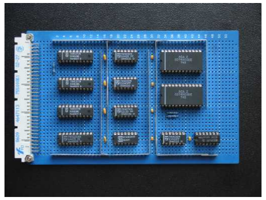
CHALMERS
UNIVERSITY OF TECHNOLOGY

Digital- och datorteknik



Föreläsning #7

Biträdande professor Jan Jonsson

Institutionen för data- och informationsteknik
Chalmers tekniska högskola

Aritmetik i digitala system

Speciella egenskaper:

Systemet arbetar med kodord (s k maskintal) med ett begränsat antal sifferpositioner:

⇒ Endast tal inom ett begränsat talområde kan användas

⇒ Spill (”overflow”) kan uppstå vid en aritmetiska beräkning i de fall då dess resultat hamnar utanför talområdet

Systemet använder kodord som representerar antingen enbart positiva tal eller både positiva och negativa tal:

⇒ En lämplig kodning måste finnas för båda typer av tal

⇒ Spill måste kunna detekteras oavsett vilken typ av tal som ingår i en beräkning

Aritmetik i digitala system

Talområde:

Det talområde som gäller för ett digitalt system beror på om man beaktar tal utan tecken eller tal med tecken.

- För tal utan tecken är minsta möjliga talvärde = 0
- För tal med tecken beror minsta möjliga talvärde på antalet sifferpositioner samt vilken talrepresentation som används. Vi återkommer strax till detta.

Givet bas r och n sifferpositioner blir talområdet för tal utan tecken:

$$\left[0,\ r^n - 1\right]$$

Exempel: För decimala tal med 4 siffror blir talområdet $\left[0,\ 10^4 - 1\right] = \left[0,\ 9999\right]$

Exempel: För binära tal med 8 bitar blir talområdet $\left[0,\ 2^8 - 1\right] = \left[0,\ 255\right]$

Aritmetik i digitala system

Addition av tal utan tecken:

Räknereglerna för addition är likvärdiga oavsett vilket talsystem som används, och kan anpassas till digitala system med n sifferpositioner:

- Om addition av två tal utan tecken ger en minnessiffra (”carry out”) till sifferposition r^n har spill inträffat, och talområdets övre gräns har passerats.

Demo: decimal addition av talen $X = 7396_{10}$

$Y = 5842_{10}$

Demo: binär addition av talen $X = 10101101_2 = 173_{10}$

$Y = 01100110_2 = 102_{10}$

Aritmetik i digitala system

Subtraktion av tal utan tecken:

Räknereglerna för subtraktion är likvärdiga oavsett vilket talsystem som används, och kan anpassas till digitala system med n sifferpositioner:

- Om subtraktion av två tal utan tecken kräver ett lån (”borrow”) från sifferposition r^n har spill inträffat, och talområdets undre gräns har passerats. Negativa tal finns ju inte för tal utan tecken.

Demo: decimal subtraktion av talen $X = 7396_{10}$

$$Y = 5842_{10}$$

Demo: binär subtraktion av talen $X = 10101101_2 = 173_{10}$

$$Y = 01100110_2 = 102_{10}$$

Aritmetik i digitala system

Subtraktion av tal utan tecken:

Räknereglerna för subtraktion är likvärdiga oavsett vilket talsystem som används, och kan anpassas till digitala system med n sifferpositioner:

- Or ow") från sif räns har pa

Hur hanteras tal med tecken i aritmetiken?

Demo

För att förstå det måste vi definiera en lämplig form av teckenrepresentation.

Demo: binär subtraktion av talen

$$X = 10101101_2 = 173_{10}$$

$$Y = 01100110_2 = 102_{10}$$

Teckenrepresentation

Tecken-belopp-representation:

Om vi låter mest signifikant bit (position 2^{n-1}) i kodordet representera tecken, och de övriga bitarna representera talets belopp (absolutvärde) får vi ett symmetriskt talområde.

kodord	värde	
0 0 0	"0"	pos
0 0 1	"1"	
0 1 0	"2"	
0 1 1	"3"	
1 0 0	"0"	neg
1 0 1	"-1"	
1 1 0	"-2"	
1 1 1	"-3"	

Exempel: Med n = 3 bitar och tecken-belopp kan man representera talområdet [−3,+3].

