5. Protočnost, resursi mašine i paralelizam

5.1 Pojam protočnosti

Protočnost (pipeline) se nalazi u mnogim oblastima ljudske delatnosti i predstavlja pokušaj paralelizacije obavljanja nekog posla. Taj koncept je prvu veliku promenu u društvu doneo kada se sa esnafske proizvodnje prešlo na manufakturnu proizvodnju, što je predstavljalo uvod u industrijsku revoluciju. U esnafu, osnovni princip je bio da svaki član esnafa pravi celokupan proizvod, dok su u manufakturnoj proizvodnji učesnici u proizvodnji specijalizovani za realizaciju neke faze proizvodnje. Manufakturna proizvodnja je specijalizacijom dovela do lakše obuke radnika i bržeg obavljanja tih specijalizovanih poslova, a između radnika su poluproizvodi razmenjivani po principu koji je kasnije doveo do realizacije proizvodnih traka u industriji.

U računarstvu, može se na primerima obične i protočne verzije MIPS procesora uočiti razlika u načinu rada u ta dva slučaja. Kada ne bi postojali protočni registri, tada bi se sa učitavanjem adrese instrukcije u programski brojač PC započelo adresiranje instrukcije. Posmatrajmo sva kašnjenja koja se tada javljaju u logici procesora do trenutka upisa rezultata instrukcije u procesorski registar. Prvo se javlja kašnjenje od trenutka generisanja adrese instrukcije, do trenutka dobijanja koda same instrukcije, što je posledica kašnjenja instrukcijske memorije (keša) od trenutka generisanja adrese do pojave podataka (instrukcije) na izlazu. To se naziva Fetch logikom. U daljem tekstu ovog poglavlja će se termini instrukcija i operacija koristiti kao sinonimi, jer u opisima rada RISC procesora dominira termin instrukcija, a u metodama paralelizacije koda termin operacija.

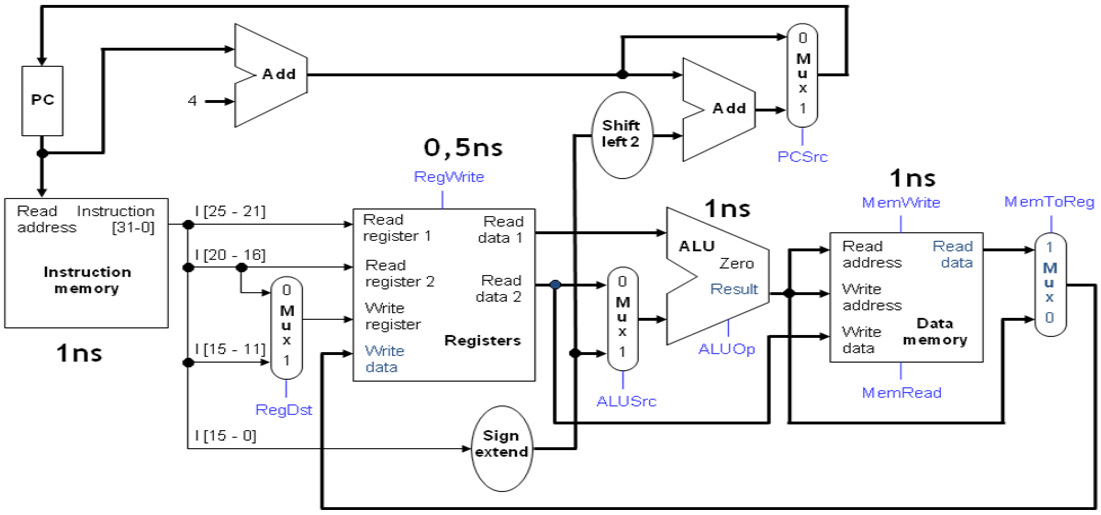
Kod MIPS procesora su adrese registara sastavni deo instrukcije, pa nema kašnjenja za generisanje adresa procesorskih registara koji se čitaju. Od trenutka kada su adresirani procesorski registri, do trenutka kada se podaci (argumenti instrukcije) pojave na izlazu, postoji kašnjenje registarske memorije pri čitanju. Ovo kašnjenje se kod MIPS procesora smatra Decode delom logike.

