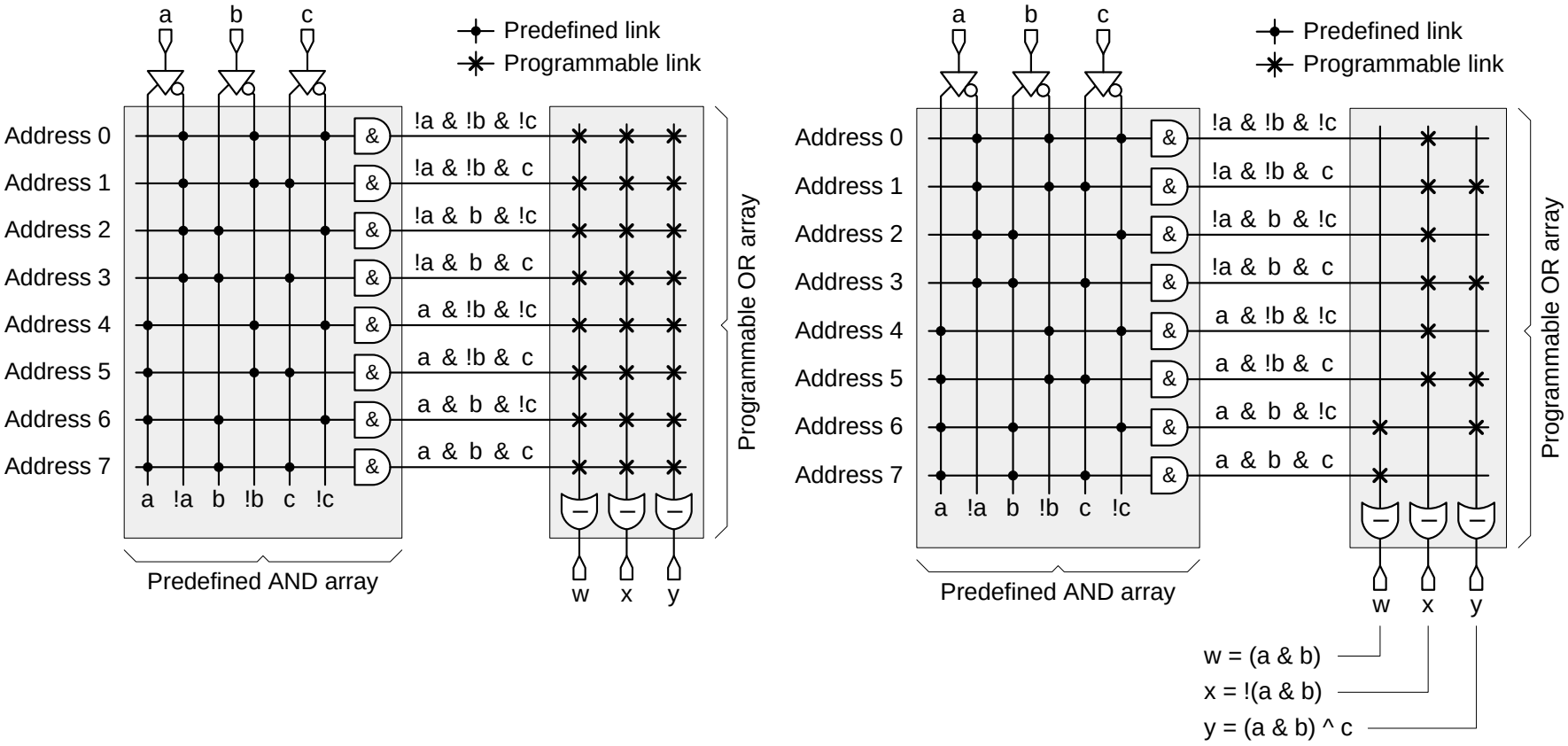
INF3430

Kretsteknologier

Programmeringsteknologier

VHDL-Access datatyper

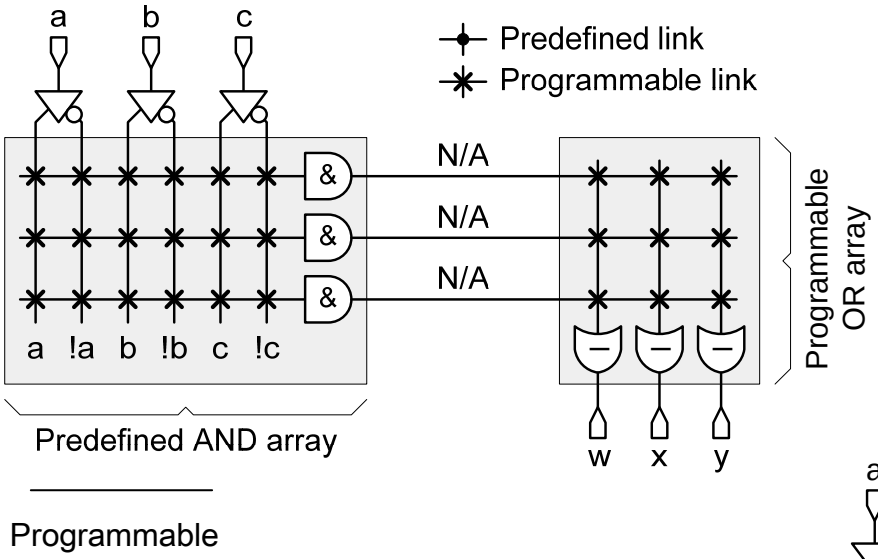
Programmable Read Only Memory

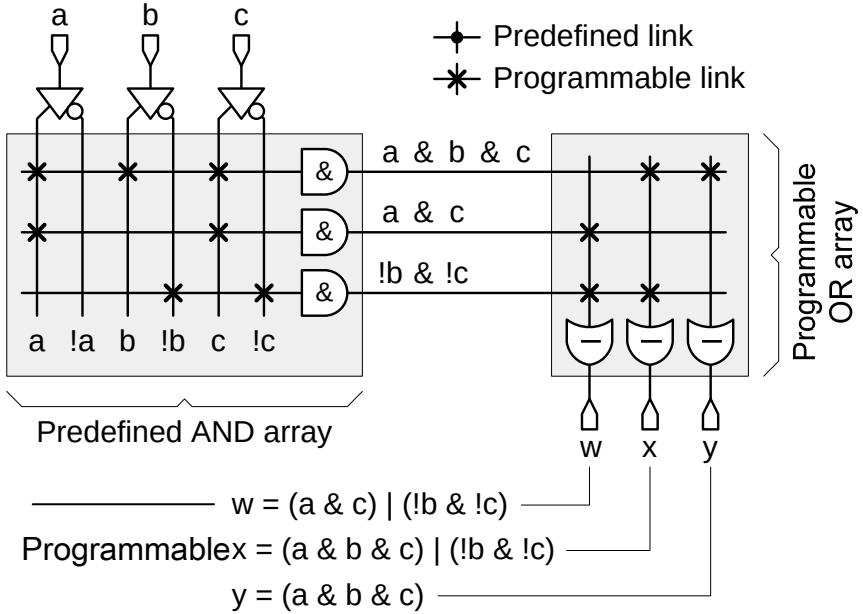


Programmable Logic Array (PLA)

- Gir ikke full dekoding (har færre enn 2^n mintermer).
- Antall produktledd må minimaliseres, men ikke antall input variable.

