ARHITECTURA CALCULATOARELOR 2003/2004
CURSUL 9

3.3 Intrare/Ieşire

După cum am spus la începutul acestui capitol, un sistem de calcul are trei componente majore:

1) UCP – Central Procesing Unit – unitatea centrală de prelucrare a datelor,
2) Memoriile - primară şi secundară
3) Echipamentele de Intrare / Ieşire cum sunt, de exemplu, imprimantele, scannere (dispozitive ce transforma o imagine de pe hârtie în valori numerice, modemuri (dispozitive ce permit transmiterea la distanțe mari a datelor).

3.3.1 Magistrale

Fizic, majoritatea calculatoarelor personale şi stațiile de lucru au o structură cunoscută. Aranjamentul obişnuit este o cutie de metal cu o placă mare de circuit imprimat în partea de jos, numită placă de bază, matherboard. Placa de baza conține cipul UCP, câteva sloturi pentru soclurile de memorie (DIMM-uri de exemplu) şi socluri pentru alte cipuri. Deasemenea există o magistrală, trasee efectiv imprimate pe cablaj, iar în lungul ei sloturi în care pot fi introduşi conectorii plăcilor I/O suplimentare. Unele plăci de bază au două tipuri de magistrale, inclusiv una de viteză mai mică pentru compatibilitatea cu plăcile I/O mai vechi şi mai lente. Structura logică a unui calculator simplu este prezentată în figura 3.4:

Această structură are o singură magistrală folosită să lege CPU, memoria şi dispozitivele intrare/ieşire; majoritatea sistemelor au două sau mai multe magistrale. Fiecare dispozitiv I/O este alcătuit din două părți: una conținând majoritatea electronicii, numit controler, şi alta conținând dispozitivul de I/O însuşi, un exemplu de dispozitiv este un disc. Controlerul este în mod obişnuit conținut de o placă cuplată într-un slot liber, excepție făcând controlerele care nu sunt opționale, ca urmare sunt plasate pe placa de bază.

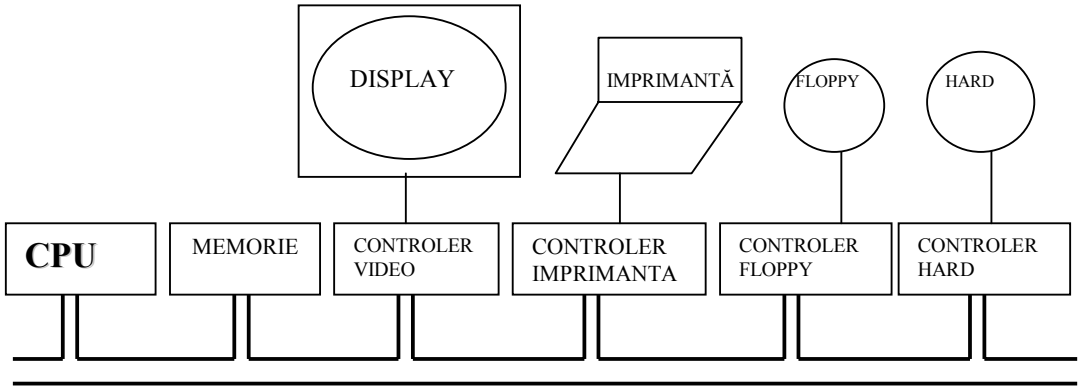


Figura 3.4 Structura logică a unui sistem de calcul

Din punct de vedere al sistemului gazdă, controlerul este "văzut" cu ajutorul unui program specializat numit driver. Driver-ul controlează dialogul dintre calculator, care lucrează cu periferice virtuale şi care foloseşte ordine generice, şi periferic, care primeşte ordine particulare specifice.