Observera att talområdet nu förvisso blir symmetriskt (min = −3, max = +3), men att vi samtidigt har fått två nollor (+0 och -0) med olika kodord.

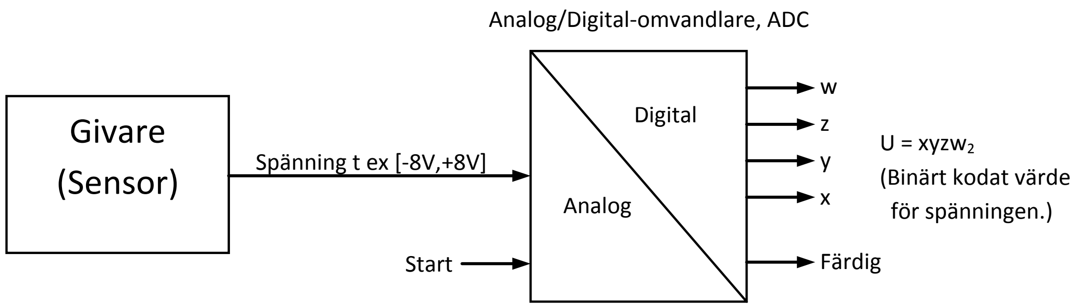
Denna representation är inte lämplig då

- vanlig binär addition inte ger rätt resultat
- den kräver olika beräkningsmetoder för olika kombinationer av positiva och negativa tal.

Teckenrepresentation

Excess-N-representation:

Vi såg tidigare att man med hjälp av en excess-N-kod kan representera både positiva och negativa tal.



Koden decimalt:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Koden binärt:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	U
Spänningsvärde:	-8V	-7V	-6V	-5V	-4V	-3V	-2V	-1V	0V	+1V	+2V	+3V	+4V	+5V	+6V	+7V	+8V

Man får här spänningen i volt genom att subtrahera 8 från kodordets värde (excess 8).

Teckenrepresentation

Excess-N-representation:

Kan denna representation, som har teckenbit och bara en nolla, leda till något bättre?

kodord	värde	
0 0 0	“-4”	
0 0 1	“-3”	
0 1 0	“-2”	neg
0 1 1	“-1”	
1 0 0	“0”	
1 0 1	“1”	
1 1 0	“2”	pos
1 1 1	“3”	

Exempel: Med n = 3 bitar och excess-4-kod kan man representera talområdet [−4,+3].

Observera att talområdet nu blir osymmetriskt (min = −4, max = +3), på grund av att talet ”0” betraktas som positivt.

Denna representation är inte heller helt lämplig då vanlig binär addition inte fungerar direkt, utan kräver att teckenbiten i resultatets kodord inverteras efter additionen för att det skall representera rätt värde.

Teckenrepresentation

Excess-N-representation:

Men vad händer om vi inverterar teckenbiten i excess-N-kod?

Invertera teckenbit

kodord	värde	
0 0 0	"-4"	
0 0 1	"-3"	neg
0 1 0	"-2"	
0 1 1	"-1"	
1 0 0	"0"	
1 0 1	"1"	pos
1 1 0	"2"	
1 1 1	"3"	

kodord	värde	
1 0 0	"-4"	
1 0 1	"-3"	neg
1 1 0	"-2"	
1 1 1	"-1"	
0 0 0	"0"	
0 0 1	"1"	pos
0 1 0	"2"	
0 1 1	"3"	

Sortera tabell

kodord	värde	
0 0 0	"0"	
0 0 1	"1"	pos
0 1 0	"2"	
0 1 1	"3"	
1 0 0	"-4"	
1 0 1	"-3"	neg
1 1 0	"-2"	
1 1 1	"-1"	

Teckenrepresentation

Excess-N-representation:

Men vad händer om vi inverterar teckenbiten i excess-N-kod?

Invertera teckenbit

kodord	värde	
0 0 0	"-4"	
0 0 1	"-3"	neg
[illegible]	[illegible]	
[illegible]	[illegible]	
1 0 0	"0"	
1 0 1	"1"	pos
1 1 0	"2"	
1 1 1	"3"	

kodord	värde	
1 0 0	"-4"	
1 0 1	"-3"	neg
[illegible]	[illegible]	
[illegible]	[illegible]	
0 0 0	"0"	
0 0 1	"1"	pos
0 1 0	"2"	
0 1 1	"3"	

Sortera tabell

kodord	värde	
0 0 0	"0"	
0 0 1	"1"	
0 1 0	"2"	pos
0 1 1	"3"	
1 0 0	"-4"	
1 0 1	"-3"	
1 1 0	"-2"	neg
1 1 1	"-1"	

Detta kallas för 2-komplementsrepresentation.