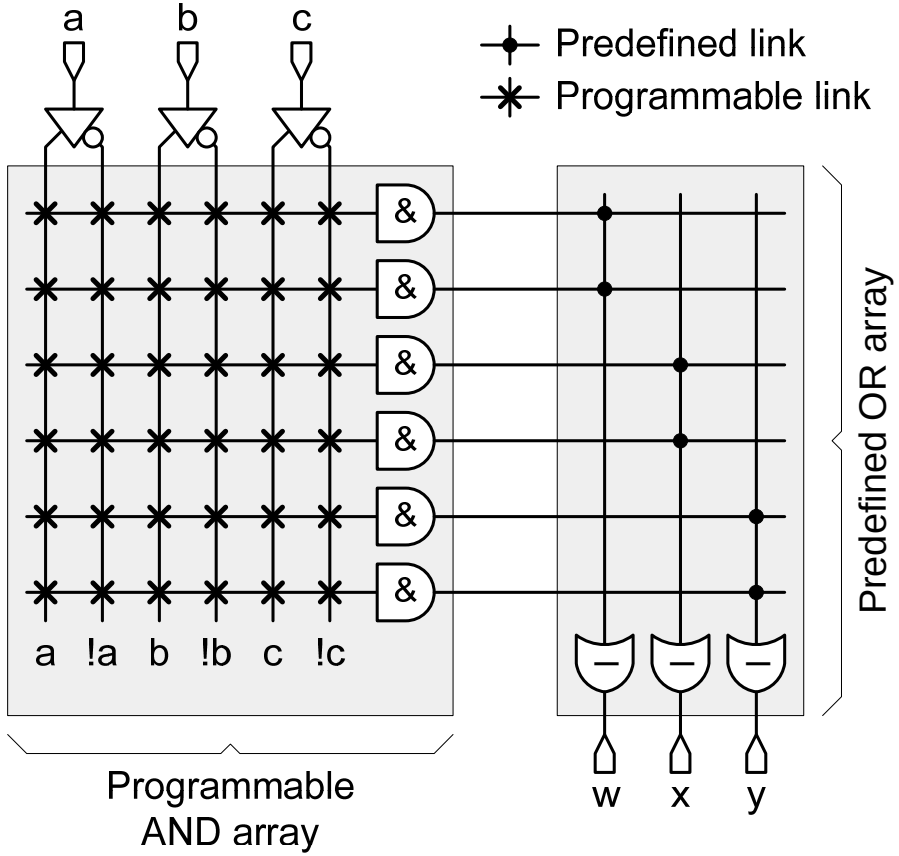
PLA eksempel





Programmable Array Logic (PAL)

- Som PLA, men med fast OR array
- Kan ikke dele produktledd mellom flere OR
- Mindre programmering (kun AND-array)
- Mindre fleksibilitet
- Re-programmerbare utgaver finnes (GAL)



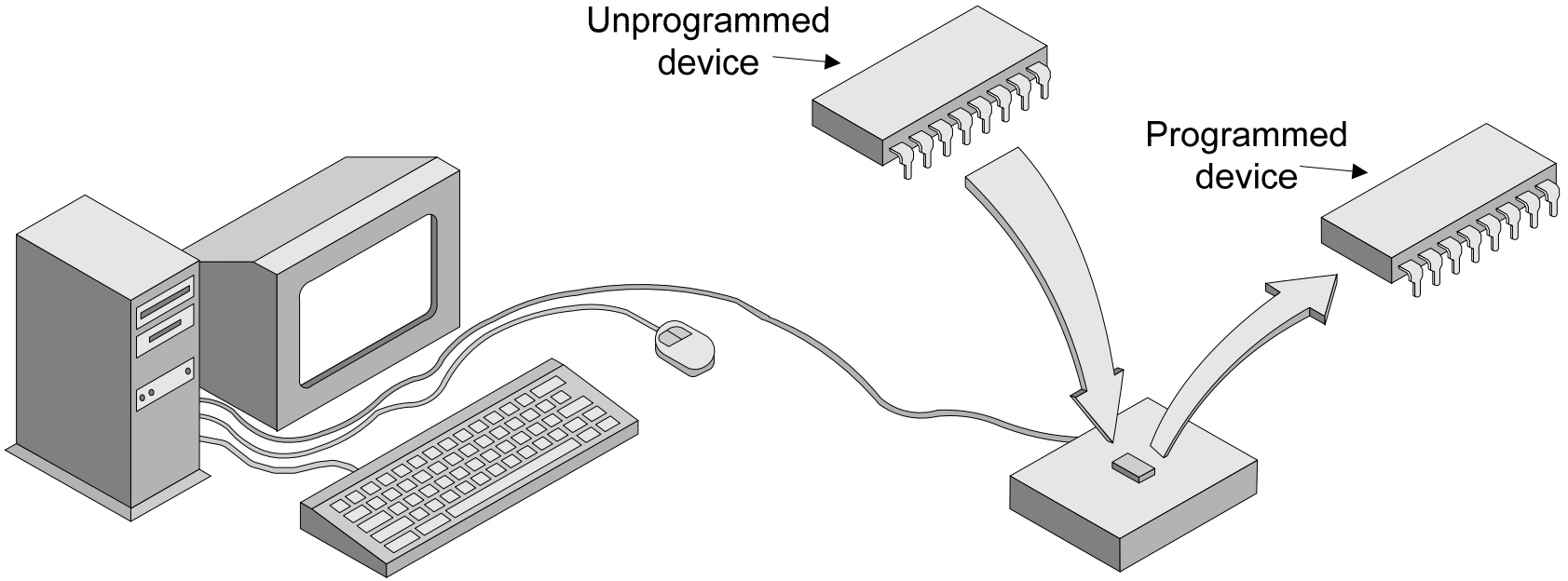
Programmerbar logikk arkitekturer

- Krets med inn/ut-linjer, logikk, vipper og programmerbare forbindelseslinjer.
- Familier:
  - SPLD - Simple Programmable Logic Device
  - CPLD - Complex Programmable Logic Device
  - FPGA - Field Programmable Gate Array
  - CSoC - Configurable System-on-Chip