Chiar dacă monitorul nu este o opţiune, controlerul video este uneori plasat pe o placă separată de placa de bază, pentru a permite utilizatorului să aleagă între plăci cu sau fără acceleratoare, memorie în plus şi aşa mai departe. Controlerul este conectat la dispozitivul său cu un cablu ataşat la un conector pe spatele cutiei.

Funcţia unui controler este de a verifica dispozitivul său de I/O şi a menţine accesul la magistrală pentru el, când un program vrea date pe disc, de exemplu, trimite o comandă controlerului discului, care apoi emite comenzi de căutare a poziţiei (seek) şi alte comenzi specifice către dispozitiv. Când pista şi sectorul corecte au fost găsite, dispozitivul începe să scoată datele ca un flux în serie de biţi către controler. Este treaba controlerului de a sparge lanţul de biţi în unităţi şi să scrie fiecare unitate în memorie aşa cum a fost asamblată. O unitate tipică este formată din unul sau mai multe cuvinte. Un controler care citeşte sau scrie informaţie din şi în memorie fără implicarea CPU se spune că realizează Direct Memory Acces, mai bine cunoscut prin acronimul său DMA. Când transferul este complet, controlerul produce o întrerupere, forţând CPU să suspende funcţionarea programului curent şi să lanseze o procedură specială numită, interrupt haudler (rutina de tratare a intreruperii), pentru a verifica dacă sunt erori, pentru a proceda în consecinţă, şi pentru a informa sistemul de operare că operaţia de I/O este acum terminată. CPU cuntinuă cu programul care a fost oprit temporar, când întreruperea a avut loc.

Magistrala nu este folosită numai de către controlerele I/O, dar şi de CPU pentru apelarea instrucţiunilor şi a datelor. Ce se întâmplă dacă CPU şi un controler I/O, vor să folosească în acelaşi timp magistrala? Răspunsul este că un cip (circuit) numit "bus arbiter" (arbitru de magistrală), decide cine este următorul utilizator al magistralei. În general dispozitivele I/O sunt preferate înaintea CPU, pentru că discurile şi alte dispozitive în mişcare nu pot fi oprite, ar a le forţa să aştepte ar duce la pierderea de date. Când nici o operaţie I/O nu este în execuţie, CPU poate avea la dispoziţie toate ciclurile magistralei pentru a accesa memoria. Totuşi când lucrează şi un dispozitiv I/O, acesta va solicita şi i se va acorda magistrala când este nevoie. Acest proces se numeşte "cycle stealing" (furarea ciclului) şi incetineşte computerul.

Acest proiect a lucrat bine pentru primele computere personale, deoarece toate componentele erau în echilibru. Totuşi, pentru că CPU-urile, memoriile şi dispozitivele I/O au evuluat mai repede, a apărut o problemă: magistrala nu mai poate face faţă sarcinii prevăzute. Pe un sistem închis, cum ar fi o staţie de lucru tehnologică, soluţia a fost să se proiecteze o magistrală nouă şi mai rapidă pentru următorul model. Pentru că în acest caz nimeni nu mută un dispozitiv I/O de pe un model vechi pe unul nou.

Totuşi, în lumea PC, oamenii işi îmbunătăţesc CPU, dar doresc să-şi păstreze imprimanta scanerul şi modemul pentru noul sistem. De asemenea, în jurul furnizării unei game largi de dispozitive I/O pentru magistrala IBM PC s-a dezvoltat o industrie uriaşă, care are un interes prea mic ca să-şi arunce investiţiile şi să o ia de la început. IBM a invăţat acest drum greu când a scos ca succesor pentru IBM PC gama PS/2. PS/2 avea o magistrală nouă şi mai rapidă, dar cea mai mare parte a celor care faceau clone (copii) au continuat să folosească vechea

magistrală PC, denumită ISA (Industry Standard Arhitecture). Mulți dintre cei care produceau dispozitive I/O şi discuri au continuat să facă şi controlere pentru acestea, astfel încât IBM s-a găsit într-o situație dificilă de a fi singurul fabricant de PC, care numai era compatibil cu IBM. Ca alternativă a fost forțat să sprijine magistrala ISA. ISA reprezintă Instruction Set Arhitecture în contextul nivelului maşină şi Industry Standard Arhitecture în contextul magistralelor.