Nakon toga dodatno kašnjenje uvodi kombinaciona logika ALJ u kojoj se izvršavaju instrukcije. U MIPS procesoru se zatim javlja kašnjenje zbog memorijskog upisa ili čitanja (kada ALJ generiše rezultat koji se upisuje u memoriju ili računa adresu) u memoriju (keš) podataka. Kada ALJ generiše adresu, tada se odmah radi i čitanje memorije sa izračunate adrese - Load operacija. Kada je u pitanju upis (Store), tada se čeka na vreme upisa u Data memoriju, a adresa za pis se uzima iz registarske memorije procesora. Ovo kašnjenje se kod MIPS procesora smatra Data Memory delom logike. Na kraju se obavlja upis izlaza ALJ ili pročitanog podatka iz memorije u registarsku memoriju, ako nije bila u pitanju instrukcija upisa u memoriju. Taj deo logike se zove Write back deo logike.

Vremena smirivanja podataka za delove logike u ukupnom kombinacionom kašnjenju u nekoj tehnologiji proizvodnje integrisanih kola je približno sledeće:

a. Adresa u programskom brojaču do stabilnog izlaza instrukcijske memorije (podrazumeva se statički ram – instrukcijski keš) - 1ns (deo logike Fetch)

b. Kašnjenje procesorskih registra kod čitanja i pisanja – 0,5 ns (delovi logike Decode i Write back)
c. Kašnjenje ALJ - 1ns (deo logike Execute)
d. Kašnjenje statičke memorije za podatke (data keš) – 1ns (deo logike Data Memory)



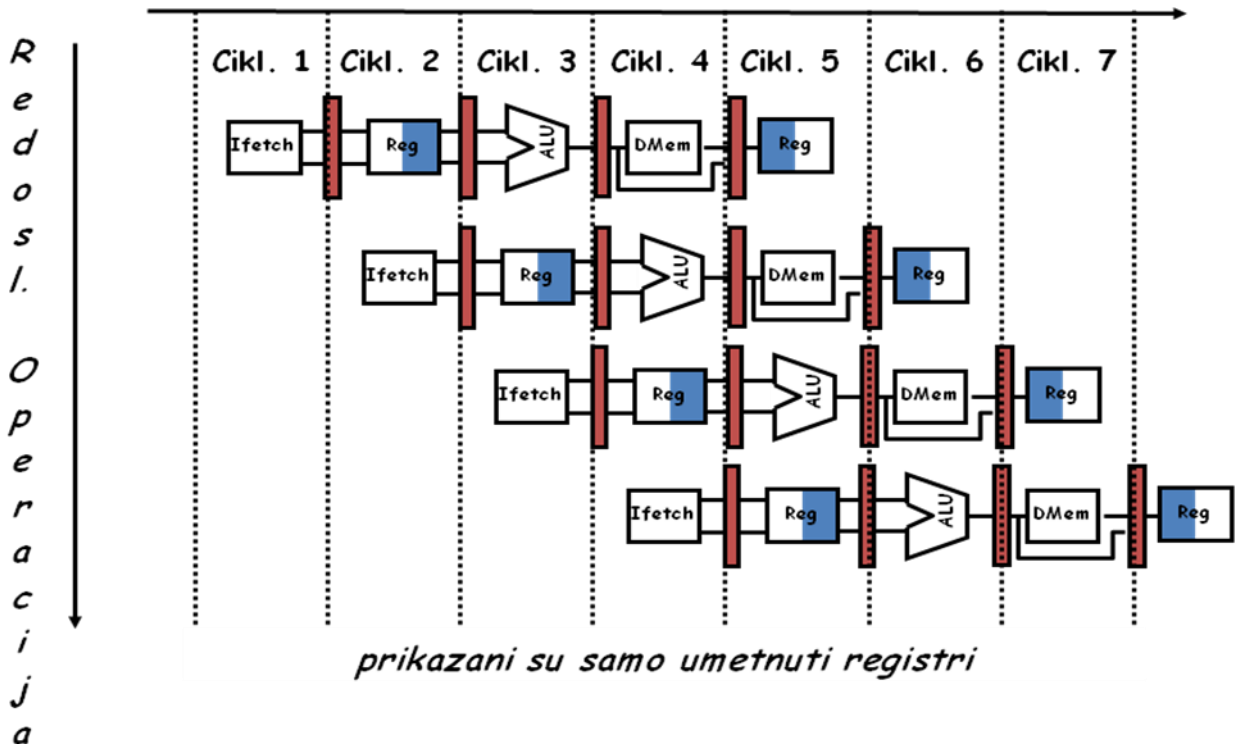
Sl. 5.1. MIPS bez protočnih stepeni i kašnjenja pojedinih delova logike.