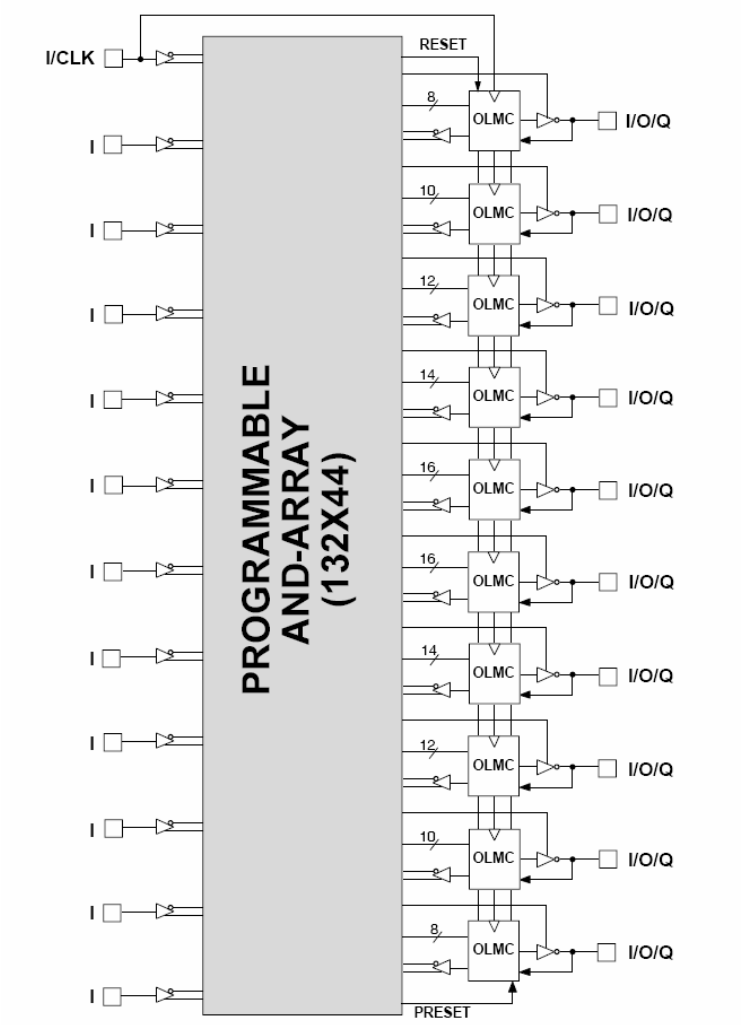
SPLD - Simple Programmable Logic Device

- Kjennetegn:
  - Minste og billigste typen av programmerbar logikk.
  - Kort pin-to-pin delay
- Andre betegnelser:
  - PROM (Programmable Read Only Memory)
  - PLA (Programmable Logic Array)
  - PAL (Programmable Array Logic, Vantis)
  - GAL (Generic Array Logic, Lattice, Lattice)
- Programmering:
  - Fuses eller ikke-flyktig minne som EPROM, EEPROM eller FLASH.

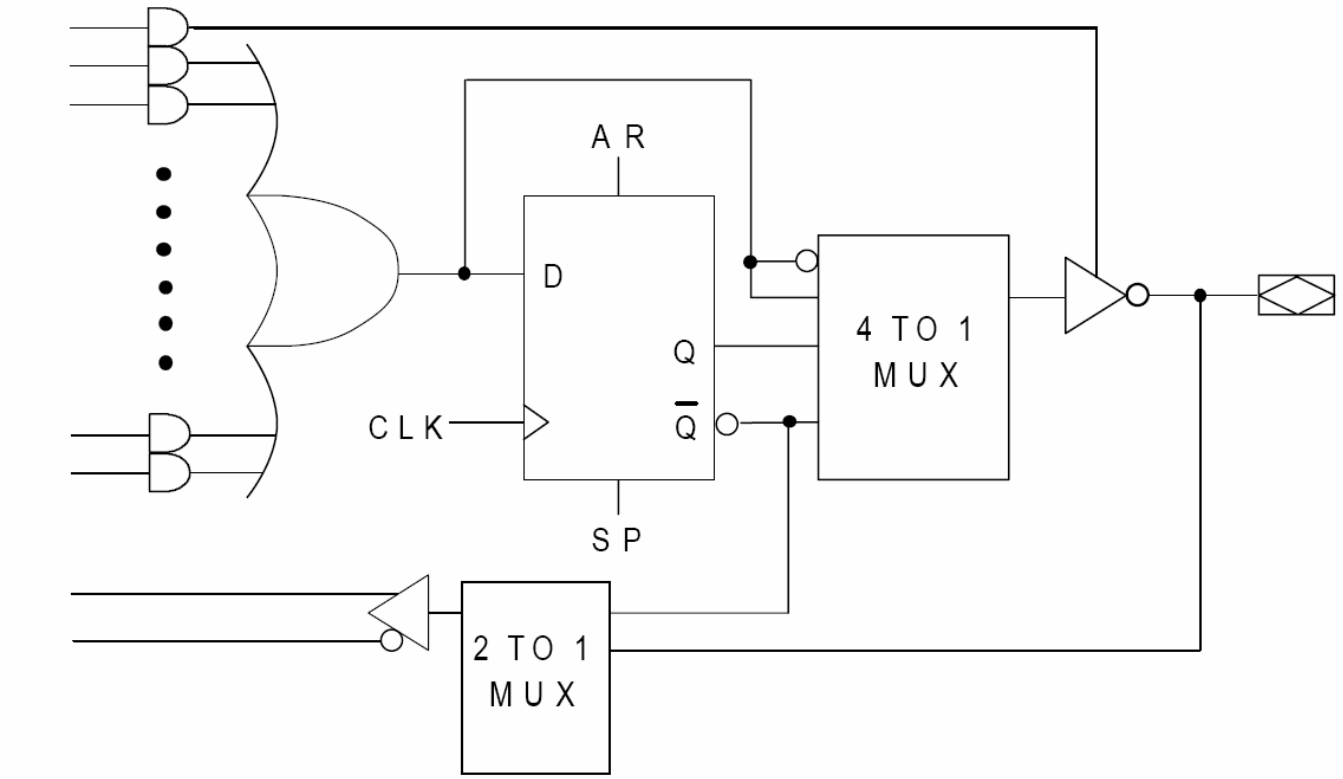
PLD programmerer



Eksempel: GAL22V10 (Lattice)



Eksempel: GAL22V10 (Lattice)