Cu toate acestea, în ciuda presiunii pieței de a nu schimba nimic, vechea magistrală era cu adevarat prea lentă, deci trebuia făcut ceva. Această siuație a făcut ca mai multe companii să creeze separat mai multe magistrale, dintre care una era vechea magistrală ISA sau succesoarea sa compatibilă, magistrala EISA (Extended ISA –ISA extinsă). Cea mai populară dintre acestea a fost PCI (Periphal Component Interconnect). A fost concepută de Intel, dar acesta a hotărât să treacă toate aceste patente în domeniul public pentru a încuraja întreaga industrie (inclusiv concurenții) să o adopte şi să lucreze pentru ea.

Magistrala PCI poate să fie folosită în mai multe configurații, dar una tipica este prezentată în figura 3.5. Aici CPU dialoghează cu un controler al memoriei pe o conexiune de mare viteză dedicată. Controlerul dialoghează cu memoria şi cu magistrala PCI direct, astfel că traficul memoriei CPU nu trece pe magistrală PCI. Totuşi, perifericele cu lățime mare de bandă (adică rată ridicată de transfer, viteză mare), cum ar fi discurule SCSI, se pot conecta direct la magistrala PCI. In plus magistrala PCI are o conexiune, o punte (bridge) la magistrala ISA, astfel încât controlerele ISA şi dispozitivele lor să poată fi folosite. O maşină de această concepție conține 3 sau 4 sloturi PCI şi măcar un slot ISA disponibile, pentru a permite clienților să se cupleze atât vechile plăci ISA (mai lente) cât şi plăcile PCI (mai rapide). Soluția este însă pe cale de dispariție. Soluția ISA a fost susținută şi de producătorii de plăci dedicate (de exemplu achiziție şi prelucrare de semnale) care nu aveau atâta vânzare încât să poată schimba proiectele la fel de repede cum reuşeau producătorii de calculatoare personale să o facă.