Ukupno kašnjenje u izvršavanju instrukcije na ovakvom procesoru bi bilo oko 4 ns (1+0,5+1+1+0,5 ns), pa bi takt za izvršavanje cele instrukcije bio približno 250 MHz. Nova instrukcija se može započeti tek nakon kompletnog završetka prethodne instrukcije. Nameće se ideja da se instrukcija obavlja tako što za svaku fazu izvršavanja instrukcije umetnu registri i da se tako izdele kašnjenja kombinacione logike. Zbog manjeg kašnjenja tako izdeljene logike, može da se podigne učestanost takta. Ti delovi kombinacione logike, zajedno sa registrima na ulazima i izlazima čini protočni stepen. Protočni niz se sastoji iz niza protočnih stepeni kod kojih izlazni registri jednog protočnog stepena predstavljaju ulazni registar nekog sledećeg protočnog stepena.

Ako su protočni stepeni vezani u niz protočnih stepeni, može u njima da se u 5 ciklusa obavi celokupna instrukcija. U slučaju MIPS procesora se protočni registri umeću na izlazu instrukcijske memorije (instrukcijski registar), na izlazu registarske memorije (ujedno ulaz ALU), izlazu ALU (memorijski adresni registar, odnosno, memorijski registar za podatke za upis i registar na putu ka regisarskoj memoriji), na izlazu za podatke memorije za podatke, a destinacioni registri u registarskoj memoriji su već imali ulogu da zapamte rezultat instrukcije u registrima.

Ako pretpostavimo da protočni registri upisuju i pamte svoj ulaz na aktivnoj ivici takt impulsa ili latch tip registara, koristi se naziv sinhrona protočnost (synchronous pipeline). U daljem tekstu dela 5.1. će biti pretpostavljeni registri koji upisuju podatke sa svog ulaza na aktivnoj ivici takta. Kod sinhrone protočnosti, svi protočni registri dobijaju takt takav da se u približno istom trenutku pojavi aktivna ivicu takta. Svaki protočni registar zahteva da podaci na njegovom ulazu budu mirni neki period vremena pre pojave aktivne ivice takta (engl. register setup time). Pored toga, od trenutka pojave aktivne ivice takta do pojave mirnih podataka na izlazu iz registara se javlja dodatno

kašnjenje koje unosi registar (engl. clock to output time). Zbir ta dva vremena je kašnjenje koje unosi protočni registar.



Sl. 5.2. Sinhroni protočni niz MIPS procesora pri izvršavanju niza instrukcija

Svi protočni registri primaju isti takt, pa samim tim se učestanost takta mora prilagoditi najsporijoj kombinacionoj logici protočnog stepena u nizu. Kašnjenje najsporije logike za faze izvršavanja na MIPS procesoru sa Sl. 5.1. bilo 1ns i ako tome dodamo kašnjenje koje unose protočni registri (clock to output ulaznog registra i setup vreme izlaznog) koje je približno 0,1 ns, dobija se najkraća perioda takta od 1,1 ns odnosno najviša učestanost takta od približno 909MHz. Sada je ukupno trajanje izvršavanja instrukcije 5,5 ns (umesto 4 ns), ali se mogu paralelizovati faze izvršavanja za različite instrukcije kao na Sl. 5.2. Svakih 1,1 ns se može započeti izvršavanje nove instrukcije.

Ako posmatramo zauzeće resursa u kontekstu List Scheduling algoritma, instrukcija traje onoliko ciklusa koliko ima stepeni u protočnom nizu. Tipovi resursa mašine su protočni stepeni i javlja se samo po jedna instanca svakog tipa resursa. Međutim, u svakom ciklusu izvršavanja instrukcije, od strane instrukcije zauzima se samo po jedan protočni stepen (resurs) u nizu. Redosled zauzimanja protočnih stepeni (resursa) je isti za sve instrukcije u ovom pojednostavljenom posmatranju. Zato niz protočnih stepeni može da uvede značajan paralelizam – srazmeran broju protočnih stepeni u nizu, jer se svakog ciklusa može započeti nova instrukcija. Kako je u protočnom nizu u svakom protočnom stepenu različit tip resursa, u kontekstu List Scheduling-a imamo po jednu instancu resursa svakog tipa resursa u konfiguraciji mašine. U svakom svom ciklusu tada instrukcija zauzima instancu novog tipa resursa. Dakle instrukcija zauzima samo po jednu (i jedinu) instancu tipa resursa svakog ciklusa u toku svog izvršavanja u nizu protočnih stepeni. Zato je u List Scheduling raspoređivanju, u ovom slučaju, dovoljno ispitati da li je slobodan resurs u prvom ciklusu operacije, jer ako jeste, biće slobodni resursi i u ostalim ciklusima operacije.