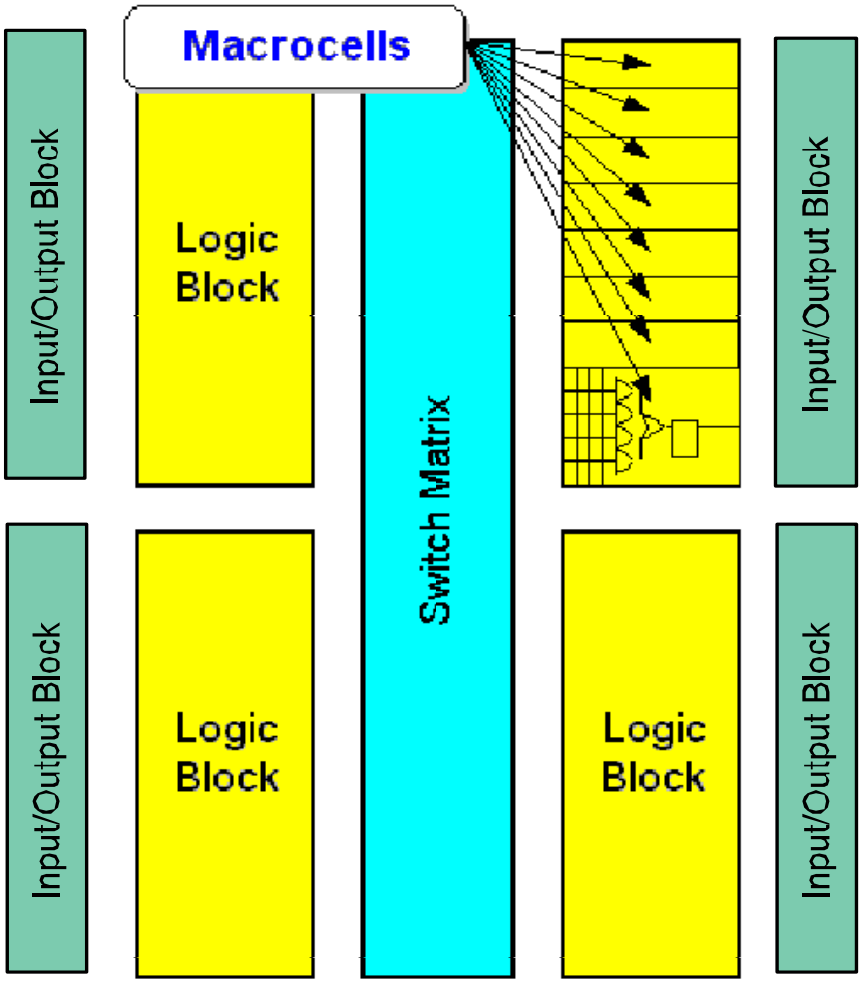


GAL22V10 OUTPUT LOGIC MACROCELL (OLMC)

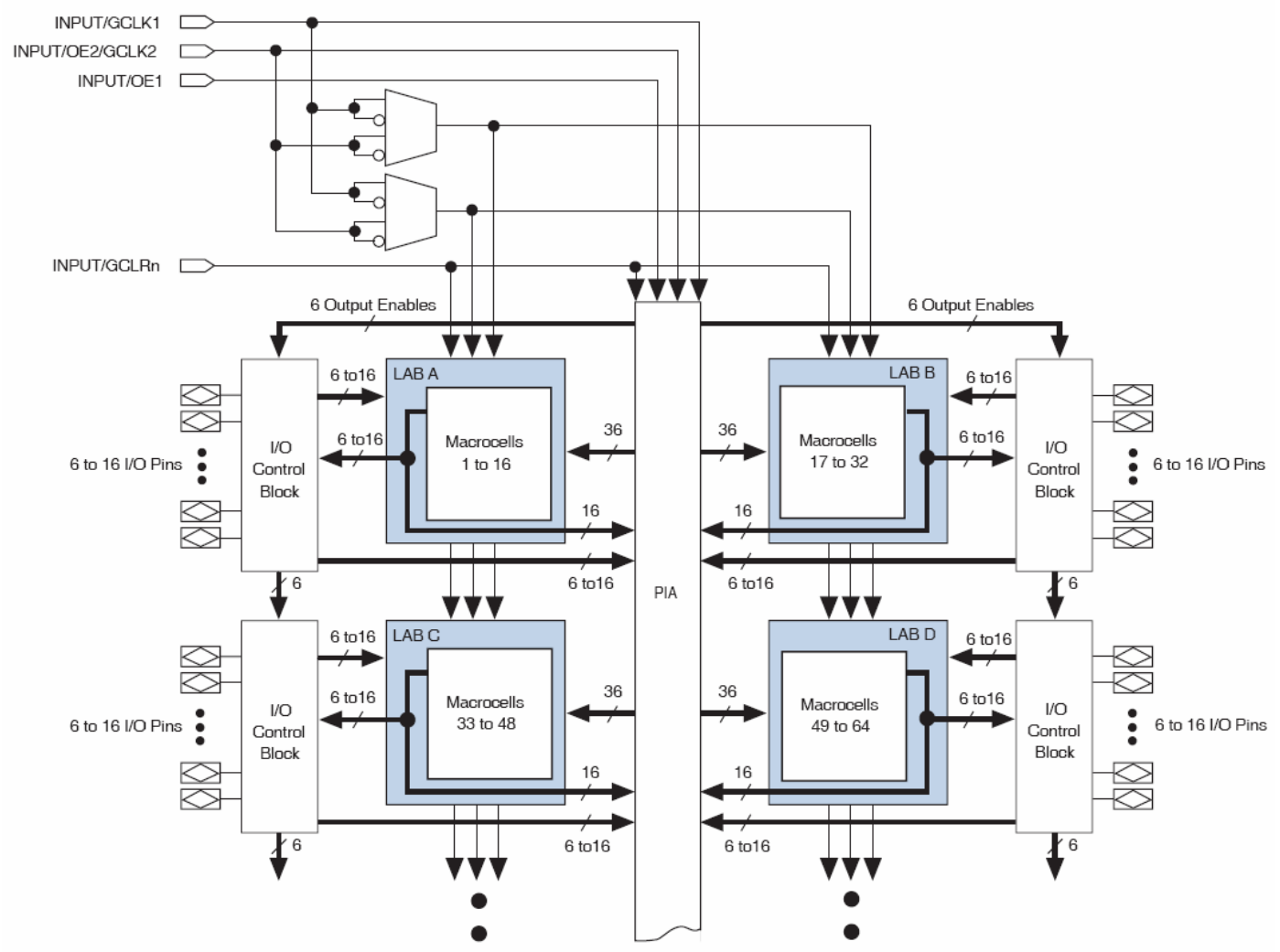
CPLD - Complex Programmable Logic Device

- Kjennetegn:
  - En typisk CPLD har 2 til 64 ganger så mye logikk som en SPLD.
  - Kort pin-to-pin delay
- Andre betegnelser:
  - EPLD (Erasable Programmable Logic Device)
  - PEEL
  - EEPLD (Electrically-Erasable Programmable Logic Device)
  - MAX (Multiple Array matriX, Altera)
- Programmering:
  - Ikke-flyktig minne som EPROM, EEPROM eller FLASH (også SRAM er nå tilgjengelig).
  - Vanlig nå med ISR (In-System Re-programable) gjennom JTAG interface.