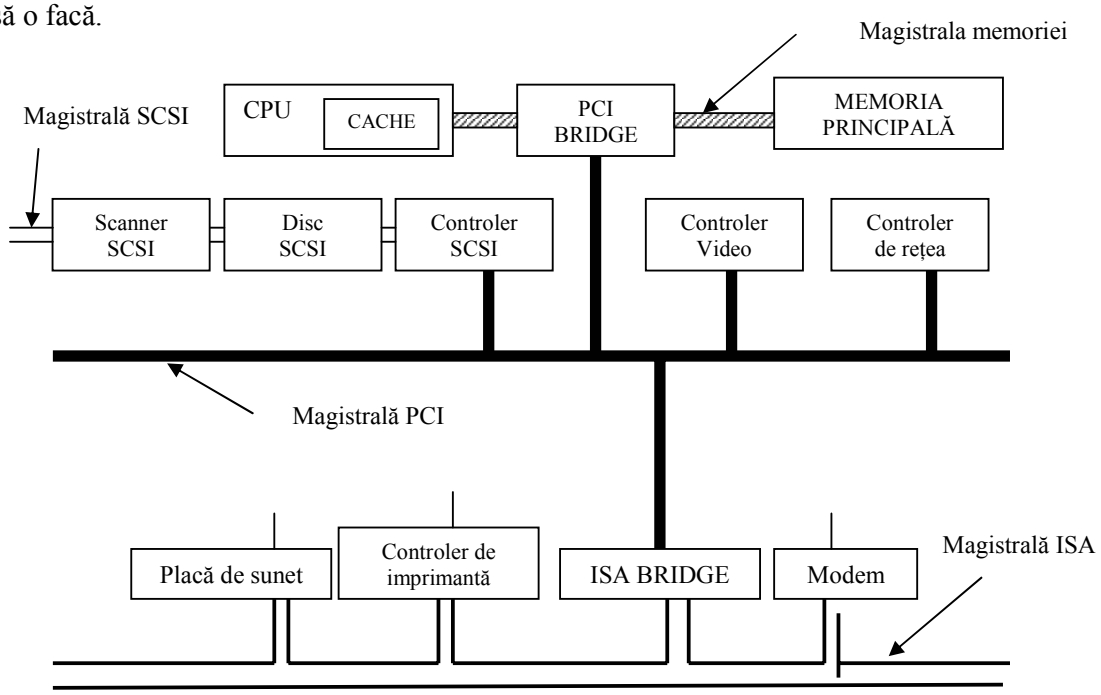


Figura 3.5 Sistem cu două magistrale

CAPITOLUL 4: Nivelul logic numeric

La baza structurii ierarhizate a unui calculator numeric, aşa cum este prezentat în figura 4.1, găsim nivelul logic numeric (digital), adevăratul hardware al calculatorului. În acest capitol, vom examina câteva aspecte ale nivelului logic digital, cum ar fi construirea blocurilor pentru studiul nivelelor înalte ale ierarhiei amintite. Acest subiect este la limita ştiinţei calculatoarelor şi a ingineriei electrice, dar materialul este independent, deci nu sunt necesare hardware anterior sau experienţe inginereşti pentru a-l folosi.

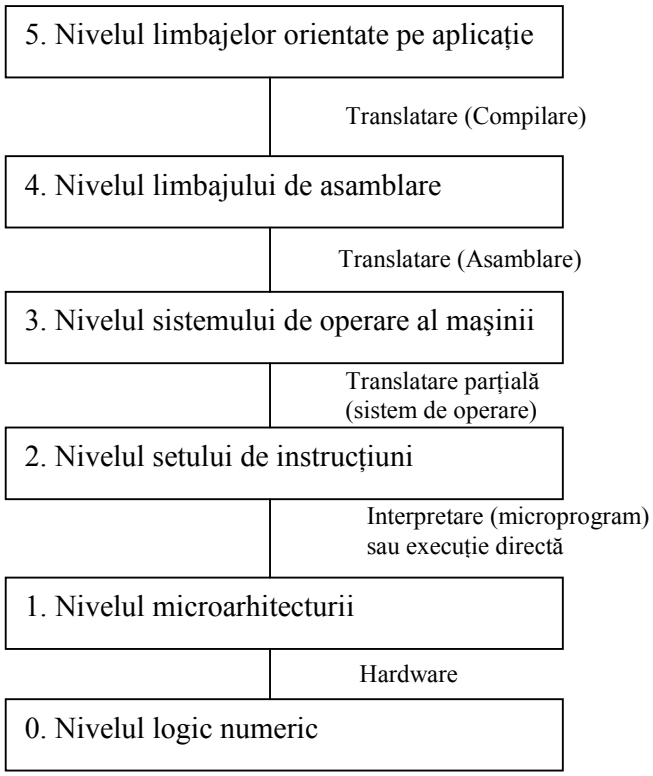


Figura 4.1 Structura ierarhizată a unui sistem de calcul

Elementele de bază din care sunt construite toate calculatoarele numerice sunt uluitor de simple. Primul pas este studiul acestor elemente de bază şi al algebrei speciale cu două valori (algebra booleana) folosită pentru a le analiza. După aceea se studiază circuitele fundamentale care pot fi construite folosind porţi în combinaţii simple, incluzând circuite pentru operaţii aritmetice. Următoarea temă este aceea de a afla cum porţile pot fi combinate pentru a stoca informaţii, ceea ce înseamnă, cum este este organizată memoria. După aceasta, se revine la subiectul CPU şi în special cum CPU pe un singur cip dialoghează cu memoria şi cu dispozitivele periferice.

