Tietokoneen komponentit

Turun yliopisto
University of Turku

Tänään

- Loogiset piirit
- Tietokoneen komponentit
  - Yhteen- ja vähennyslasku
  - Väylä
  - Rekisterit
  - Kello

Loogiset piirit

Loogiset piirit

- Transistorien ja diodien avulla muodostetaan kytkentöjä, ns. ***loogisia piirejä*** (*logical circuits*)
  - Piirit kuvaavat päätöksentekoa eli logiikkaa, kun korkea ja matala jännitetaso tulkitaan totuusarvoina tosi ja epätosi
- Loogiset piirit muodostuvat ***porteiksi*** tai ***veräjiksi*** (gates) kutsutuista komponenteista
- Portti toteuttaa jonkin totuusarvoisen funktion yhden tai useamman syöttö- ja tulossignaalin välillä

Porttien toteutuksesta

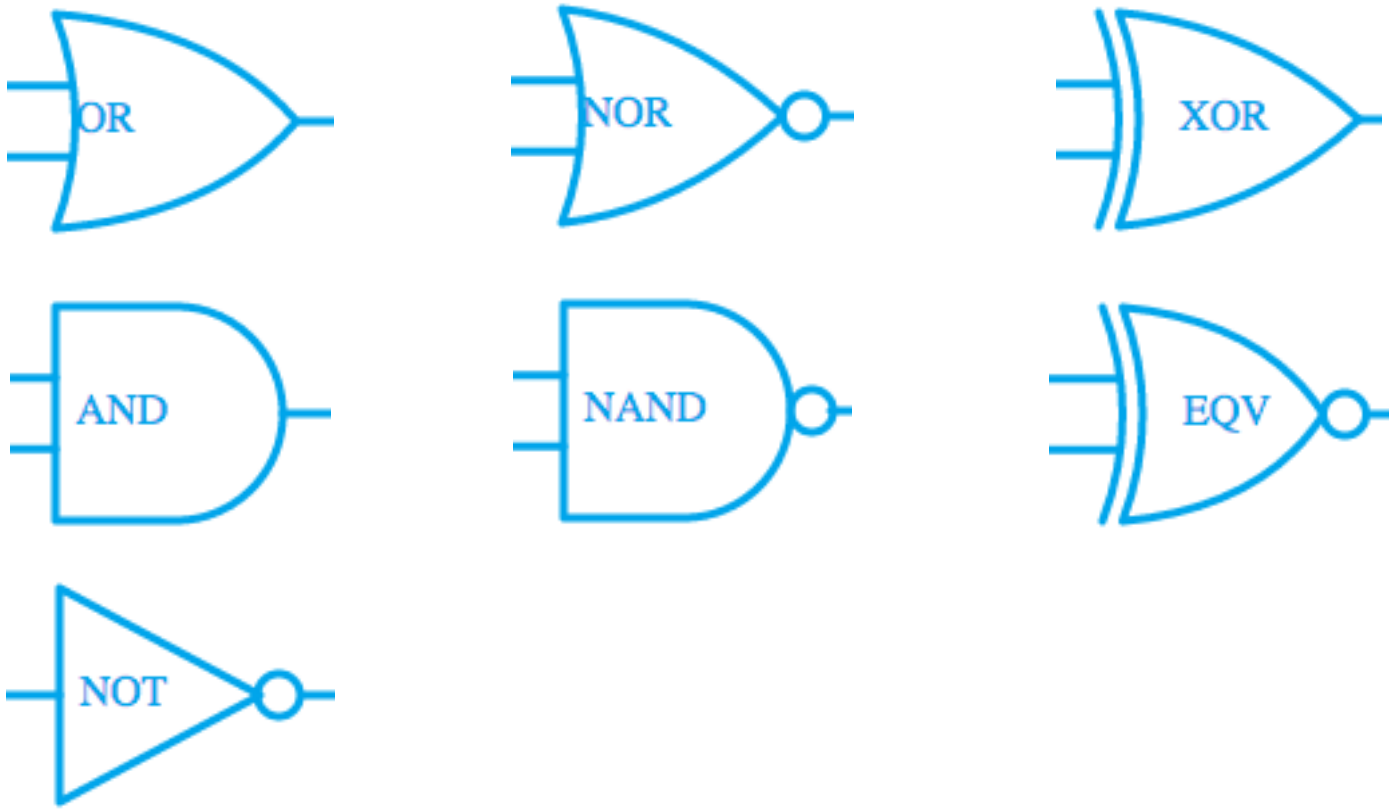
- Portit toteutetaan erilaisin transistorien ja diodien kytkennöillä
  - Emme mene teknisiin yksityiskohtiin, tämä ei ole tietotekniikan kurssi
  - Tarkastelemme korkeammalla abstraktiotasolla
- Piirit nähtävissä Boolen lausekkeiden vaihtoehtoisena esitysmuotona
  - Oleellista, että lausekkeet toteutettavissa puolijohteiden avulla
- Porttien totuusfunktiot voidaan määritellä ***totuustaulujen*** avulla
- Seuraavassa loogisten funktioiden NOT, AND, OR, NAND (= Not AND), NOR (= Not OR), XOR (=eXclusive OR) ja EQV (=EQVivalence) totuustaulut, Boolen lausekkeet ja käytetyt piirrossymbolit

Portteja

x	y	x AND y xy	x OR y x + y	x NAND y ~(xy)	x NOR y ~(x + y)	x XOR y x ⊕ y	x EQV y ~(x ⊕ y)
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