Kada se započinje neko izvršavanje instrukcija, ako protočni niz ima n stepeni, tek na kraju n-tog ciklusa, kada svakog ciklusa započinjemo novu instrukciju, je popunjen niz protočnih stepeni i paralelno se izvršava n instrukcija. Kako u popunjenom protočnom nizu može da se startuje izvršavanje nove instrukcije svakog ciklusa, niz protočnih stepeni ima paralelizam srazmeran broju protočnih stepeni. Naravno, sve instrukcije nalaze se u različitim fazama izvršavanja. Posebno dobra osobina protočnog niza je da izlaz (izlazni registar) jednog protočnog stepena predstavlja ujedno ulaz (ulazni registar) drugog protočnog stepena. Na ovaj način uprošćeno je prosleđivanje rezultata i argunenata u odnosu na paralelizam u kome bi postojale nezavisne jedinice u paraleli bez protočnosti, a koje obavljaju istu funkciju kao celokupni niz protočnih stepeni.

Kontrolnim signalima definiše se funkcija ulaz – izlaz kombinacione logike svih protočnih stepeni u svakom ciklusu. Na taj način definiše se željena obrada u protočnom stepenu. Kod RISC procesora, najlakše se uočava funkcija kontrolnih signala koji utiču na funkcije ulaz – izlaz na primeru aritmetičko-logičke jedinice koja na svojim ulazima i izlazima ima protočne registre. Kontrolni signali tada određuju instrukciju koju obavlja aritmetičko logička jedinica. Oni se dovode do kombinacione logike Execute faze takođe preko protočnog registra.

Zbog jednostavnosti funkcije zauzeća resursa mašine od strane instrukcije kod protočnog niza, kada sve instrukcije traju isti broj ciklusa, krajnje je trivijalno uklapati instrukcije kod List Scheduling-a. Dakle, kod svih instrukcija koje se obrađuju u protočnom nizu, redosled i broj resursa određenog tipa se poklapa. Zato započinjanje nove instrukcije svakog ciklusa znači automatski izbegavanje konflikta u korišćenju resursa. Naravno, konfiguracija mašine se čak ni u slučaju RISC procesora ne može posmatrati kao jedan protočni niz, mada se prilikom uprošćenog posmatranja to često radi. U realnosti, ne traju ni sve instrukcije isti broj ciklusa. Osim toga, prosleđivanje rezultata od instrukcije koja generiše rezultat do instrukcije čiji je to argument konplikuje raspoređivanje, jer se moraju uzeti u obzir prave zavisnosti po podacima. Zato ovo dosadašnje razmatranje treba posmatrati samo kao analizu osnovnog koncepta protočnosti.

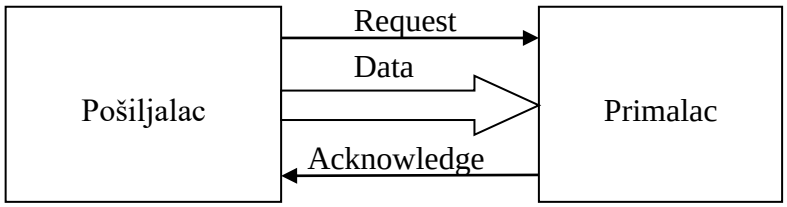
Posebno je interesantan problem uslovnog grananja u slučaju protočnog niza. Ako instrukcija – operacija generiše signal na osnovu koga se donosi odluka o skoku, tada bi se moralo sačekati sa prvim fazama izvršavanja instrukcija iza instrukcije uslovnog skoka, dok se ne bude poznat ishod skoka. Načini za izbegavanje takvog neželjenog pražnjenja protočnog niza mogu da budu selidbe operacija preko grananja na dole preko grananja u vreme prevođenja ili dinamička predikcija grananja. Kada se ne bi radile takve transformacije, bilo bi neophodno pražniti niza protočnih stepeni i samim tim imati neiskorišćene resurse i smanjeni paralelizam. Selidbe preko grananja na dole popunjavaju u tom slučaju protočni niz sigurno korisnim operacijama. Kod dinamičke predikcije, bolje srednje iskorišćenje postiže se ako pretpostavimo ishod skoka i odmah popunimo niz operacijama (instrukcijama) iz pretpostavljene grane. To je spekulativno izvršavanje ciklusa operacija po kontroli - iz pretpostavljene grane. Ako je u jednostavnom protočnom nizu došlo do greške u predikciji, ne moraju se poništavati rezultati predviđenih operacija, jer nisu ni izašli iz protočnog niza. Tada se samo poništavaju sve faze izvršavanja svih operacija koje su spekulativno na bazi predikcije ušle u protočni niz, pa se praktično prazni protočni niz i gubi na paralelizmu. Međutim, ako se pogodi ishod skoka, tada nema gubitka u paralelizmu i resursi protočne mašine maksimalno se koriste. Dakle, i u slučaju jednostavnog protočnog niza, neophodno je

