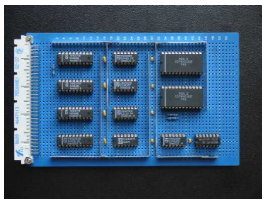
Digital- och datorteknik



Föreläsning #3

Biträdande professor Jan Jonsson

Institutionen för data- och informationsteknik
Chalmers tekniska högskola

Logikgrindar

Från data till digitala byggblock:

Kursens inledande föreläsningarna har introducerat bitar, bitsträngar och binära koder, d v s det som representerar de data som vår dator kommer att bearbeta.

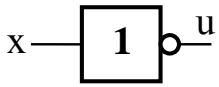
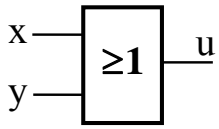
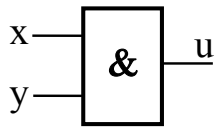
Nästa steg blir nu att ta fram de byggblock (digitala kretsar) som senare på lämpligt sätt kombineras till större byggblock och blir det som utgör datorns grundläggande delar.

För detta syfte behöver vi logikgrindar, digitala kretselement som utför enklare logiska operationer. I en logisk operation representerar varje bit ett sanningsvärde* som kan vara antingen SANT (’1’) eller FALSKT (’0’).

***) Vi har alltså stött på ännu en kodning (tolkning) av binära siffror.**

Logikgrindar

Grundläggande logikoperationer och dess symboler:

Grind (Gate)	Symbol	Funktions-tabell	Booleskt uttryck
INVERTERARE (ICKE, NOT)		x | u 0 | 1 1 | 0	u = x’
ELLER (OR)		x | y | u 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1	u = x + y
OCH (AND)		x | y | u 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1	u = x · y

Funktionstabell:

Beskriver de möjliga logiska värden som kan påföras en grinds ingångar samt det resulterande värdet på grindens utgång.

Logikgrindar

Regelverk för logiska operationer och grindnät:

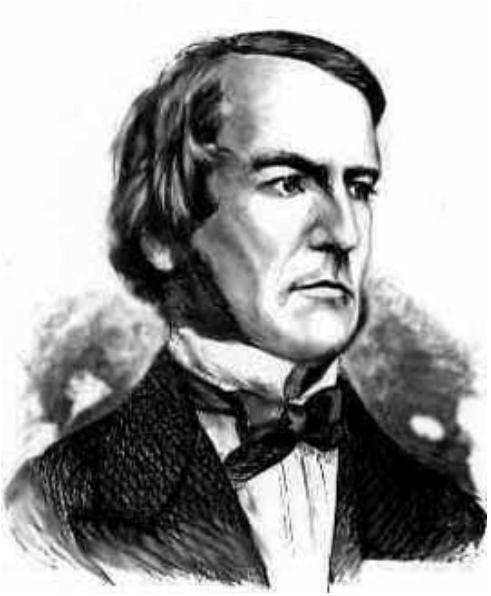
För bearbetning av variabler (symboler) som representerar tal finns regler som beskriver hur dessa variabler får manipuleras, för att exempelvis lösa ekvationssystem och uttrycka formler. Detta regelverk kallas algebra.

Ett motsvarande regelsystem finns för bearbetning av variabler som representerar logiska värden. Dessa regler kan användas för att beskriva och analysera beteendet hos logikgrindar, t ex för påvisa ekvivalens mellan två givna grindnät eller för att ta fram kretslösningar med så få grindar som möjligt.

Detta speciella regelverk kallas Boolesk algebra.

Boolesk algebra

George Boole



Claude Shannon

På 1930-talet jobbade Shannon med switchnät, nät uppbyggda av kretselement som kan växla mellan två tillstånd. Kretselementen bestod vid den tiden av reläer.

För att modellera och analysera kretselementen på en matematisk form använde han den algebra som Boole presenterat i mitten av 1800-talet.

Boolesk algebra

Egenskaper:

Den booleska algebran definieras av

- en uppsättning värden som variabler och konstanter kan anta
- en uppsättning grundläggande operationer som kan utföras på variabler och konstanter
- en uppsättning räkneregler

De värden som kan antas är SANT (1,$\top$) eller FALSKT (0,$\bot$)

De grundläggande operationer som kan utföras är:

ICKE (negation): $\bar{x}$ (*alt*. x' *el*. $\neg x$)

ELLER (disjunktion): x + y (*alt*. x $\vee$ y)

OCH (konjunktion): x $*$ y (*alt*. xy, x $\cdot$ y *el*. x $\wedge$ y)

Boolesk algebra

Räkneregler: (Postulat)

P1. $x + y = y + x$ (kommutativa lagarna)
$x * y = y * x$

P2. $x * (y + z) = x * y + x * z$ (distributiva lagarna)
$x + (y * z) = (x + y) * (x + z)$

P3. $x + 0 = x$ (neutralt element)
$x * 1 = x$

P4. $x + \overline{x} = 1$ (komplementlagarna)
$x * \overline{x} = 0$

Boolesk algebra

Räkneregler: (Teorem)

T5. $x + (y + z) = (x + y) + z$ (associativa lagarna)
$x * (y * z) = (x * y) * z$

T6. $\overline{x} * \overline{y} = (\overline{x + y})$ (De Morgans lagar)
$\overline{x} + \overline{y} = (\overline{x * y})$

T7. $x + x = x$ (idempotent element)
$x * x = x$

T8. $x + 1 = 1$ (annihilerande element)
$x * 0 = 0$

T9. $(\overline{\overline{x}}) = x$ (dubbel negation)

Ett möjligt framtida scenario

För att illustrera hur användbar Boolesk algebra är i digital- och datorteknik, föreställer vi oss ett scenario i en inte alltför avlägsen framtid:

Du är helt ny på ditt jobb som konstruktör av digitala system, och har fått i uppgift att avsluta ett pågående projekt. Allt går bra tills du kommer till en delkomponent som tyvärr använder för många logiska grindar för att projektet skall uppfylla kraven.