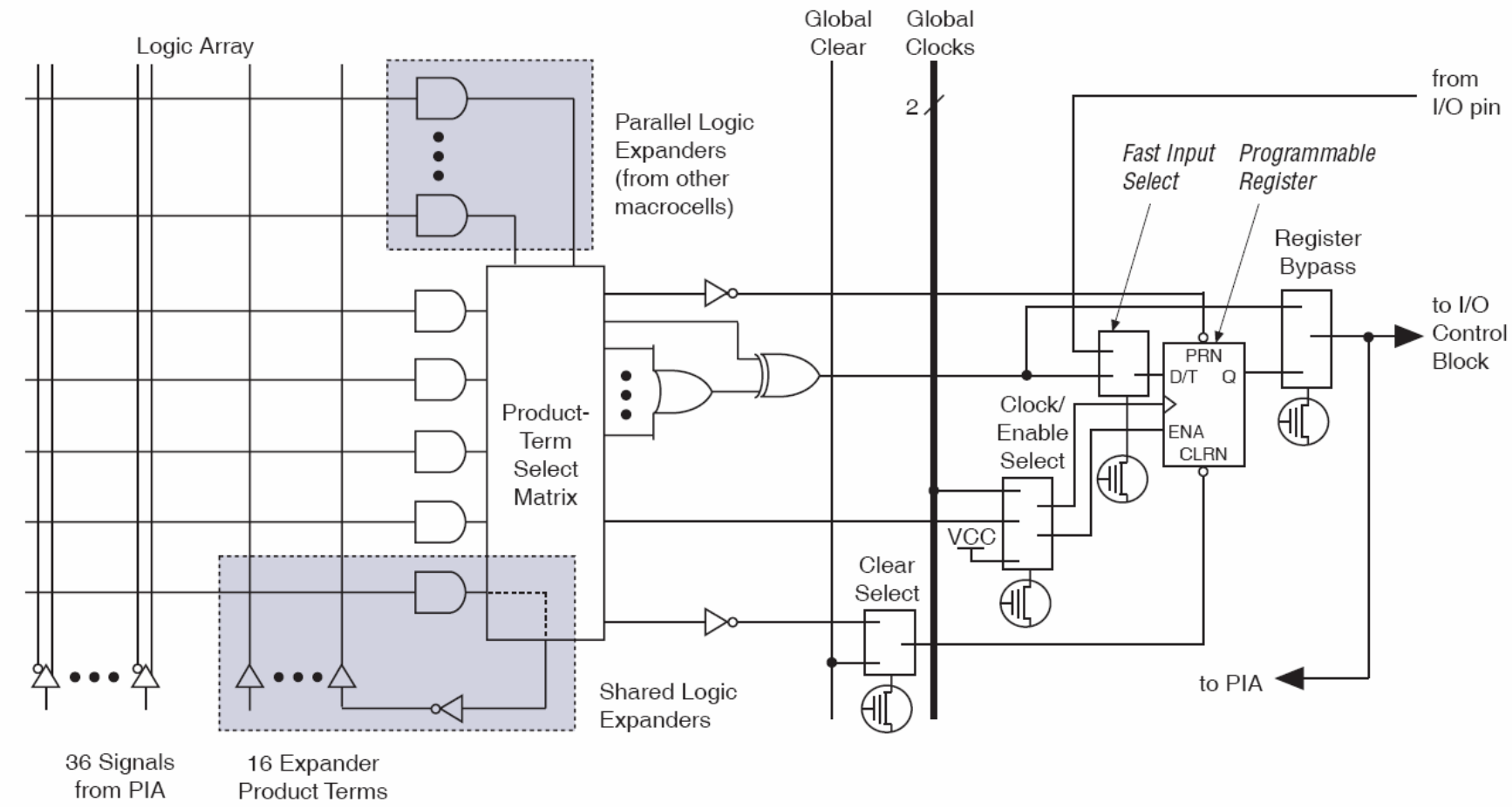
CPLD


Macrocells
Input/Output Block
Logic Block
Switch Matrix
Input/Output Block
Input/Output Block
Logic Block
Logic Block
Input/Output Block

Eksempel-Max7000S (Altera)



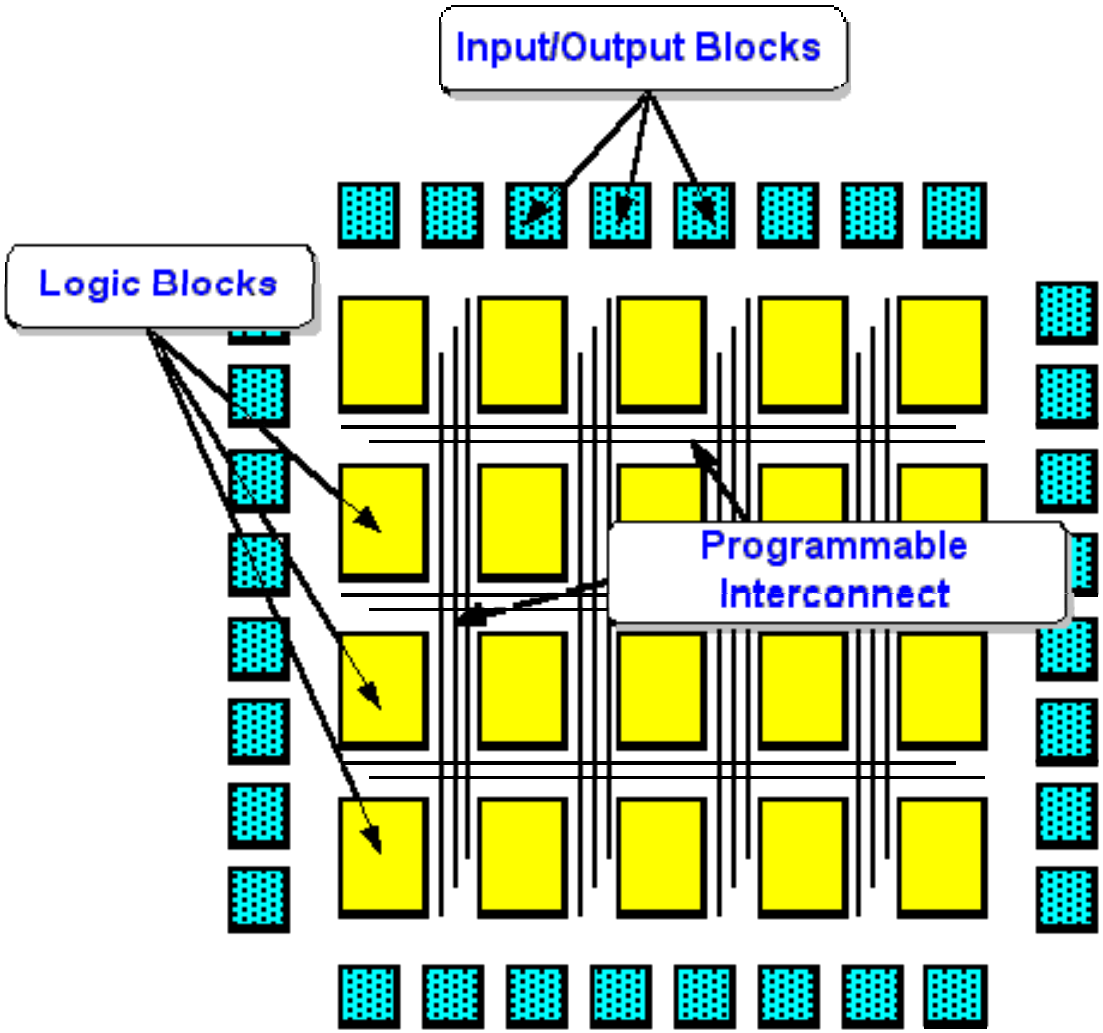
Eksempel-Max7000 (Altera)