obezbediti dobru predikciju grananja, kako bi se na nivou faza operacije ostvarilo spekulativno izvršavanje po kontroli kod koga je velika verovatnoća da će se spekulativno urađeni delovi operacija iskoristiti, a ne odbaciti.

5.2. Asinhrona protočnost

Osnovna ideja kod asinhrone protočnosti je da ne postoji jedinstven takt za protočne registre. Umesto toga, postoji protokol rukovanja (handshake) kojim se obezbeđuje razmena podataka između kombinacione logike protočnog stepena i ulaznih i izlaznih registara. Protokol rukovanja je dvofazni i biće prikazan prvo za generalni slučaj pošiljaoca i primaoca.

U dvofaznom protokolu rukovanja, pošiljalac osigura da su svi podaci skupa podataka koje treba da preda primaocu (data) validni (i stabilni) pre nego što pošalje zahtev (Request) prema primaocu. Primalac prihvata podatak i šalje signal potvrde (Acknowledge) kojim javlja da je preuzeo validne podatke od pošiljaoca. Taj signal ujedno znači da je data dozvola pošiljaocu da ukloni podatak i da može da postavi nove vrednosti na linije za podatke.



Sl. 5.3. Dvofazni protokol rukovanja

Neka su pošiljalac i primalac protočni stepeni kod kojih je izlazni registar izostavljen, jer je izlazni registar pošiljaoca ujedno ulazni registar narednog protočnog stepena. Tada kombinaciona logika pošiljaoca, u trenutku kada su podaci (Data) na izlazu kombinacione logike stabilni, šalje signal request. Asinhroni registar primaoca odmah prima podatak, ako je od svog (narednog) protočnog stepena dobio signal potvrde da može da promeni vrednost registra. U suprotnom, čeka potvrdu i po dobijanju prima podatak. Čim registar primaoca primi podatak, odmah šalje potvrdu (Acknowledge) da ulazni registar pošiljaoca može da primi novi podatak.

Zbog navedenog mehanizma potvrda, najsporiji protočni stepen određuje dinamiku kojom će da funkcioniše ceo protočni niz. On će usporavati ranije protočne stepene u nizu jer će sledeći protočni stepen najsporije dobijati Request signal i samim tim najsporije prihvatati rezultat. Pošto, tek po prijemu rezultata taj naredni stepen može da generiše Acknowledge signal, ulazni registar najsporijeg stepena tek tada dobija signal da može da upiše novu vrednost. Po upisu nove vrednosti on šalje svojem prethodniku u nizu Acknowledge signal i time dopušta da se u njega upisuje nova vrednost. Ovo se ponavlja do početka niza. Protočni stepeni iza najsporijeg ne mogu da dobijaju ulazne podatke brže nego što ih generiše najsporiji stepen.

Asinhrona protočnost je retko primenjivana kod procesora, a glavni razlog za češću primenu sinhrone protočnosti je što u sinhronoj protočnosti postoje stabilna stanja pre pojave aktivne ivice takta. To omogućava lakše testiranje dizajna procesora i realizaciju

izvršavanja programa u stabilnim koracima od po jednog ciklusa takta – single step. Kod sinhrone protočnosti postoje dve slabosti kod kojih asinhrona protočnost ima prednosti. Zaustavljanje celog protočnog niza kod asinhrone protočnosti se trivijalno izvodi tako što se ne pošalje signal potvrde prethodnom **susednom** protočnom stepenu i samim tim se prirodno zaustave svi protočni stepeni pre tog protočnog stepena. U slučaju sinhrone protočnosti, mora se do centralnog generatora takta poslati signal da se zaustavi ceo protočni niz. Ovo kašnjenje može da bude veliko i može da izazove usporavanje takta procesora zbog kašnjenja izazvanog propagacijom signala za zaustavljanje takta generatoru takta koji tipično nije susedan. Druga prednost je što se ne mora distribuirati takt, jer se usled različitih kašnjenja signala takta do protočnih registara narušava ideja da svi protočni registri istog trenutka dobiju aktivnu ivicu takta.