Mer specifikt har du fastnat vid funktionen $f = (x+y)(x+z)$ *som tar tre logiska grindar i anspråk, men som du misstänker kan realiseras med färre grindar. Om du hittar en sådan lösning skulle produkten kunna färdigställas (varandes lite billigare och strömsnålare), vilket säkerligen skulle uppskattas av din chef.*

Ett möjligt framtida scenario

Då deadline för uppgiften är redan i morgon, och då du känner dig orutinerad vad gäller denna del av digitalteknik som kallas "kretsminimering", rådfrågar du en kollega som lovar att ta en titt på detta snarast möjligt.

När arbetsdagen är nästan till ända kommer din kollega med en lösning som bara behöver två grindar: $g = x+yz$. Din kollega har dock bråttom hem och hinner inte stanna för att förklara hur han fått fram lösningen, utan säger bara "Trust me!"

Då du vet att din kollega gått på Högskolan i X, och inte på Chalmers, vill du helst verifiera att den nya lösningen ger samma funktion som den första. Du vill ju inte lämna ifrån dig en felaktig design det första du gör på ditt nya jobb …

Ett möjligt framtida scenario

Visa på lämpligt sätt att

$$(x + y)(x + z) = x + y * z$$

Metod 1:

Ställ upp en sanningstabell och visa att de två uttrycken ger samma utfall för varje kombination av uttryckens variabler.

Metod 2:

Visa ekvivalensen algebraiskt genom att tillämpa räknereglerna i den Boolska algebran.

Ett möjligt framtida scenario

En tid har gått och du tänker tillbaka på den lyckade starten på ditt jobb. Fast det retar dig lite att du inte hade egen kunskap för att lösa detta. Därför tar du fram ditt gamla material från kursen i Digital- och datorteknik, och går igenom det i lugn och ro under en helg, så att du kan vara ordentligt förberedd nästa gång en liknande uppgift dyker upp.

Så: hur borde du ha löst uppgiften … ???

Boolesk algebra

SP- och PS-form:

I det framtida scenariot observerade vi att funktionen hos en krets kan uttryckas på två olika former:

- Som en summa av produkter, SP-form, även kallad disjunktiv form. Din kollegas lösning, $x+yz$, är exempel på sådan form.
- Som en produkt av summor, PS-form, även kallad konjunktiv form. Den lösning du ville förbättra, $(x+y)(x+z)$, är exempel på sådan form.

SP-form och PS-form för samma funktion är alltså ekvivalenta, så endera formen kan väljas. Dock kan den ena i praktiken vara mer lämpad, t ex på grund av begränsningar i hur många grindar som totalt får användas, eller på grund av begränsningar i vilken typ av grindar som får användas.

Boolesk algebra

Mintermer och maxtermer:

Utgående från funktionstabellen för en given Booleskt funktion är det möjligt att härleda funktionens SP-form respektive PS-form:

- För SP-form skall vi identifiera de rader i tabellen som har ett funktionsvärde lika med ’1’. Den unika produkten av invariabler för en sådan rad kallas för en minterm. Summan av alla mintermer kallas för SP normal form.
- För PS-form skall vi identifiera de rader i tabellen som har ett funktionsvärde lika med ’0’. Den unika summan av invariabler för en sådan rad kallas för en maxterm. Produkten av alla maxtermer kallas för PS normal form.

Observera att varje minterm och maxterm är unik och innehåller samtliga invariabler.

Boolesk algebra

Ta fram SP normal form för

$$(x + y)(x + z)$$

Kommentar:

Vi väljer SP-form eftersom vi såg att din kollegas lösning, med det minsta antalet grindar, hade den formen.

Karnaughminimering

Minimering av Booleska funktioner:

Om vi inte är nöjda med den lösning som ges av funktionens SP normal form eller PS normal form, kan vi härleda ett minimalt uttryck genom Karnaughminimering.

Denna metod bygger på att man för in funktionstabellen i en matris med invariablernas olika värden längs rader och kolumner.

Varje ruta i matrisen representerar funktionsvärdet för en minterm (för SP form) eller en maxterm (för PS form).

Karnaughminimering

Minimering av Booleska funktioner:

För att ge möjlighet till effektiv eliminering av onödiga variabler skall intilliggande rader respektive kolumner representera termer ordnade enligt Gray-kod, d v s de skiljer sig åt i en bitposition.

Ringa in så stora grupper av 1:or (för SP form) eller 0:or (för PS form) som möjligt, och ta fram uttrycken för dessa.

Gör Karnaughminimering på SP normal form för

$$(x + y)(x + z)$$