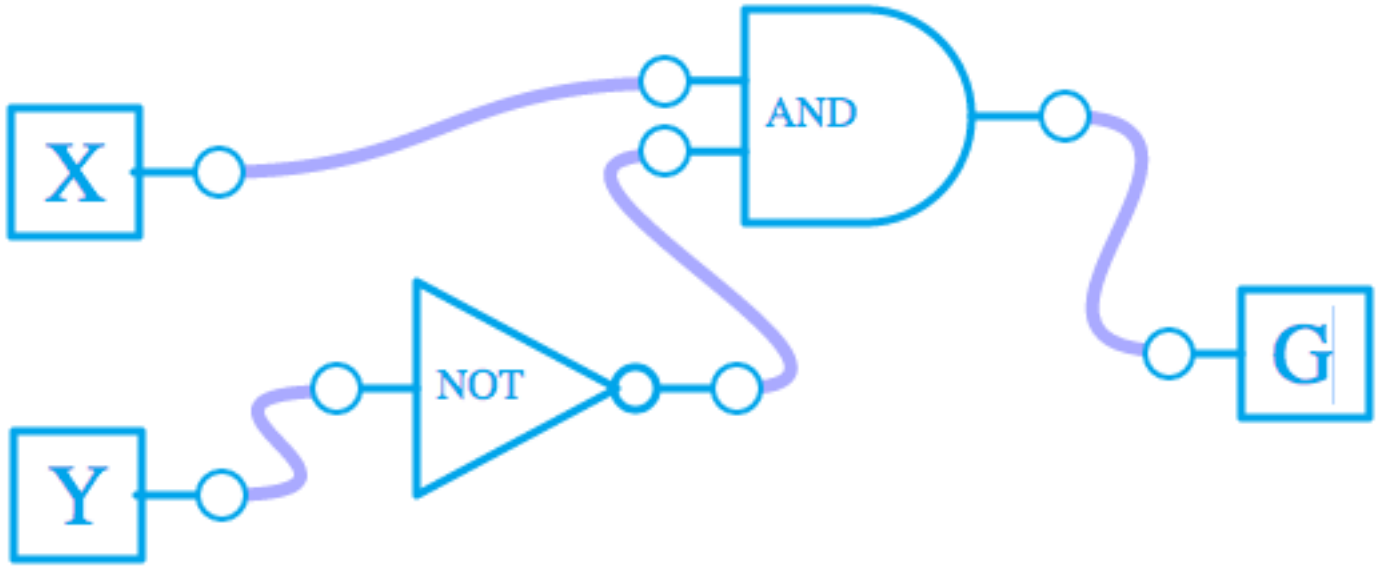
x	NOT x ~x
0	1
1	0

Porttien piirrossymbolit



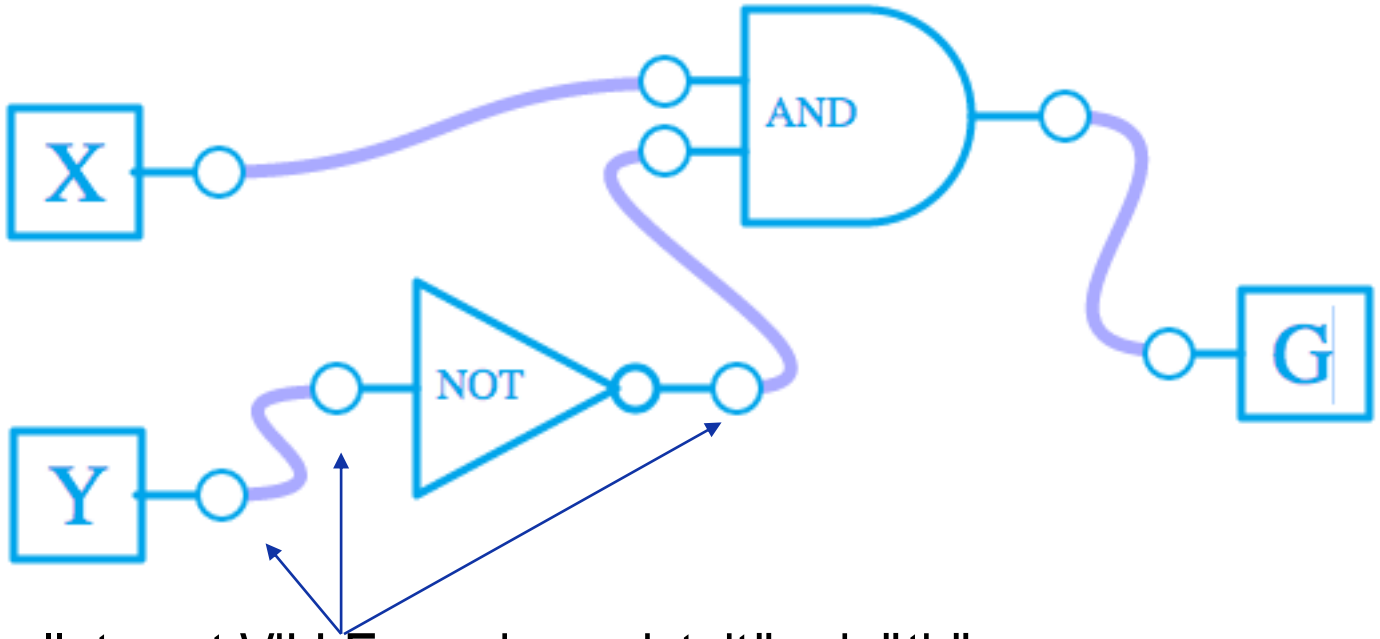
Looginen piiri - esimerkki

- Piiri lausekkeelle $g(x, y) = x \text{ AND NOT } y$



Looginen piiri - esimerkki

- Piiri lausekkeelle $g(x, y) = x \,\mathrm{AND}\, \mathrm{NOT}\, y$



- ”Ylimääräiset” pallot ovat ViLLEn raahauspisteitä, eivätkä varsinaisesti kuulu piirin piirtoon

Signaalista loogisessa piirissä

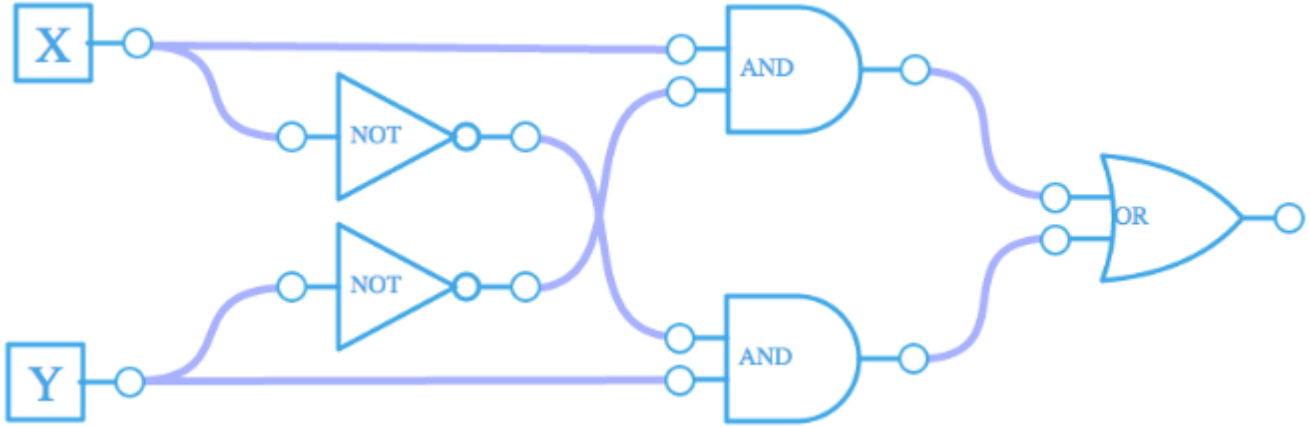
- Signaalin haaroitus on sallittua:
  - Yhdestä kohdasta lähtevä signaali kopioituu identtisenä jokaiseen lähtevään haaraan
- Signaalin yhdistämisessä tapahtuu aina jokin looginen operaatio
- Looginen operaatio on aina merkittävä asianmukaisella portin symbolilla

Loogisista porteista

- Perusportteja yhdistelemällä voidaan rakentaa loogisia piirejä moniin eri tarkoituksiin
- Piirit esitetään Boolen lausekkeina
- Lausekkeita voidaan siis sieventää Boolen algebran avulla
- Sieventämisen tarkoitus: tarvittavan ”raudan” määrää voidaan vähentää matematiikkaa käyttäen

Tarvittavat portit

- Vähemmällä määrällä perusportteja tullaan toimeen
- Esim. XOR muiden porttien avulla:



- Kaikki perusportit voidaan esittää vain NAND-portteja (tai NOR-portteja) käyttäen
  - Jokainen tietokoneen piiri voitaisiin rakentaa vain näillä porteilla

Miten piirejä muodostetaan?

- Piiri voidaan muodostaa ”keksimällä”
- On systemaattisia tekniikoita, katsotaan esimerkkiä:
- Muodostetaan piiri funktiolle $f = f(x, y, z)$ jolla on oikealla oleva totuustaulu
- Lauseke voidaan lausua taulukon avulla:
  - Valitaan x, y, z-kombinaatiot joilla f=1
  - Jos syötteen arvo on tässä 1, syötteen symboli mukaan sellaisenaan, jos [illegible] on 0, syöte komplementoidaan
  - Rivin (ehkä komplementoidut) syötteet yhdistetään AND-operaatiolla
  - Rivit yhdistetään OR-operaattorilla

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$(\text{NOT } x \text{ AND } y \text{ AND } z) \text{ OR} (x \text{ AND NOT } y \text{ AND } z) \text{ OR} (x \text{ AND } y \text{ AND NOT } z)$$

Lisää aiheesta:

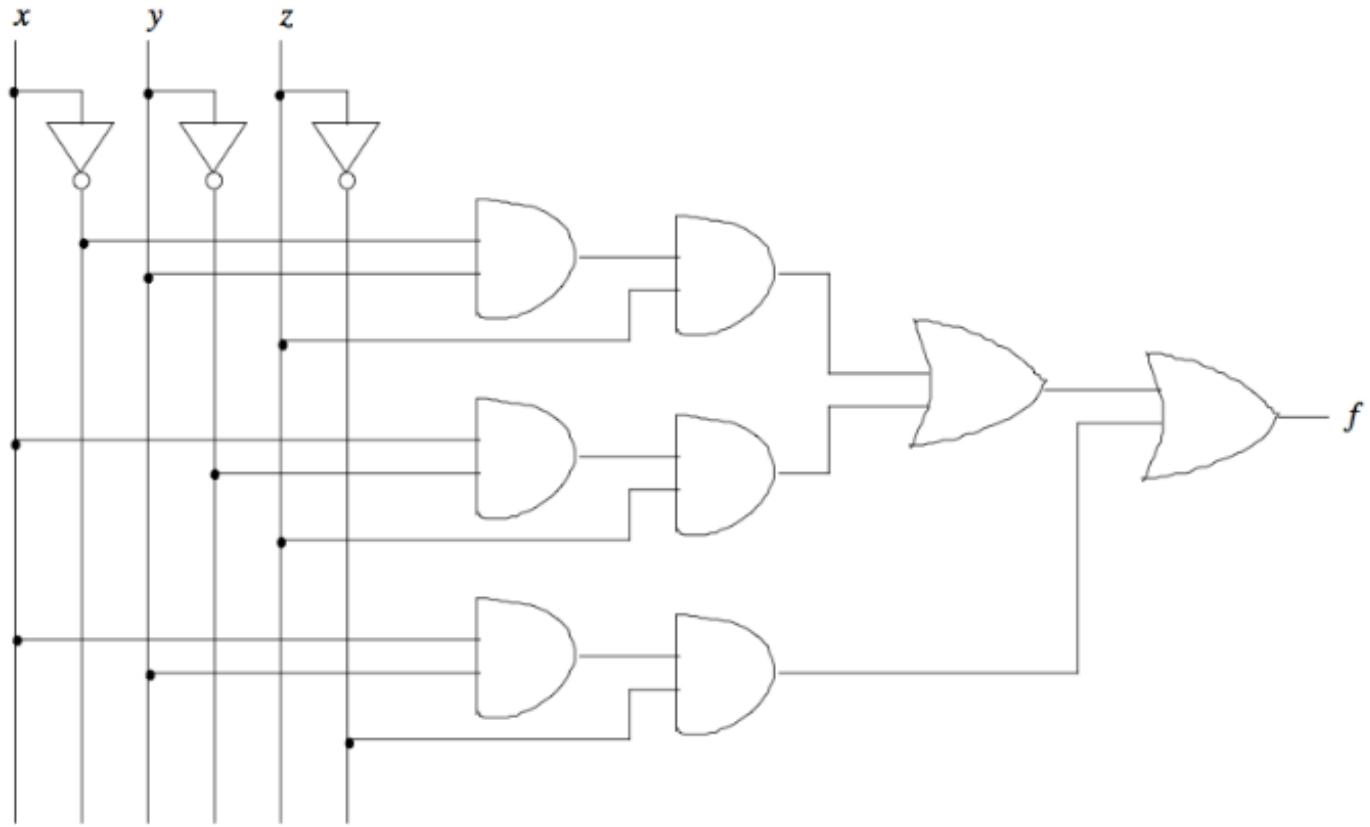
https://fi.wikipedia.org/wiki/Disjunktiivinen_normaalimuoto

Miten piirejä muodostetaan?

$$(\text{NOT } x \text{ AND } y \text{ AND } z) \text{ OR } (x \text{ AND NOT } y \text{ AND } z) \text{ OR } (x \text{ AND } y \text{ AND NOT } z)$$

- Eli

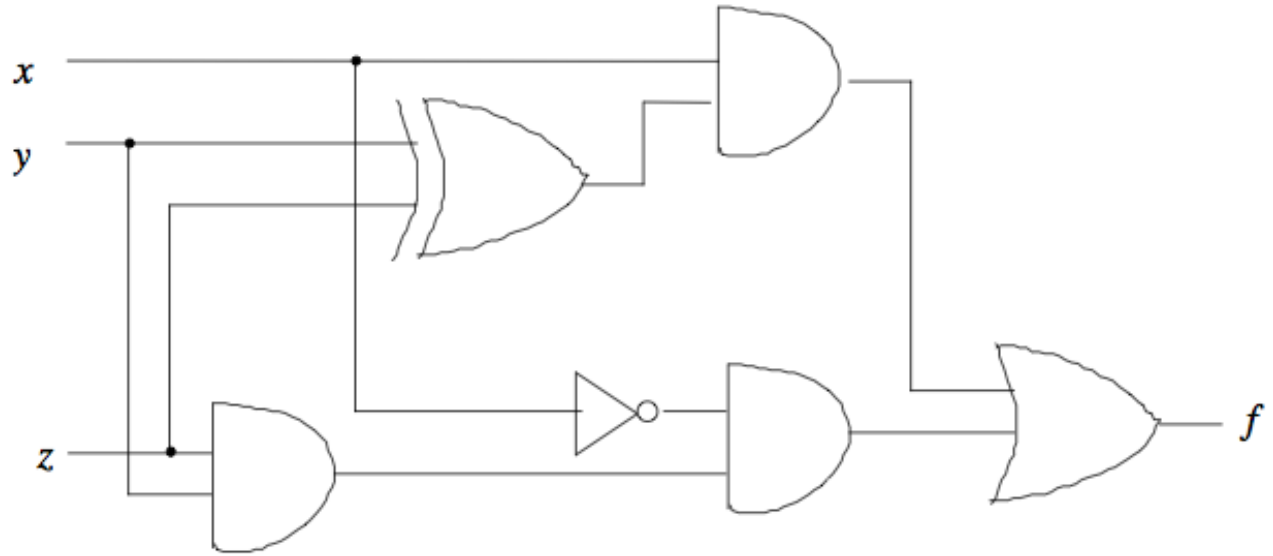
$$f(x, y, z) = \bar{x}yz + x\bar{y}z + xy\bar{z}$$



Miten piirejä muodostetaan?

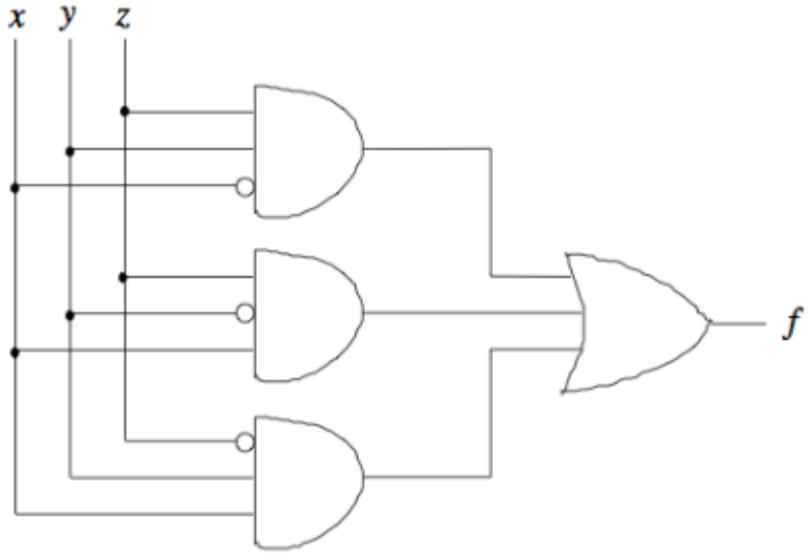
- Toisaalta

$$\begin{aligned} f(x,y,z) &= \bar{x}yz + x\bar{y}z + xy\bar{z} \\ &= \bar{x}yz + x(\bar{y}z + y\bar{z}) \\ &= \bar{x}yz + x(y \oplus z) \end{aligned}$$



Piirien piirtotekniikkaa

- AND- ja OR-porteille voidaan sallia useita syötteitä
- NOT-portti voidaan esittää pienellä ympyrällä syöttölinjassa
- Aikaisemmalle piirille saadaan siis:

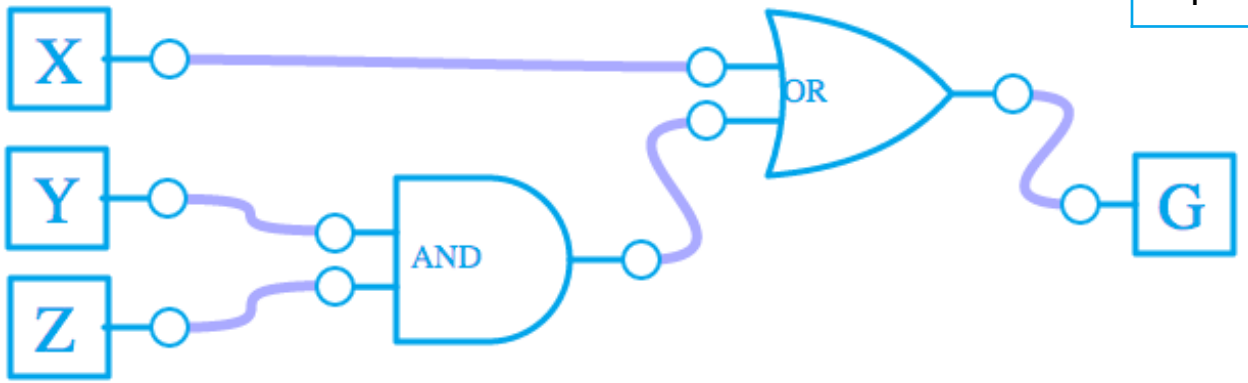


- Jos ympyrän ja portin välissä on lyhyt viiva, kyseessä on ViLLEn piirtämä kaavio, jossa nämä ympyrät tarkoittavat kytkentöjen ”raahauspisteitä”. Esim. seuraava kalvo