5.3. Protočnost u talasima

Protočnost u talasima je tehnologija u kojoj se protočnost realizuje u kombinacionoj logici bez registara! Pretpostavimo da postoji kombinaciona logika sa velikim kašnjenjem između dva registra. Ako je kašnjenje kroz kombinacionu logiku jednako na svim putanjama signala, stabilne validne vrednosti mogu da putuju kroz kombinacionu logiku u talasima. Tada ulazni registar može u intervalima vremena kraćim od ukupnog kašnjenja kombinacione logike da menja svoje vrednosti, a da stabilne vrednosti putujući kao talasi kroz kombinacionu logiku ĆčuvajuĆ vrednosti koje odgovaraju prethodnoj vrednosti ulaznog registra. Putujući kroz logiku, talas menja vrednosti u skladu sa logičkim funkcijama kombinacione logike. Kada talas sa stabilnim vrednostima stigne do izlaznog registra, postoji interval u kome on mora da prihvati stabilne vrednosti na svom ulazu.

Takt ulaznog i izlaznog registra ima periodu kraću od vremena kašnjenja kombinacione logike zahvaljujući pamćenju intervala stabilnih vrednosti kroz talase u svakom delu kombinacione logike. Da bi se postiglo podjednako kašnjenje na svim putevima kroz kombinacionu logiku, potreban je specifičan projekat integrisanih kola koji se realizuje pomoću posebnih alata za računarsko projektovanje kola. Današnja tehnologija obezbeđuje da protočnost u talasima funkcioniše tako da je takt na ulaznom i izlaznom registru 4 puta brži nego što je takt klasične protočnosti. Limiti su vezani za nemogućnost idealnog izjednačavanja putanja kašnjenja i potrebna širina stabilnog talasa da bi registar mogao da prihvati vrednosti talasa na izlazu.

Razlog za uvođenje protočnosti u talasima je što ne postoje kašnjenja registara zamenjenih talasima, pa se može podići viša učestanost takta nego u slučaju uvođenja protočnih registara. Osnovna mana protočnosti u talasima je što se ne može zaustaviti protočni niz i što samim tim ceo niz mora da radi i kada nema validnih podataka za obradu.

5.4. Završne napomene za sinhronu protočnost

Kod MIPS procesora je bilo uočljivo da se instrukcijska memorija (instrukcijski keš) i memorija podataka (data keš) prilikom čitanja javljaju sa svojim kombinacionim kašnjenjem u protočnim stepenima kao deo ekvivalentne ĆkombinacioneĆ logike protočnih stepeni. U slučaju memorije podataka (Data memory) upis u memoriju podataka nije izazivao nikakav dalji efekat u ciklusu kada se radio upis. Međutim, kada bi se u nekom od kasnijih ciklusa desilo čitanje iste lokacije te memorije, bila bi

promenjena funkcija ulaz-izlaz ekvivalentne ĆkombinacioneĆ logike protočnog stepena. Zato se izvršavanje programa na procesorima iz ugla protočnosti može svesti na promene funkcije ulaz-izlaz kombinacione logike protočnih stepeni.

Iako se protočnost najčešće posmatra kao linearni niz protočnih stepeni, u procesorima često postoje ciklusi protočnih stepeni. Veze između protočnih stepeni mogu da imaju bilo koju strukturu usmerenog grafa, jer registri koji čine izlaz nekog protočnog stepena mogu da budu delovi ulaznih registara nekoliko različitih protočnih stepeni. Osim toga, broj protočnih stepeni u linearnom nizu protočnih stepeni može da se menja u zavisnosti od tipa operacije koja se izvršava. Kako se kod superskalarnih procesora još kompleksnije iskazuju takvi problemi i rešenje mora da bude mnogo generalnije, razmatranje tih detalja je ostavljeno za poglavlje o superskalarnim procesorima.