Teckenrepresentation

2-komplementsrepresentation:

Med denna representation fungerar vanlig binär addition med såväl positiva som negativa tal!

kodord	värde	
0 0 0	"0"	
0 0 1	"1"	↓ pos
0 1 0	"2"	
0 1 1	"3"	
1 0 0	"-4"	
1 0 1	"-3"	↑ neg
1 1 0	"-2"	
1 1 1	"-1"	

Man kan alltså addera en godtycklig kombination av positiva och negativa tal, med en och samma metod och utan att behöva göra efterjusteringar:

"1" + "2" = 001 + 010 = 011 = "3"

"-1" + "0" = 111 + 000 = 111 = "-1"

"-2" + "-1" = 110 + 111 = 101 = "-3"

"3" + "-3" = 011 + 101 = 000 = "0"

Teckenrepresentation

2-komplementsrepresentation:

Med denna representation fungerar vanlig binär addition med såväl positiva som negativa tal!

Man kan alltså addera en godtycklig kombination av positiva och negativa tal, med en och samma metod och utan att

kodord	värde	
0 0 0	“0”	
0 0 1	“1”	
0 1 0	“2”	
0 1 1	“3”	
1 0 0	“-4”	
1 0 1	“-3”	
1 1 0	“-2”	
1 1 1	“-1”	

Observera att det i dessa fall verkar som om spill i minnessiffran kan ignoreras så länge som kvarvarande bitar ger rätt resultat!

”-2” + ”-1” = 110 + 111 = 101 = ”-3”

”3” + ”-3” = 011 + 101 = 000 = ”0”

Vilken egenskap hos 2-komplementsrepresentationen gör att detta är möjligt?

2-komplementsrepresentation

Vad är ett 2-komplement?

2-komplementet till ett tal Y i det binära talsystemet är det tal (−Y) som vid addition av de två talens kodord ger resultatet 0.

Mer generellt gäller att r-komplementet till ett tal Y i ett talsystem med bas r är det tal (−Y) som vid addition av de två talens kodord ger resultatet 0.

Givet den bas r och det antal sifferpositioner n som ett digitalt system arbetar med definieras r-komplementet för ett tal Y på följande sätt:

$$Y_{r\text{-komplement}} = r^n - Y$$

För binära tal definieras alltså 2-komplementet för ett tal Y som:

$$Y_{2\text{-komplement}} = 2^n - Y$$

2-komplementsrepresentation

2-komplement i praktiken

Ett effektivt sätt att arbeta med 2-komplement är att observera följande:

$$Y_{\text{2-komplement}} = 2^n - Y = 2^n - 1 + 1 - Y = \underbrace{\boxed{(2^n - 1 - Y)}}_{\text{1-komplement}} + \underbrace{\boxed{1}}_{\text{"carry-in"}}$$

Uttrycket inom parentes kallas för <u>1-komplementet</u> av Y, och erbjuder möjligheten att ta komplementet på varje siffra var och en för sig.

Den återstående termen ’1’ använder man sedan som minnessiffra (”carry in”) i position 2^0 vid addition, tillsammans med 1-komplementet.

För binära tal betyder det att man byter ut varje bit y_i mot $1 - y_i$ (vilket ju innebär en logisk invertering av varje bit)

Exempel: binärt tal med 8 bitar $\quad Y_{\text{1-komplement}} = 2^8 - 1 - Y = 11111111_2 - Y$

För decimala tal kan man på samma sätt använda 9-komplementet.

Aritmetik i digitala system

Subtraktion av tal utan tecken:

Med 2-komplementaritmetik kan man utföra subtraktionen X – Y av två tal utan tecken genom att utföra additionen X + $Y_{2\text{-komplement}}$

- Om additionen av två tal utan tecken med 2-komplementaritmetik inte ger en minnessiffra till sifferposition 2^n har spill inträffat.

