja i operacije pomaka.

Slika 2: Operacija zbrajanja

Riječi su rastavljene na blokove od 32-bita. Koristi se maksimalno 24 od 32 bita kako bi se omogućilo byte poravnavanje. Prilikom operacija zbrajanja i oduzimanja operacije se vrše blok po blok. Operacije zbrajanje i oduzimanja omogućuju unos carry bita kao i overflow bit koji izlazi van.

3 Mikroinstrukcije

Zadatak dekodera instrukcija je generacija kontrolnih signala unutar procesora. Jedan ovakav općeniti dekoder se može razbiti na 2 manja dekodera:

1. Dekoder instrukcije u jednu ili više instrukcija (nazovimo ih mikroinstrukcije)
2. Dekoder mikroinstrukcija u kontrolne signale unutar procesora

Povijesni razlog uvođenja mikroinstrukcija je bio olakšavanje dekodiranja kompleksnih instrukcija. Mikroinstrukcije (mikrokod općenito) su olakšali rješavanje sljedećih problema kod CISC procesora:

- Kako je set instrukcija postajao sve kompleksniji, sve je teže bilo pronaći efikasno kodiranje kojim bi se olakšala aktivacija podatkovnih putova unutar procesora
- Paralelizacija koju je moguće napraviti često nije bila očita tokom dizajna procesora
- Kompleksna povezanost između seta instrukcija i podatkovnih putova
- Reprogramabilnost u slučaju pronađenog kvara (Intel Core 2)

Za RISC i VLIW procesore postoje argumenti za i protiv uporabe mikroinstrukcija, međutim, mi ih koristimo radi jednostavnosti i dosljednosti arhitekture. Najčešće će za RISC procesore vrijediti preslikavanje 1 na 1. VLIW procesori kodiraju više jednostavnih operacija unutar jedne instrukcije i stoga ovdje opet očekujemo preslikavanje 1 na 1 za svaku operaciju.

Mikroasembler (IInstructionDecoder u našem slučaju) rastavlja ulaznu instrukciju na mikroinstrukcije koje mikromašina unutar procesora izvršava. U HW-u je to realizirano na način da postoji mikrosekvencer koji iterira po mikroinstrukcjima i izvršava ih. Mikroasembler odgovara prvom dekoderu, dok mikrosekvencer predaje rezultat prvog dekodera 2. dekoderu. Mikrosekvencer i 2. dekoder se mogu grupirati u mikromašinu.

Ovisno o tome da li je potrebno naknadno dekodiranje prije nego se kontrolni signali generiraju, postoje horizontalne i vertikalne mikroinstrukcije. Vraćajući se na početno razbijanje na 2 dekodera, horizontalne mikroinstrukcije imaju trivijalni oblik 2. dekodera. U programskoj realizaciji nije moguće napraviti ovakvu podjelu (programska realizacija već sama po sebi omogućava i koristi prekompleksno dekodiranje) pa je stoga i nećemo.

Popisati ih, LLVM kao najbliža referenca

Očekivane prednosti jednake su prednostima koje VM-ovi trenutno nude jer su mikroinstrukcije tandem IL, a mikromašina tandem VM. Kao što VM i IL bytecode nude portabilnost, tako i mikroinstrukcije apstrahirane ovdje trebaju nuditi presliku bilo kojeg instrukcijskog seta u jedan predefinirani set naredbi koje će naše ponuđene mikromašine znati izvršavati. Pri tome se jedna instrukcija iz seta preslikava u jednu ili više mikroinstrukcija.

Programski su realizirane preko Command patterna. Trenutno su realizirane sljedeće:

AddMicroinstruction zbraja 2 registra i rezultat sprema u registar za rezultat

SubtractMicroinstruction oduzima 2 registra i rezultat u registar za rezultat

ConditionalMicroinstruction izvršava određenu podmikroinstrukciju u slučaju da je zadovoljen neki od uvjeta

JumpMicroinstruction skače na određenu lokaciju

MoveMicroinstruction vrijednost jednog registra sprema u drugi registar

LoadMicroinstruction učitava iz memorije (s određene adrese) u određeni registar

StoreMicroinstruction sprema u memoriju na određenu adresu iz određenog registra

3.1 Poboljšanja

Treba izdvojiti općenitu implementaciju mikromašine (nazovimo je IMicroMachine ili IExecutionUnit) tako da je dovoljno samo je uključiti u procesor za omogućiti rad. Tada je također moguće i specijalizirati mikromašinu da izvršava samo određene mikroinstrukcije (podskup), a ostale prosljeđuje. Nastavak na ovo su razne kombinacije grafove koje će predstavljati serijske/paralelne/kombinirane podatkovne putove.

Za dodati pipeline treba dodati i posebnu instrukciju za flush pipeline kada se dogodi hazard koji nije moguće obraditi. Nakon toga, dovoljno je samo pospojiti specificirane mikromašine u seriju stupnjeva koliko ih pipeline ima.

Može se dodati poseban IDispatcher čija je služba raspodjeljivanje posla kada postoji više paralelnih mikromašina koje mogu izvršavati isti podskup mikroinstrukcija.

Trenutne mikroinstrukcije bi se možda trebale spustiti razinu niže tako da uistinu predstavljaju podatkovne puteve (prema RTL - register transfer level). Primjerice, JumpMicroinstruction bi se mogla razbiti na MoveMicroinstruction odrediša u PC i LoadMicroinstruction sa adrese PC u IR. Eventualno bi se trenutne mikroinstrukcije mogle ostaviti kakve jesu (jer služe svrsi savršeno), a dodati novi paket koji bi omogućio bolju reprentaciju podatkovnog puta (stavljanje vrijednosti u registar, aritmetičke operacije nad registrom itd...).