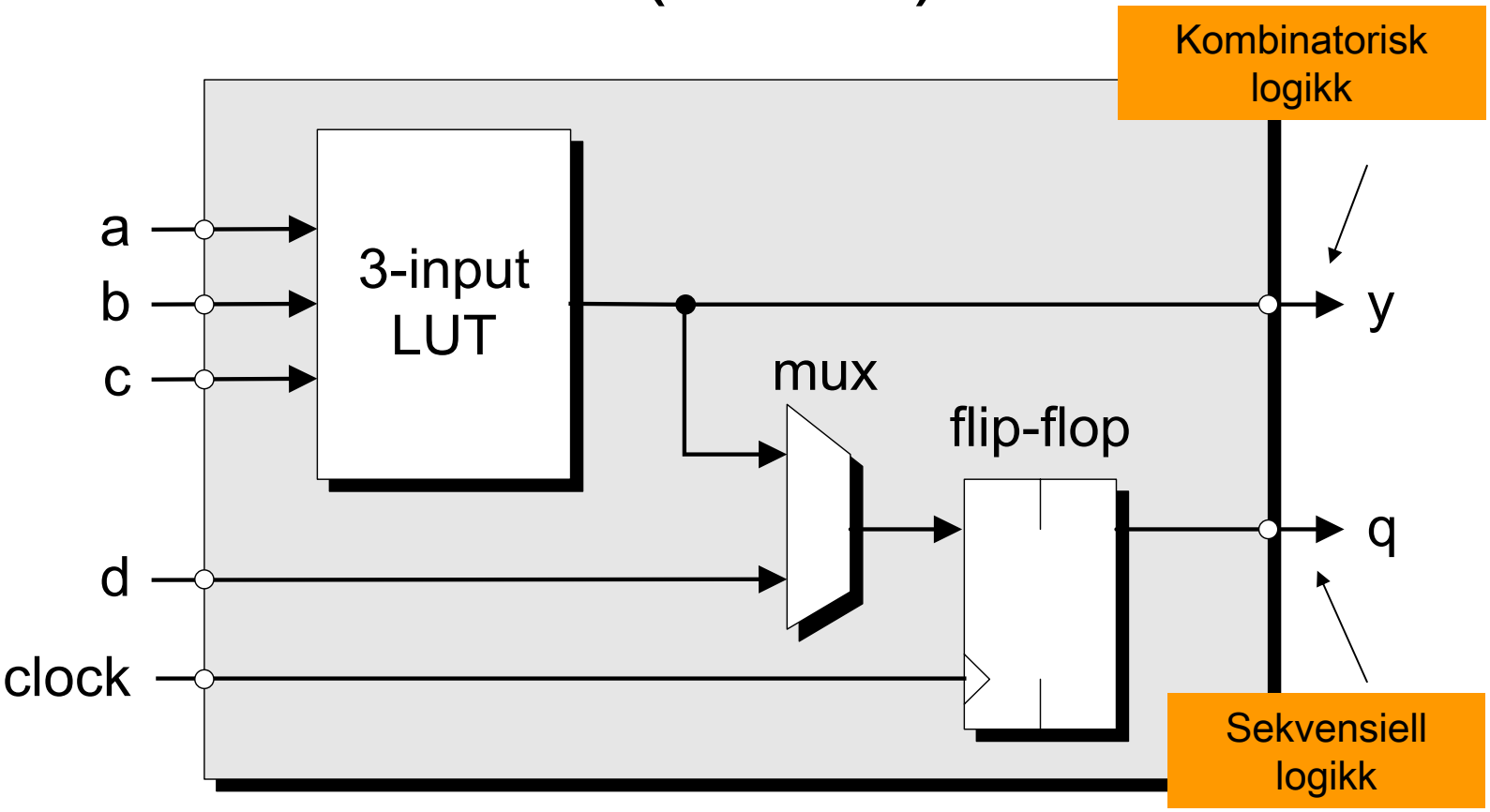
FPGA - Field Programmable Gate Array

- Kjennetegn: Tilbyr typisk en større mengde logikk på en krets enn det CPLD gjør.
- Andre betegnelser:
  - LCA (Logic Cell Array)
  - pASIC (programmable ASIC), Cypress
  - FLEX, APEX (Altera)
  - ACT (Actel)
  - ORCA (Lucent)
  - Virtex (Xilinx)
  - pASIC (QuickLogic)
- Programmering: Statisk minne (SRAM) eller anti-fuse teknologi (også noen med EEPROM eller FLASH finnes).