  Man inser detta genom att observera att 2-komplementet innehåller en term 2^n som skall finnas kvar i ett resultat utan spill.

$$X - Y = X + (-Y) = X + Y_{2\text{–komplement}} = X + 2^n - Y = 2^n + X - Y$$

Demo: binär subtraktion av talen

$$X = 10101101_2 = 173_{10}$$

$$Y = 01100110_2 = 102_{10}$$

Aritmetik i digitala system

Talområde för tal med tecken:

Givet bas r och n sifferpositioner blir talområdet för tal med tecken i r-komplementrepresentation:

$$\left[-\left(\frac{r^n}{2}\right), \left(\frac{r^n}{2}\right)-1\right]$$

Exempel: För decimala tal med 4 siffror blir talområdet

$$\left[-\left(\frac{10^4}{2}\right), \left(\frac{10^4}{2}\right)-1\right]=\left[-5000,\ 4999\right]$$

Exempel: För binära tal med 8 bitar blir talområdet

$$\left[-\left(\frac{2^8}{2}\right), \left(\frac{2^8}{2}\right)-1\right]=\left[-128,\ 127\right]$$

Aritmetik i digitala system

Addition och subtraktion av tal med tecken:

Aritmetik för tal med tecken kan anpassas till binära digitala system med n bitars kodord på följande sätt:

- Kodorden har 2-komplementrepresentation.
- Addition X + Y utförs direkt på kodorden.
- Subtraktion X – Y utförs medelst additionen $X + Y_{2\text{-komplement}}$.
- Detektering av spill vid addition eller subtraktion av två tal med tecken kan inte ske enbart via minnessiffran i position 2^n, utan kräver ytterligare analys av talens och resultatets egenskaper. Varför det blir så kommer vi att visa litet senare.

Aritmetik i digitala system

Grindnät för addition:

Vid manuell addition S = X + Y av två binära heltal X och Y med n bitar kan vi se att beräkningen kan betraktas som en serie av additioner av enskilda bitar, med början i position 2^0 och med slut i position 2^{n-1}:

- En bitaddition i position 2^i tar bit x_i från X och bit y_i från Y, samt en inkommande minnessiffra (”carry in”) c_i
- Resultatet från en bitaddition i position 2^i är dels en bit s_i i slutresultatet och dels en utgående minnessiffra (”carry out”) c_{i+1}

	c_n	c_{n-1}		c_{i+1}	c_i		c_1	c_0
		c_{n-1}	…	…	x_i	…	x_1	x_0
+		y_{n-1}	…	…	y_i	…	y_1	y_0
		s_{n-1}	…	…	s_i	…	s_1	s_0

Aritmetik i digitala system

Grindnät för addition:

Man bör alltså kunna bygga en komponent (ett grindnät) för addition av en enskild bit, och sedan seriekoppla (kaskadkoppla) n stycken sådana komponenter för att erhålla en n-bitars adderare.

- Ett grindnät för bitaddition som inte tar hänsyn till inkommande minnessiffra kallas för en halvadderare ("half adder").
- Ett grindnät för bitaddition som tar hänsyn till inkommande minnessiffra kallas för en heladderare ("full adder").

	c_n	c_{n-1}		$\boxed{c_{i+1}}$	c_i		c_1	c_0
		c_{n-1}	…	…	x_i	…	x_1	x_0
+		y_{n-1}	…	…	y_i	…	y_1	y_0
		s_{n-1}	…	…	$\boxed{s_i}$	…	s_1	s_0

Aritmetik i digitala system

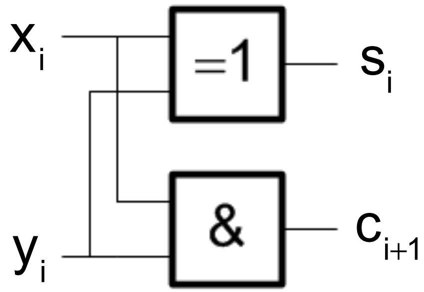
Grindnät för addition:

Härledning av grindnät för en halvadderare:

x_i	y_i	s_i	c_{i+1}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$s_i = x_i \oplus y_i$$

$$c_{i+1} = x_i * y_i$$



AND-funktion

XOR-funktion