Observatie:
Porţile logice şi algebra booleana au fost introduse la începutul cursului şi studiate la alte cursuri. Circuitele logice numerice, adică circuitele integrate combinaţionale (multiplexoarele, decodificatoarele, comparatoare, matricele logice programabile PLA etc.) sunt studiate la alte cursuri.

4.1. Circuite aritmetice

Vom începe cu un simplu registru de deplasare, apoi să vedem cum sunt realizate sumatoarele, iar în final să examinăm ALU, care joacă un rol central în orice calculator.

4.1.1. Registre de deplasare

Primul circuit aritmetic prezentat este un registru de deplasare cu 8 intrari şi 8 iesiri, ca în figura 4.2, 8 biţi de intrare sunt prezenţi pe liniile D_0 ,....., D_7. Ieşirea, care este intrarea deplasată cu un bit, este disponibilă pe liniile S_0 ,.....,S_7. Linia de control C determină direcţia de deplasare, 0 pentru stâga, 1 pentru dreapta. Pentru a vedea cum funcţionează circuitul trebuie să remarcaţi perechile de porţi AND pentru toţi biţii cu excepţia porţilor de la capete. Când C=1, membrul drept al fiecărei perechi este activat, transferând bitul de intrare corespunzător la ieşire. Deoarece poarta AND din dreapta este conectată la intrarea porţii OR din dreapta sa, se va realiza o deplasare spre dreapta a bitului. Când C=0 membrul stang al porţii AND se va activa realizând o deplasare spre stânga.

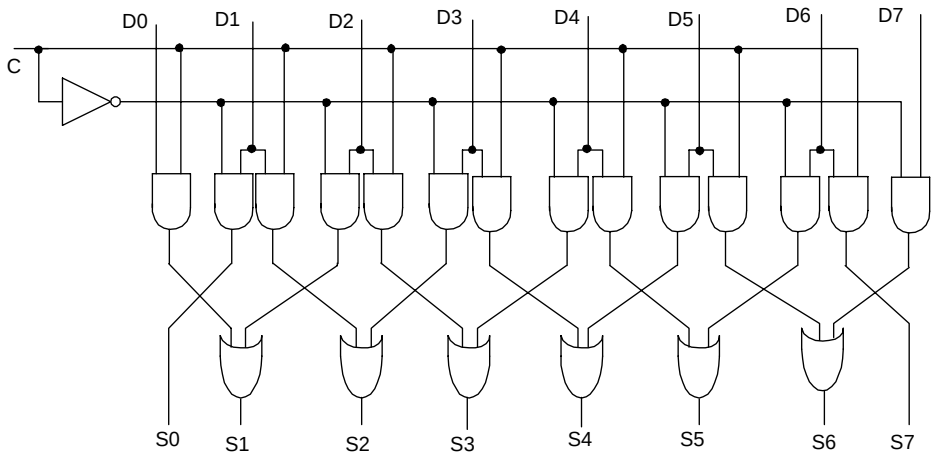


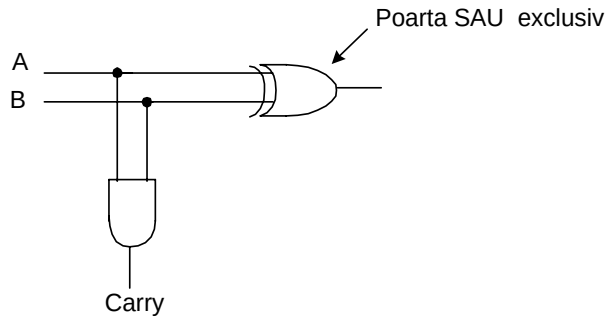
Figura 4.1 Un registru de deplasare stânga/dreapta cu 1 bit

4.1.2. Sumatoare

Un calculator care nu poate aduna întregi este aproape de neconceput. În consecinţă un circuit hard care realizează adunări este o parte esenţială a oricărui CPU. Tabela de adevăr pentru adunarea întregilor pe 1 bit este prezentată în figura 4.2a. Sunt prezente două ieşri: suma intrărilor A şi B, şi transferul (carry) către următoarea poziţie (spre stânga). Un circuit pentru calcularea atât a bitului sumă, cât şi a bitului de transport este ilustrat în figura 4.2b. Acest circuit simplu este cunoscut în general sub numele de “half adder” (sumator pe jumătate).