Piirin muodostus - esimerkki

- Muodostettava piiri funktiolle $g = g(x, y, z)$
- Huomataan, että g saa arvon 1, kun x = 1 (riippumatta y:n ja z:n arvoista) ja x:stä riippumatta kun y=z=1
- ”Keksitään” siis $g(x, y, z) = x \,\mathrm{OR}(y \,\mathrm{AND}\, z)$

x	y	z	g
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



Tietokoneen komponentteja

Tietokoneen komponentteja

- Tietokone on periaatteessa suuri määrä portteja sisältävä looginen piiri
- Rakenne on erittäin modulaarinen, laajempia komponentteja:
  - Aritmetiikasta huolehtiva yksikkö
    - Kahden komplementtilukujen yhteen- ja vähennyslasku laitteistollisesti
    - Kerto- ja jakolasku matalan tason ohjelmilla tai laitteistollisesti
    - Korkeamman tason laskenta konekielellä tai korkeamman tason ohjelmilla
  - Muisti
    - Tietojen automaattisen käsittelyn välttämätön edellytys
    - Kohta tarkastellaan yhden bitin muistavaa laitetta (ns. kiikku) sekä n-bittisen rekisterin konstruoimista kiikuista

…Tietokoneen komponentteja

- Väylät
  - Siirtävät tietoa komponenttien välillä
  - Jaettavissa loogisesti tieto-, ohjaus- ja osoiteväyliin
    - Tieto- ja osoiteväylien avulla tiedon siirto rekisterien ja keskusmuistin välillä
    - Ohjausväylän avulla toteutetaan koneen ohjelmoitavuus
- Loogiset vertailut
  - Mahdollistavat välttämättömien valinta- ja toistorakenteiden toteutuksen
  - Kurssilla tarkasteltavat operaatiot ovat kahden komplementtilukujen nolluuden ja negatiivisuuden tarkastaminen

…Tietokoneen komponentteja

- Rakenne on erittäin modulaarinen, komponentteja
  - Kello
    - Tahdistaa eri komponenttien toimintaa
    - Kellotaajuus määrää koneen nopeuden
    - Kellotaajuuden nostoa rajoittavat komponenttien fysikaaliset ominaisuudet
  - Syöttö ja tulostus
    - Näyttö ja näppäimistö loogisessa mielessä yksinkertaisia
    - Käytännössä tärkeitä komponentteja

Yhteenlasku

Yhteenlasku

- Yhteenlasku kahden komplementtiluvuilla toteutetaan tavallisella ”kynällä ja paperilla allekkain” –algoritmilla
  - Sarakkeessa olevat numerot lasketaan yhteen
  - Lisätään mahdollinen muistinumero edellisestä sarakkeesta
  - Tulos kirjoitetaan viivan alle
  - Mahdollinen syntynyt muistinumero seuraavaan sarakkeeseen

Puolisummain

- Merkitään kahden bitin yhteenlaskun 2-bittisen tuloksen
  - ensimmäistä bittiä m:llä (muistinumero)
  - jälkimmäistä bittiä s:llä (sarakesumma)
- Yhteenlaskua $\mathrm{x+y} = \mathrm{m,s}$ kuvaa totuustaulu:
- Taulusta nähdään että

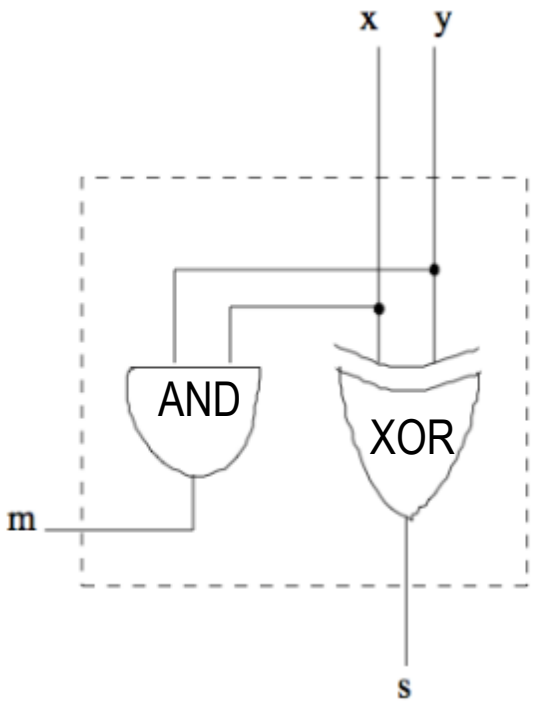
$$m = xy$$

$$s = x \oplus y$$

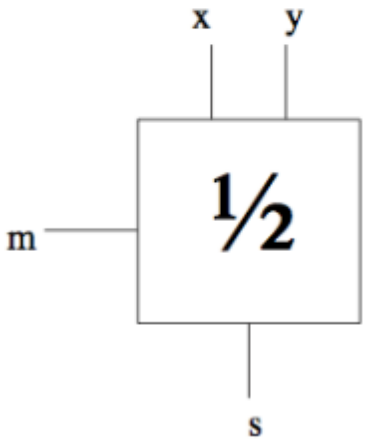
- Tällainen laite on ns. ***puolisummain*** (*half-adder*)

x	y	m	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Puolisummain

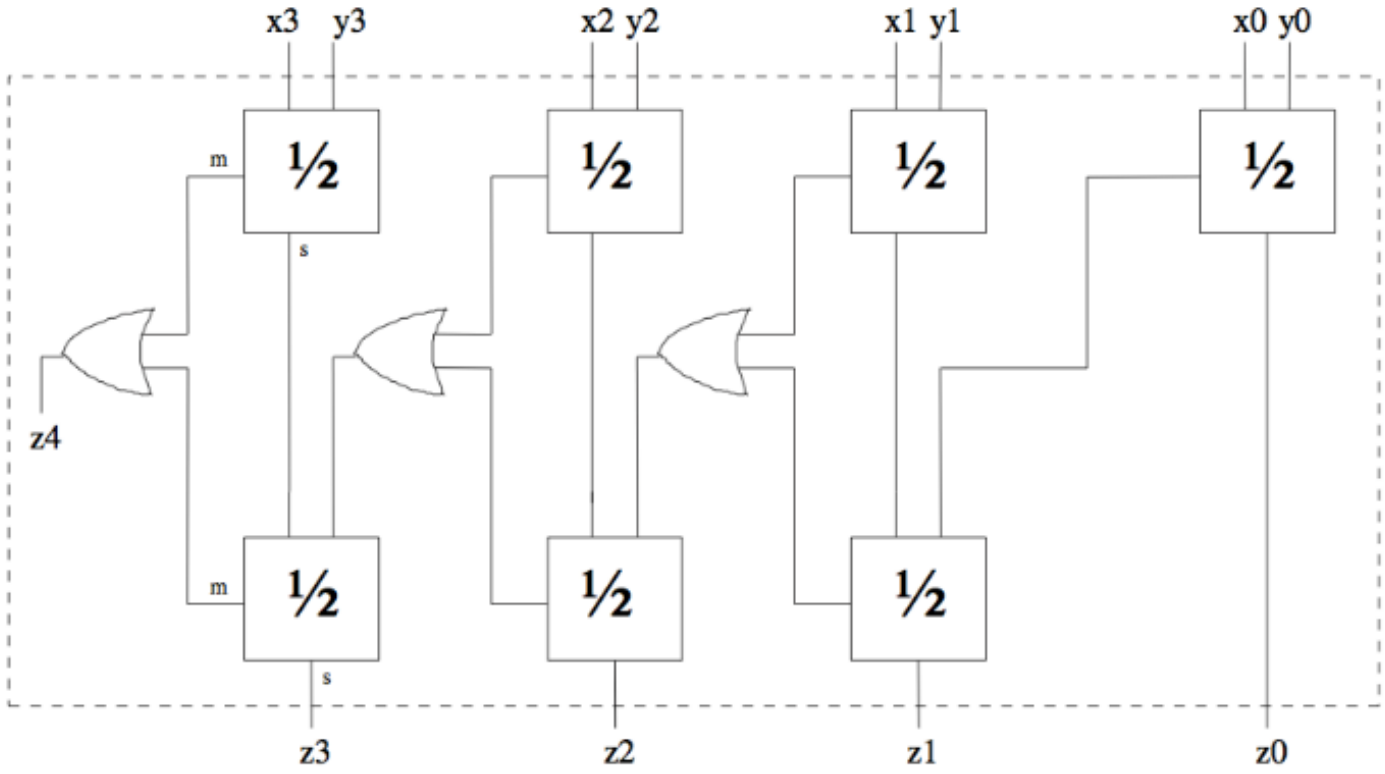


Merkitään



Pidempien lukujen laskeminen

- Pidempien lukujen laskeminen vaatii useita puolisummaimia
- n-bittinen ***kokosummain*** (*full adder*) konstruoidaan modulaarisesti 2n-1:stä puolisummaimesta. Alla 4-bittinen kokosummain.