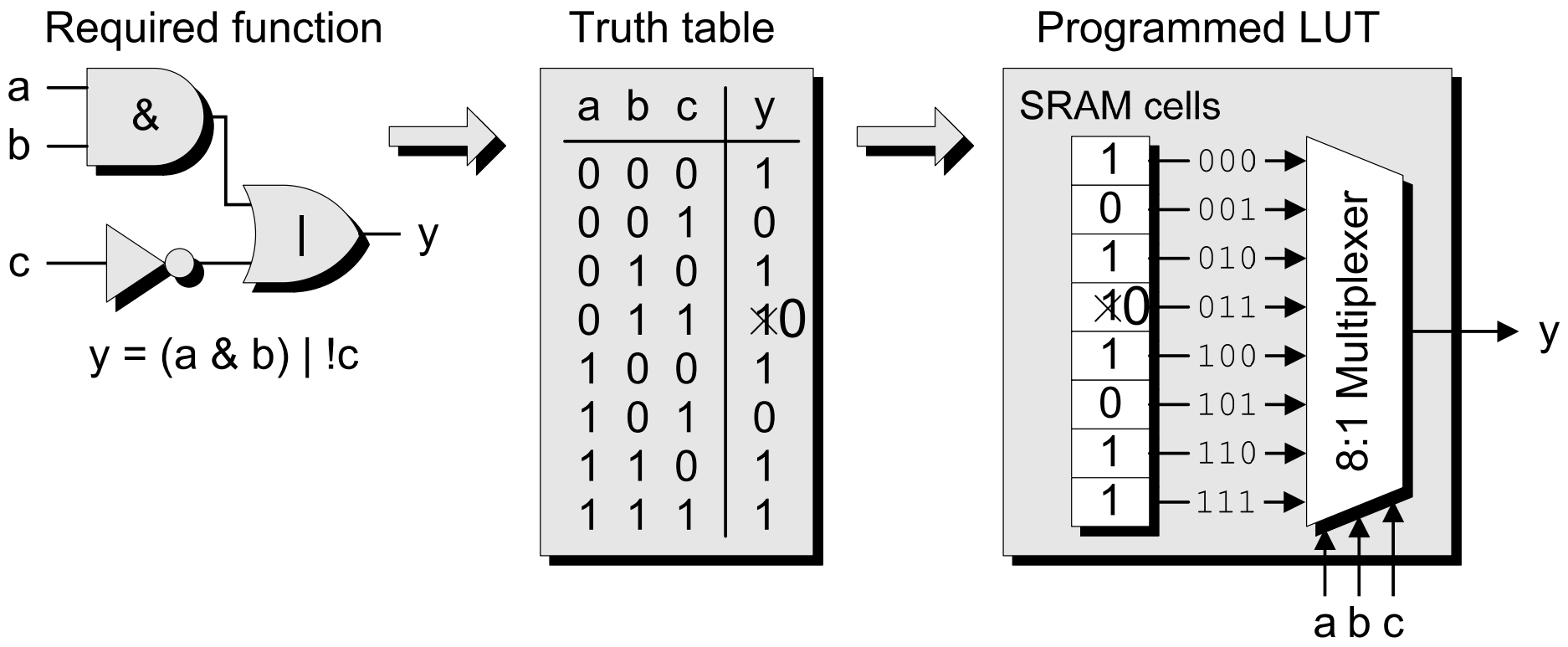
FPGA


Input/Output Blocks
Logic Blocks
Programmable
Interconnect

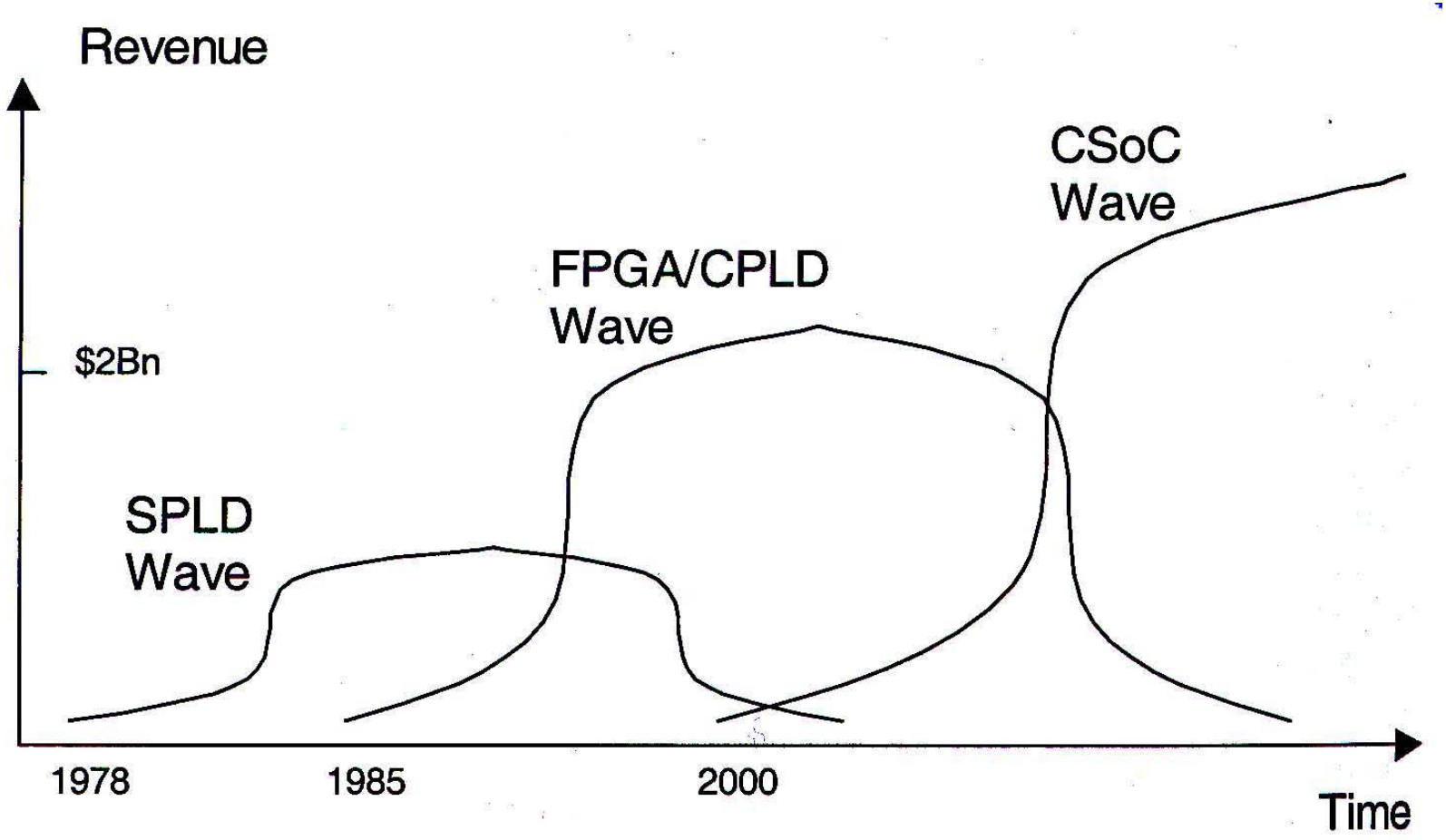
Eksempel på logikk blokk/celle (FPGA)