Deşi un sumator pe jumătate este adecvat pentru adunarea biţilor de pondere mică a doua cuvinte de intrare multibit, el nu va face acest lucru pentru un bit poziţionat la mijlocul cuvântului deoarece el nu va folosi transportul de la rangul din dreapta. Ca soluţie este nevoie de un sumator complet ca în figura 4.3. Din inspectarea circuitului, ar trebui sa reiesă clar că un sumator complet este construit din 2 sumatoare pe jumătate. Linia de ieşire “sum” este 1 dacă un număr impar din biţii A, B şi “carry in” sunt 1. Bitul “carry out” este 1 dacă atât A cât

şi B sunt 1, sau numai unul din ei este 1 şi bitul "carry in" este deasemenea 1. Împreună cele 2 sumatoare pe jumătate generează atât bitul suma cât şi bitul de transport.



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figura 4.2 (a) Tabela de adevăr pentru adunarea pe un bit (b) Un circuit pentru un sumator pe jumătate.

A	B	"Carry in"	Suma	"Carry out"
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

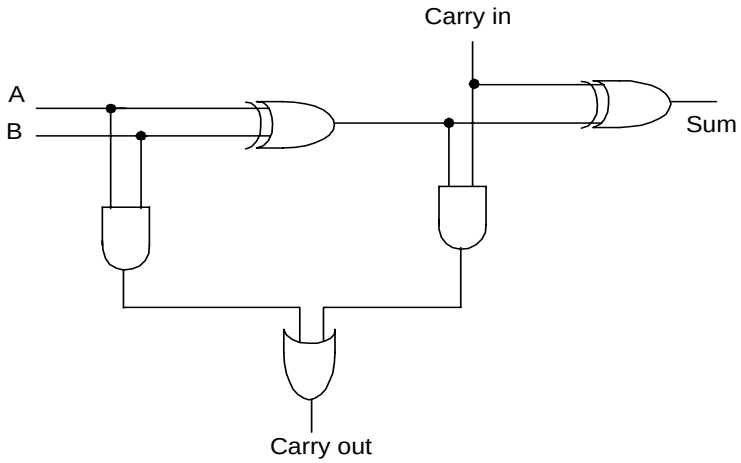


Figura 4.3 a) Tabela de adevăr a unui sumator întreg b) Circuitul pentru sumatorul întreg

Pentru a construi un sumator, să zicem pentru două cuvinte pe 16 biți, trebuie să construim 16 replici ale circuitului din figura 4.3. Transportul unui bit "carry out" este folosit ca linie "carry in" pentru vecinul său din stânga. Transportul la cel mai din dreapta bit este legat la 0. Acest

tip de sumator este denumit “ripple carry adder” (sumator cu propagare) deoarece în cel mai rău caz adăugând 1 la 11...111 (binar), însumarea nu se completează până când transportul nu se propagă pe tot traseul de la cel mai din dreapta bit până la cel mai din stânga bit.

Există de asemenea, şi sunt de obicei preferate, sumatoarele care nu prezintă această întârziere, şi care sunt mai rapide. Ca un exemplu simplu de sumator rapid consideraţi spargerea unui sumator pe 32 biţi într-unul pe 16 biţi pentru prima jumătate şi unul de 16 biţi pentru jumătatea superioară. Când începe însumarea, sumatorul superior nu poate lucra deoarece nu va cunoaşte transportul (carry in) timp de 16 însumări.