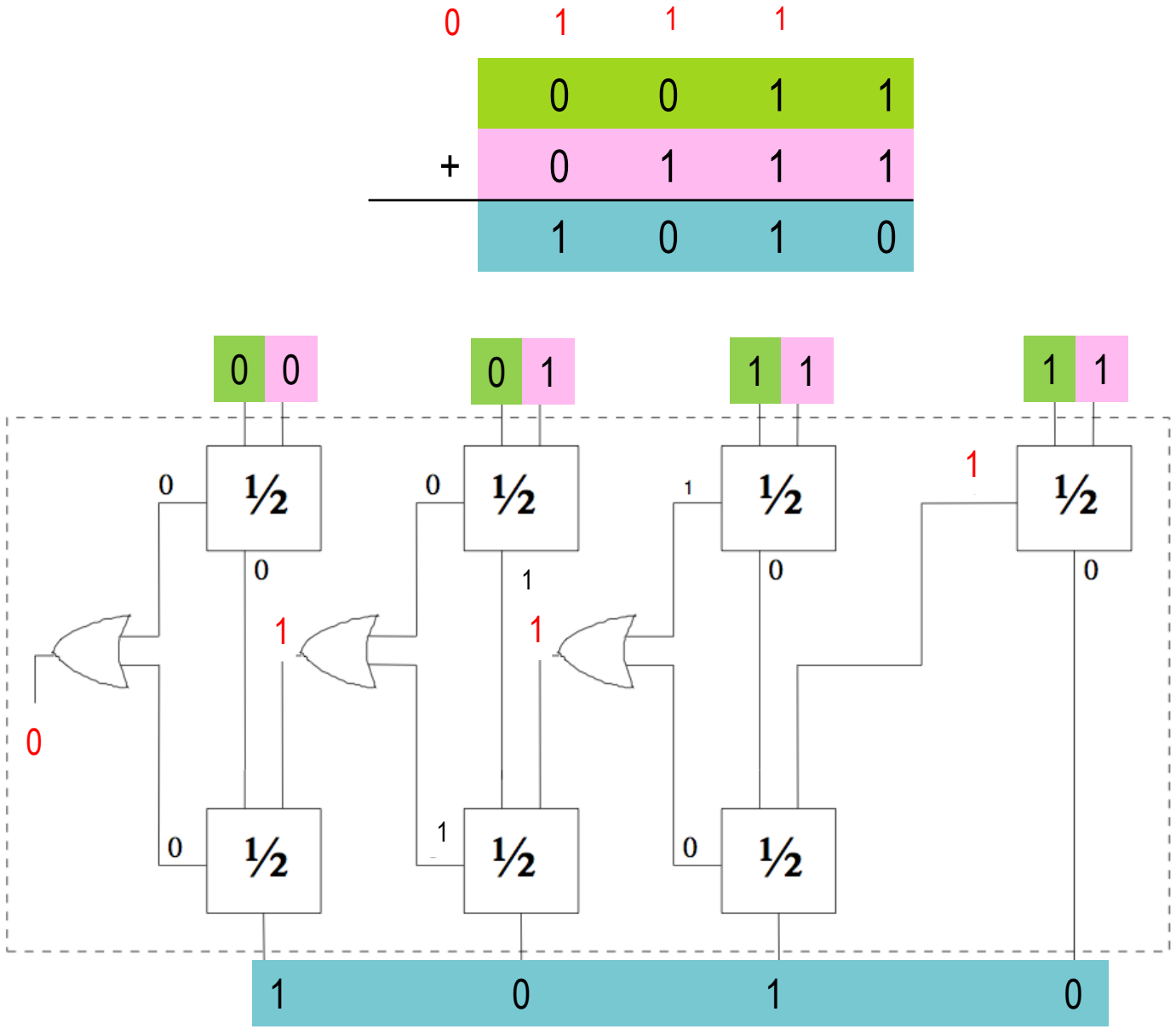
4-bittinen kokosummain

- Piiri laskee yhteenlaskun

  $$x_3x_2x_1x_0 + y_3y_2y_1y_0 = z_4z_3z_2z_1z_0 \cong z_3z_2z_1z_0 \pmod{2^4}$$

- Ulostulo z_4 siis vuotaa 4-bittisillä luvuilla
- Kahden komplementin luvuilla 4 bitillä voi esittää luvut -8…7
  - Yhteenlaskun tulos ei välttämättä mahdu 4 bittiin, tapahtuu ylivuoto ja tulos virheellinen (mutta oikein mod 2^4)
  - Huomattava, että tulos voi olla oikein vaikka $z_4 = 1$
    - Esim. kahden komplementtiesityksillä 1_{10}-1_{10} = 0001_2+1111_2=0000_2, $z_4 = 1$

4-bittinen kokosummain - esimerkki



Turun yliopisto
University of Turku

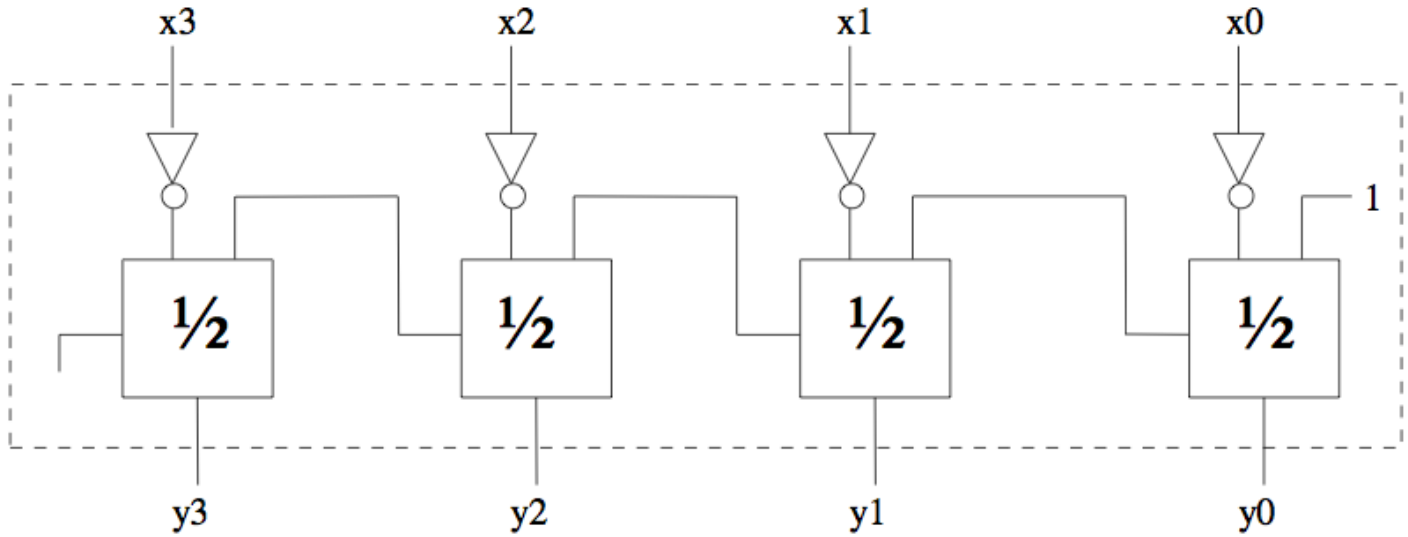
Kokosummaimen looginen piiri

- Looginen piiri kuvaa ”kynällä ja paperilla allekkain” –algoritmin
- Algoritmin looginen piiri on helppo toteuttaa elektroniikalla
  - Kussakin sarakkeessa (oik. puoleisinta lukuun ottamatta) suoritetaan kaksi puoliyhteenlaskua: lukujen bittien yhteenlasku ja muistinumeron lisäys
  - OR-portti riittää, koska (nollasta poikkeava) muistinumero voi syntyä vain toisessa sarakkeen puoliyhteenlaskuista
    - Jos ylemmässä ½ -yhteenlaskussa syntyy muistinumero m=1, on tulos s=0, jolloin alemmassa ½-yhteenlaskussa toinen yhteenlaskettava on 0 joten sen syntyvä muistinumerokin on 0
    - Jotta alemmassa ½-yhteenlaskussa syntyisi muistinumero m=1, pitää yhteenlaskettavien olla 1 ja 1; tällöin ylemmässä ½-yhteenlaskussa syntynyt tulos on siis1, jolloin syntynyt muistinumero on 0

Vähennyslasku

Vähennyslasku

- Vähennyslasku voidaan toteuttaa kokosummaimen avulla muuntamalla toinen yhteenlaskettava vastaluvukseen:
  - Kahden komplementtiluvuilla tämä onnistuu muuttamalla bitit komplementeikseen ja lisäämällä tulokseen ykkönen



Vähennyslasku

- Voidaan toteuttaa myös suoraan kuten ”kynällä ja paperilla allekkain”-algoritmi
  - Kun nollasta vähennetään ykkönen, täytyy lainata 10_2 seuraavasta sarakkeesta

Puolivähennin

- Merkitään kahden bitin vähennyslaskun 2-bittisen tuloksen
  - ensimmäistä bittiä l:llä (lainaus)
  - jälkimmäistä bittiä e:llä (sarake-erotus)
- Vähennyslaskun x-y totuustaulu:
- Taulusta nähdään että

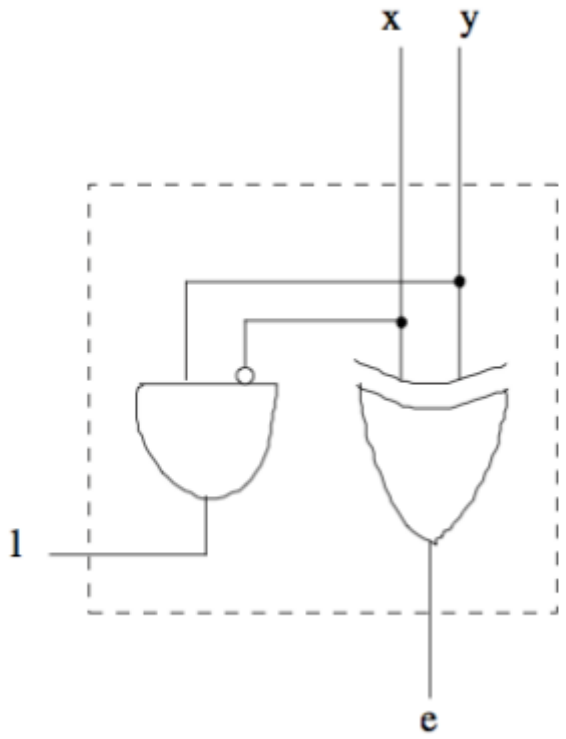
$$l = \overline{x}y$$

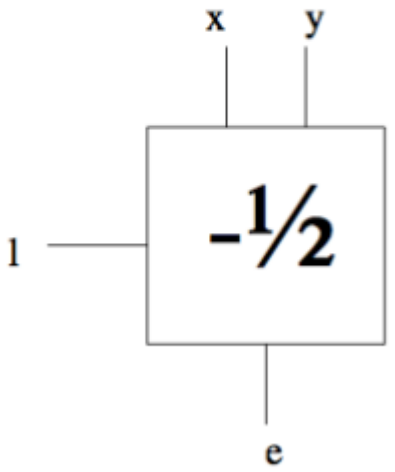
$$e = x \oplus y$$

- Tällainen laite on ns. ***puolivähennin*** (*half-substractor*)

x	y	l	e
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Puolivähennin





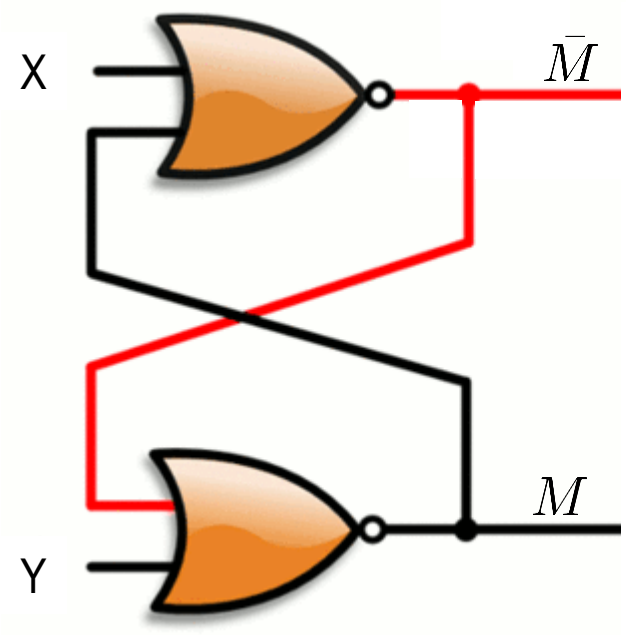
- Kokovähennin voidaan konstruoida modulaarisesti puolivähentimistä

Kiikku

Kiikku

- Muisti: jokin tapahtuma riippuu syötteen lisäksi myös aiemmin tapahtuneesta
- Muistin toteutus vaatii siis loogisen piirin, jolla on historia
  - Sen toiminta riippuu aiemmin tapahtuneesta (eli aiemmasta syötteestä)
- Piirin historia kuvataan sen tilana, aiempien toimintojen lopputuloksena
- Yksinkertaisin muistava piiri on yhden bitin muistava ***kiikku*** (*flip-flop*)

Kiikun muistiosa



- Ristiinkytketty kahden NOR-portin piiri
- x ja y ovat muistiosan syötteitä, M muistettu bitti

Muistiosan toiminta totuustaulun avulla

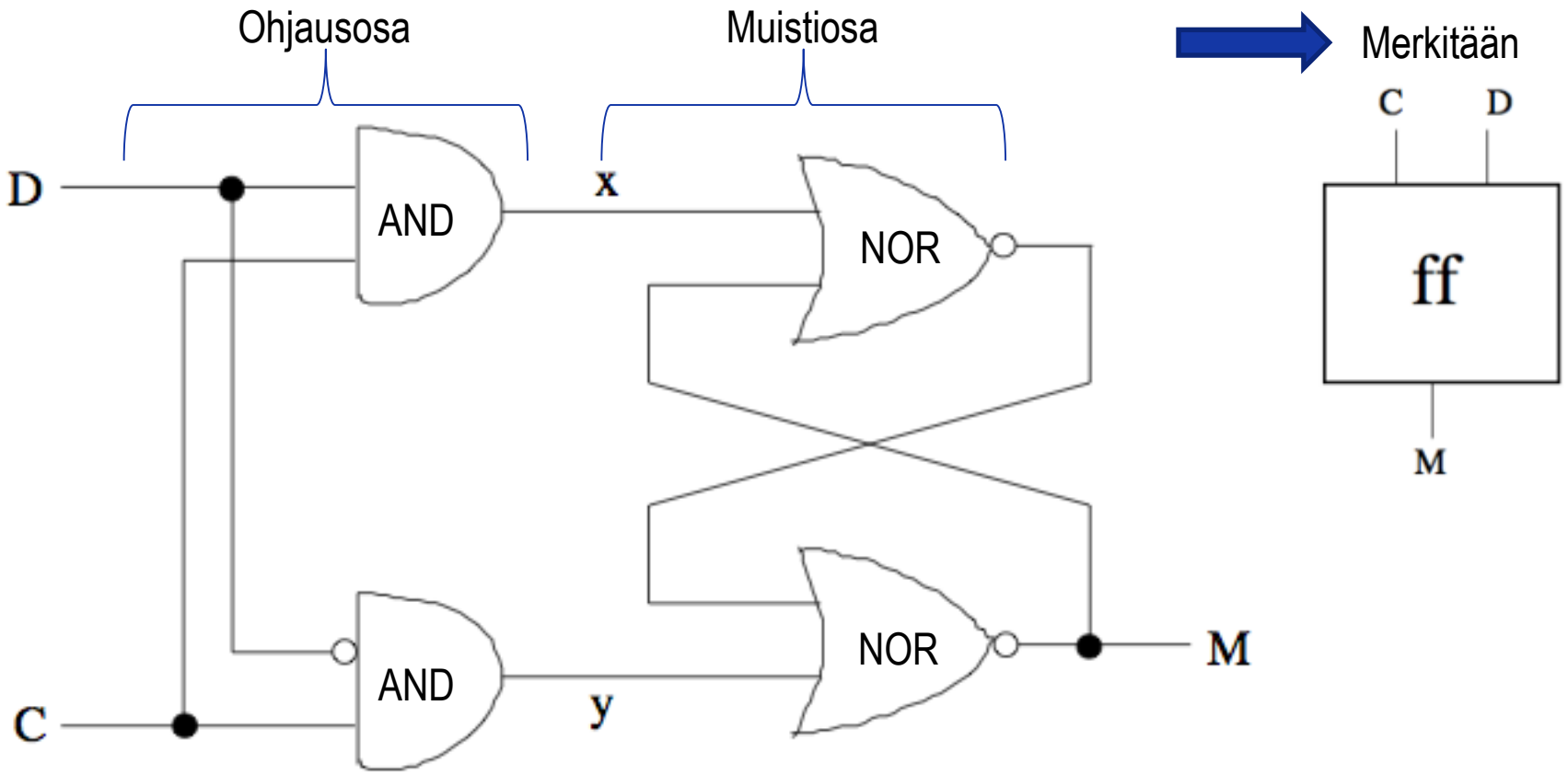
M_t on M:n vanha arvo ajanhetkellä t ja M_{t+1} M:n uusi arvo ajanhetkellä t+1