Oricum, luaţi în calcul această modificare. În loc să aveţi o singură jumătate superioară, se poate realiza sumatorul cu 2 jumătăţi superioare în paralel duplicând hardware-ul pentru jumătatea superioară. Circuitul constă în 3 sumatoare pe 16 biţi: unul pentru jumătatea inferioară şi două pentru cea superioară, U_0 şi U_1, care rulează în paralel. Un zero este intodus în U_0 ca transport (carry), iar în U_1 este intodus 1 ca transport. Acum, ambele vor porni în acelaşi timp cu partea inferioară, dar numai unul va fi corect. După 16 cicluri de însumare se va cunoaşte care este transportul în partea superioară, deci acum se poate selecta jumătatea superioară corectă din cele două răspunsuri disponibile. Acest truc reduce timpul de însumare cu un factor de 2. Un astfel de sumator este denumit sumator cu selecţia transportului (carry select adder). Acest truc poate fi repetat apoi pentru a construi fiecare sumator pe 16 biţi din sumatoare pe 8 biţi şi aşa mai departe.

4.1.3. Unităţi logice aritmetice

Majoritatea computerelor conţin un singur circuit pentru realizarea operaţiilor AND, OR, NOT şi SUM (însumare) a două cuvinte maşină. De obicei un astfel de circuit pentru cuvinte pe n biţi este construit din n circuite identice pentru fiecare bit individual. Figura 4.4 este un exemplu simplu de astfel de circuit denumit ALU. Acesta poate calcula oricare din următoarele 4 funcţii: A AND B, A OR B, A+B, $\overline{B}$, în funcţie de combinaţia liniilor de intrare F_0, F_1: 00, 01, 10 sau 11. De remarcat că A+B înseamnă suma aritmetică a lui A şi B, nu OR boolean.

Partea din stânga jos a ALU conţine un decodificator pe 2 biţi pentru a genera semnalele de control F_0 şi F_1. În funcţie de valorile lui F_0 şi F_1 numai unul din cele 4 linii este selectată. Setarea acestei linii permite ca ieşirea din funcţia selectată să ajungă la poarta OR pentru ieşire. Partea din stânga sus conţine logica de calcul pentru A AND B, A OR B şi B negat, dar cel mult unul din aceste rezultate este trnsferat la poarta OR finală în funcţie de liniile de validare ce provin din decodor. Deoarece exact una din linii va fi 1, numai una din cele 4 porţi AND ce conduc la poarta OR va fi validată, celelalte trei vor fi 0, independent de A şi B.

În plus pentru a putea utiliza A şi B ca intrări pentru operaţii logice şi aritmetice, este de asemenea posibil să se forţeze la zero oricare din ele negând ENA sau ENB. Este de asemenea posibil să se obţină $\overline{A}$ setând INVA. În condiţii normale, ENA şi ENB sunt ambele 1 şi INVA va fi 0. În acest caz A şi B sunt introduse în ALU fără modificări.

Colţul din dreapta jos al ALU conţine un sumator complet pentru calcularea sumei dintre A şi B inclusiv manipularea biţilor de transport, deoarece este posibil ca astfel de circuite să fie