x	y	M_t	M_{t+1}	
0	0	0	0	Pidä tilaa
0	0	1	1	
0	1	0	0	Aseta 0
0	1	1	0	
1	0	0	1	Aseta 1
1	0	1	1	

Kiikku

- Ongelma: syötteet x=y=1 ei sallittuja, tällöin $M = \bar{M} = 0$
  - Periaatteessa ~M-ulostulosta voitaisiin luopua, mutta
  - Fyysisissä piireissä suurin ongelma on, että kun siirrytään tilasta (x,y)=(1,1) tilaan (x,y)=(0,0) (eli pidä tilaa), niin x tai y saattavat (hyvin useinkin) mennä 0:ksi eri aikaan
  - Tällöin ei voida ennustaa, tuleeko M:n lopulliseksi arvoksi 1 vai 0!
- Ratkaisu: Kiikku muodostetaan kahdesta osasta: ohjausosasta ja muistiosasta

Kiikku



Kiikku

- Kiikulla on kaksi syötettä: kontrollibitti C ja databitti D
- Tilaa kuvaa tulosbitti M (komplementti ~M voidaan myös ottaa)
- Kiikkua käytetään:
  - Kun databitti D halutaan kirjoittaa muistiin M, asetetaan kontrollibitti C ykköseksi, jolloin M saa saman arvon kuin D
  - Kun D on kirjoitettu muistiin, kontrollibitti C asetetaan nollaksi. Tällöin M ei muutu, vaikka D muuttuisikin.

Kiikku

x	y	M_t	M_{t+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1

- Jompikumpi linjoista x=CD ja y=C~D on aina nolla
- Yhdistelmä x=y=1 ei mahdollinen
- Kun muistiosan ohjaus (x,y)=(0,0), ristiinkytkentä säilyttää entisen tilansa (oli se 0 tai 1)
- Ohjauksella (x,y)=(1,0) asettuu ristiinkytkentä tilaan M=1 ja ohjauksella (x,y)=(0,1) tilaan M=0
- Se toimii siis halutulla tavalla:
  - Kun C=0, muistiosan ohjaus (x,y)=(0,0) säilyttää ristiinkytkennän tilan ja piiri muistaa M:n
  - Kun c=1, muistiosa ohjaus (x,y)=(D,~D) pakottaa piirin tilaan M=D

Kiikun toiminta

- Kiikun (ja muiden loogisten piirien) tarkastelussa huomattava, että peräkkäinen johtojen seuranta johtaa virheelliseen tulkintaan
  - Etenkin piirin sisältäessä ristiinkytkentöjä
- Ideana: tietyillä syötteillä piiri jää stabiiliin tilaan
  - Kiikun piirissä on ristiinkytkentöjen takia periaatteessa mahdollista, että M jäisi heilahtelemaan tilojen 0 ja 1 välillä, jolloin piiri olisi käyttökelvoton
  - Ohjausosa rakennettu niin, että näin ei voi käydä

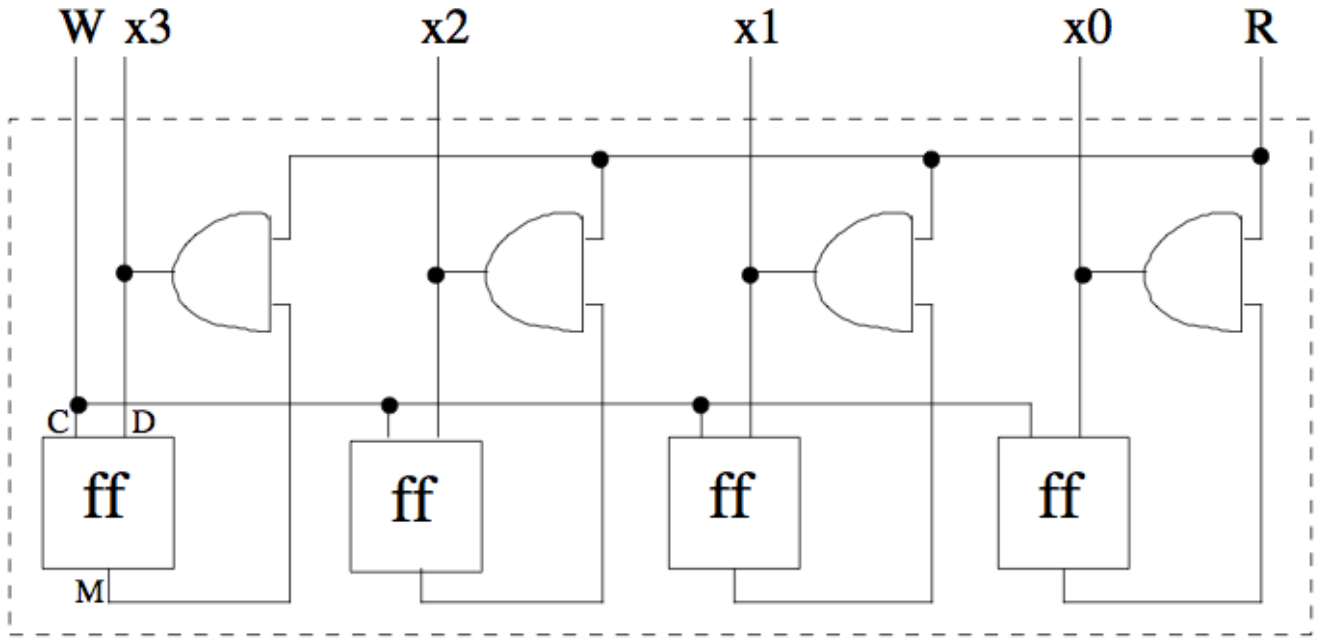
Kiikun toiminta – ilman totuustaulua

- *Tapaus 1*: **Muista bitti M**. *Tulee asettaa C=0*. Silloin x=y=0 D:stä riippumatta.
  - Muistiosan ylempään NOR-porttiin menee 0 ja M ja ulostulo on siis ~M
  - Alempaan NOR-porttiin menee syötteenä ~M ja 0, joten ulos tulee M.
  - M:n arvo ei muutu eli piiri muistaa M:n
- *Tapaus 2*: **Tallenna M=0**. *Tulee asettaa C=1 ja D=0*. Nyt x=0 ja y=1.
  - Koska y=1, alemmasta NOR-portista tulee aina ulos 0
  - Näin ollen M saa arvon 0 riippumatta M:n vanhasta arvosta
- *Tapaus 3*: **Tallenna M=1**. *Tulee asettaa C=1 ja D=1*. Nyt x=1 ja y=0.
  - Alemman NOR-portin ylempi syöttö on aina 0 ja koska y on aina 0, alemman portin ulostulo on aina 1
  - M saa siis arvon 1 M:n vanhasta arvosta riippumatta

Rekisteri

Rekisteri

- Useamman bitin mittaisen jonon (ns. sanan) tallentamiseen käytetään ***rekisteriä***
  - Rekisteri rakennetaan kiikuista
- Nelibittinen rekisteri:



Rekisteri

- Rekisterissä on neljä kiikkua
  - Kytketty 4 linjan väylään, jota käytetään lukemiseen ja kirjoittamiseen
- W (write) ja R (read) toimivat ohjausbitteinä
  - Lepotilassa R = W = 0
  - Kun W = 1 ja R = 0, väylällä oleva luku $x_3x_2x_1x_0$ kirjoittuu rekisteriin (kukin bitti omaan kiikkuunsa)
    - Aikaisemmin tallennettu bitti ei häiritse, sillä AND-portti antaa arvon 0 R:n arvon ollessa 0
  - Kun R = 1 ja W = 0, piirii antaa johtoihin x_3, x_2, x_1, x_0 aiemmin tallennetut bitit

Lisää komponentteja

Väylät

- Eri komponentteja yhdistäviä tiedonsiirtoyhteyksiä kutsutaan ***väyliksi*** (*bus*)
- Väylä: kokoelma komponentteja yhteenliittäviä johtoja
- Toiminnallisesti kolmenlaisia:
  - Tietoväylä – käytetään varsinaiseen tiedon siirtoon
  - Ohjausväylä – ohjaa tiedon siirtoa, mm. käynnistäen siirron
  - Osoiteväylä – määrää minne tai mistä tieto siirretään
- Sama fyysinen väylä voi (joskus) toimia useammassa tehtävässä

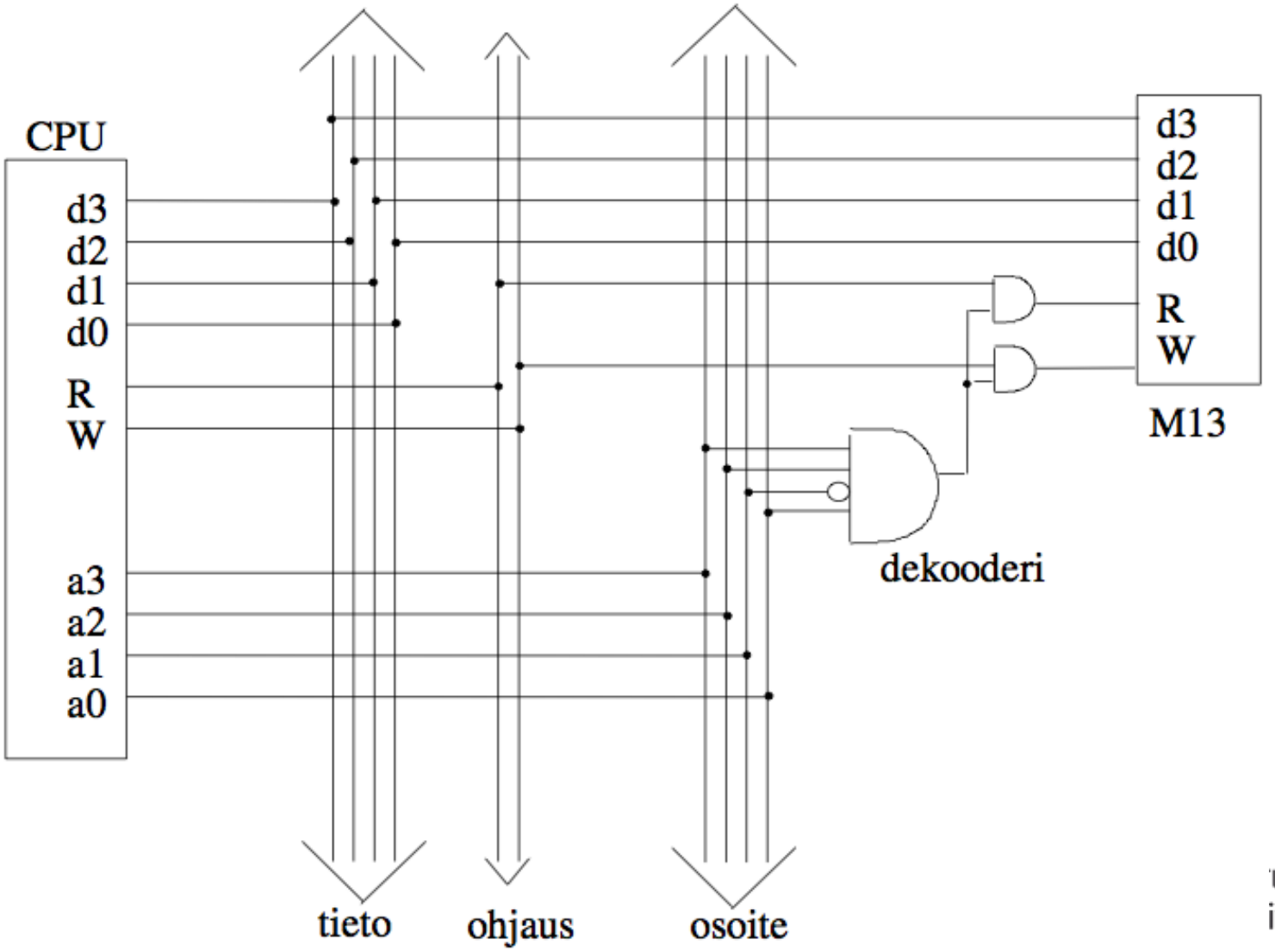
Väylät

- Kun tietoa halutaan tallentaa johonkin muistipaikkaan
  - Tallennettava tieto lähetetään tietoväylään
  - Kyseisen muistipaikan osoite lähetetään osoiteväylään
  - Tieto toimenpiteen laadusta lähetetään ohjausväylään
- Tietoväylä voi siirtää ainoastaan yhden sanan kerrallaan
  - Välttämätöntä määrätä mitkä komponentit saavat käyttää väylää milloin
    - Voidaan tehdä (periaatteessa) liittämällä komponentit väylään AND-piirillä, jonka toisena syötteenä aktivoiva ohjaussignaali
    - Kun ohjaussignaali päällä, komponentti aktivoituu ja tieto siirtyy väylään
  - Todellisuudessa usean komponentin liittäminen väylään ei onnistu AND-piirillä vaan esim. kolmitilapuskureilla ja multiplekserillä

Väylät

- Tarkastellaan keskusyksikköä, joka kytketty tieto-, ohjaus- ja osoiteväylään (seuraava kalvo)
- Kuhunkin väylään kytketty lukuisia rekistereitä, joilla oma osoite
- Kullakin muistipaikalla oma dekooderi, joka tunnistaa ko. muistipaikan
  - Tunnistaminen: kun dekooderille annetaan ko. muistipaikan osoite binäärisenä, sen tuloksena tulee olla 1; kaikilla muilla osoitteilla 0

Väylät



Väylät

- Kuvassa muistipaikan M13 ohjauskäskyt tunnistava dekooderi
  - Aktivoituu osoitteesta 13 = 1101_2
  - Tällöin a3=1, a2=1, a1=0, a0=1
- Ohjausbiteillä R ja W ohjataan tiedon siirron suuntaa
  - Tieto: d (data), bitit d3, d2, d1, d0
- Esim. kun CPU:ssa R=1, W=0, niin M13:sta R=0 ja W=1
  - Eli CPU:n data kopioituu muistipaikkaan M13 (M13:n data muuttuu, CPU:n ei)
- Esim. kun CPU:ssa R=0, W=1, niin M13:sta R=1 ja W=0
  - Eli muistipaikan M13 data kopioituu CPU:n databitteihin (CPU:n data muuttuu, muistipaikan M13 ei)

Ohjauslogiikka

- Yhteen liitetyt komponentit tarvitsevat ohjaussignaaleja toiminnan valvontaan
- Ohjaussignaali määrää esim.
  - Milloin kiikku voi tallentaa syötteensä
  - Mikä komponentti voi siirtää tietoaan väylään
- Ohjaussignaalit tulevat tietokoneen ohjauslogiikkakomponentista
  - Vastaa oikean tiedon siirtämisestä sopivana aikana komponentilta toiselle
  - Varmistaa tietoihin kohdistettavien operaatioiden suorituksen

Ohjauslogiikka ja kello

- Ohjauslogiikkakomponentin tärkeä osa on kello
- Kello tuottaa sakara-aaltoa, vuorottelevia ykkös- ja nollasignaaleja tietyllä nopeudella
- Näillä varmistetaan, että
  - tietokoneen komponentit toimivat synkronisesti
  - kaikki komponentit saavat riittävästi aikaa operaatioidensa toteuttamiseen
- Yhden signaalin aikana tietokoneessa toteutetaan yksi alkeistapahtuma
  - Tietokone toimii siis sitä nopeammin, mitä suurempi kellotaajuus on
  - Kellotaajuutta rajoittavat puolijohdekomponenttien ominaisuudet

Tulossa

- Huomenna tutoriaali päivän aiheesta
- Ensi viikolla aiheena mikro-ohjelmointi