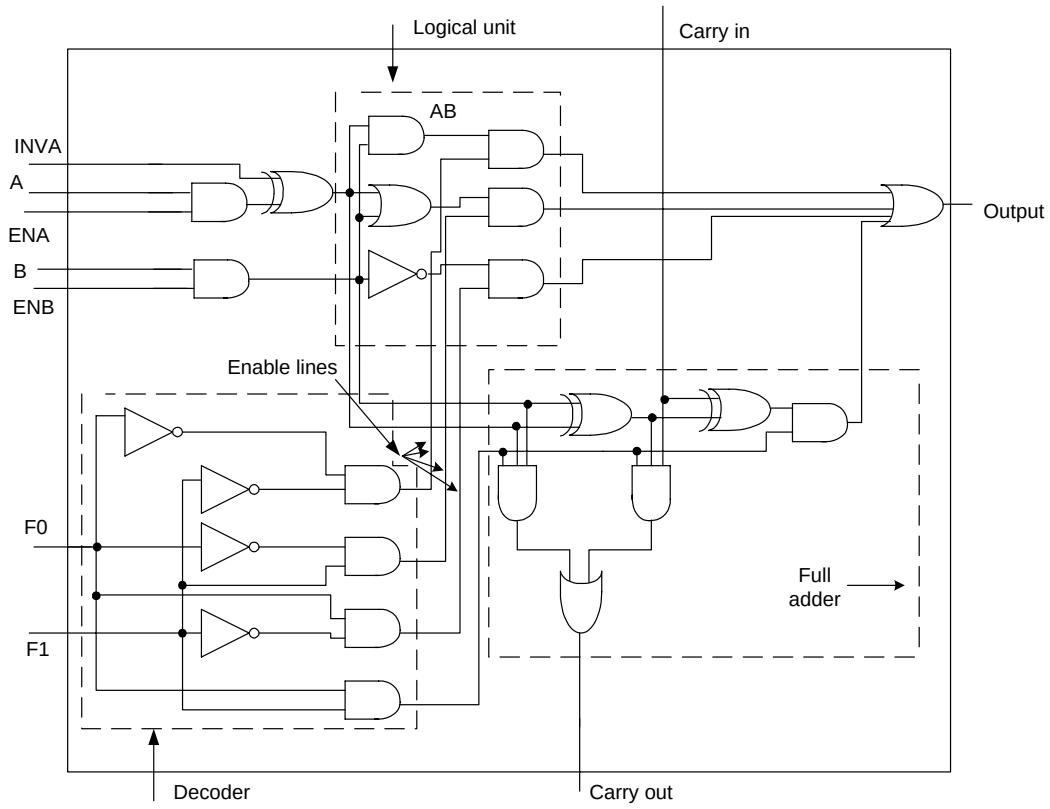


Figura 4.4 ALU pe 1 bit

conectate împreună pentru a realiza operații la nivelul de cuvânt. Circuite ca în figura 4.4 sunt disponibile şi sunt cunoscute sub numele de "bit slices". Acestea permit proiectantului de calculatoare să construiască un ALU de orice mărime doreşte.

În figura 4.5 se prezintă o ALU pe 8 biți construită din 8 ALU pe un bit. Semnalul INC este util numai pentru operații de adunare. Când este prezent, acesta incrementează (adică adună 1) rezultatul, făcând posibilă calcularea sumelor de genul A+1 şi A+B+1. Semnalele de validare (Enable) şi Inversare (Invert) nu sunt marcate pentru a simplifica figura.

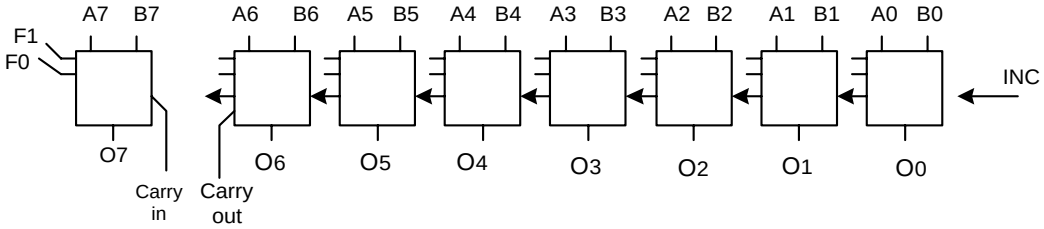


Figura 4.5 8 ALU pe 1 bit conectate pentru a realiza 1 ALU pe 8