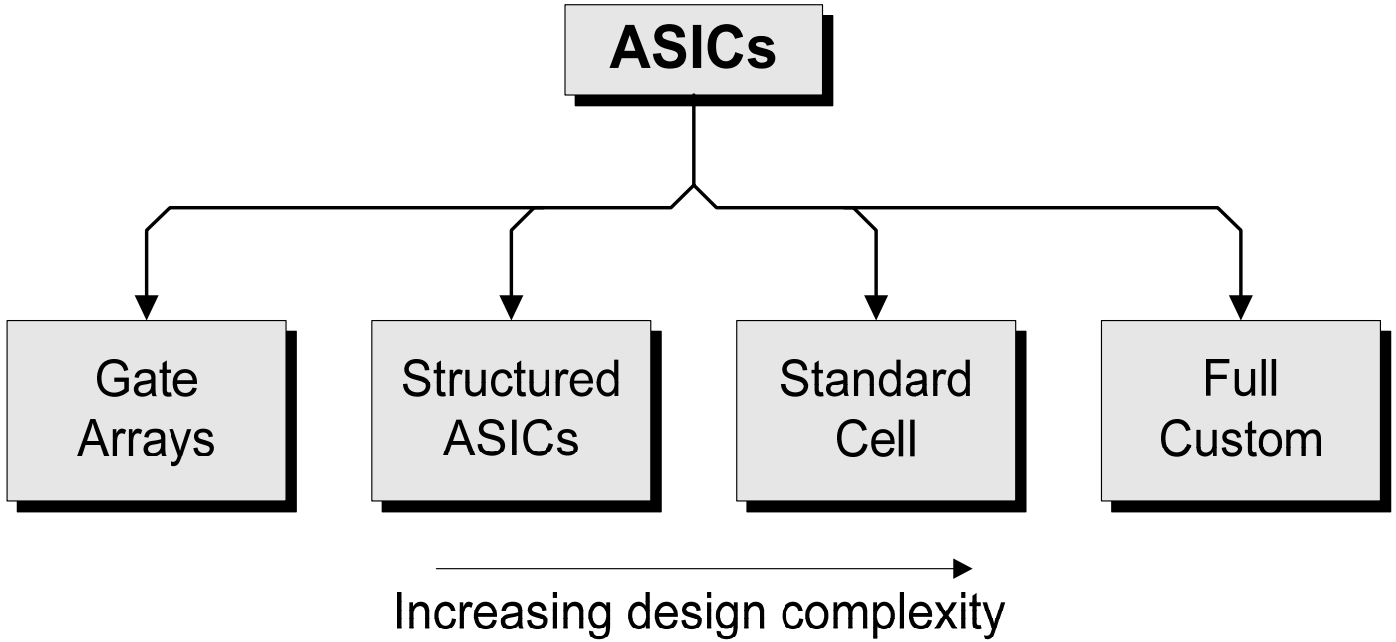
Look-Up Table (LUT)



Utvikling programmerbare kretser



ASIC: Application Specific Integrated Circuit



ASIC: Brukerutviklet IC

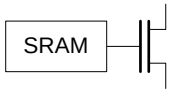
Når bør en bruke FPGA (CPLD)?

- Status: Førstevalg for digital logikk design med unntak av:
  - Enkle eller tidskritiske design (ASIC/PAL/diskret logikk bedre)
  - Produkt skal produseres i stort antall (ASIC bedre)
  - Veldig komplekse design (ASIC)
  - Design der minimalisering av effektforbruk er kritisk (mobile applikasjoner)
- ASIC vs FPGA (2003)
  - 1500 – 4000 nye ASIC prosjekter hvert år
  - 450 000 nye FPGA prosjekter hvert år

Fordeler med FPGA vs. ASIC

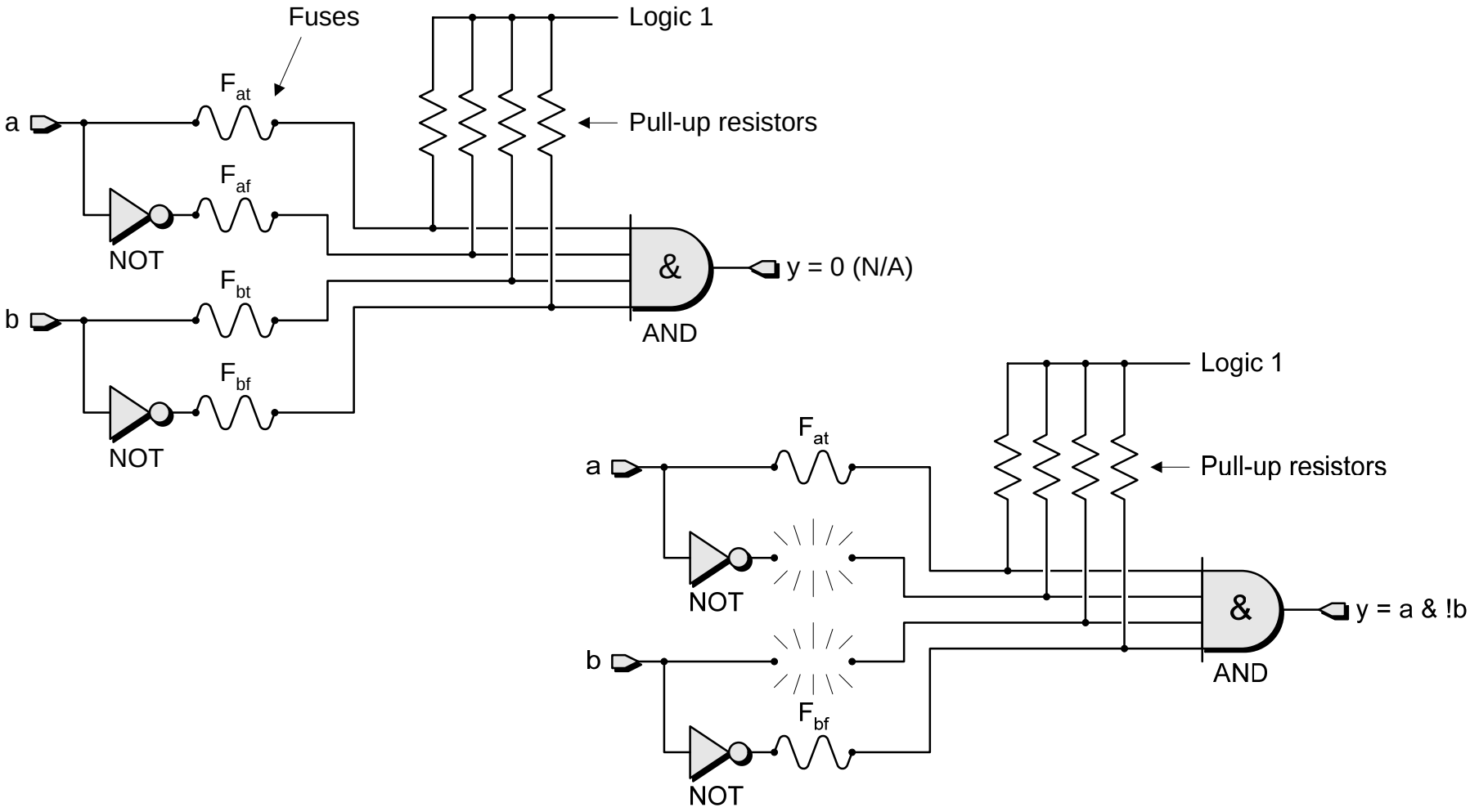
- Kortere utviklingstid på grunn av enkel re-programmering.
- Kan re-programmeres ”i system” ute i felt.
- Mindre økonomisk risiko (produksjon av en ASIC-krets er dyrt og den kan ikke re-programmeres).

Programmeringsteknologier for programmerbar logikk (Kap 2 og 4)

Technology	Symbol	Predominantly associated with ...
Fusible-link		SPLDs
Antifuse		FPGAs
EPROM		SPLDs and CPLDs
E^2PROM/ FLASH		SPLDs and CPLDs (some FPGAs)
SRAM		FPGAs (some CPLDs)

- Alle teknikker trenger ekstra utstyr
- Varierer i overhead og kompleksitet

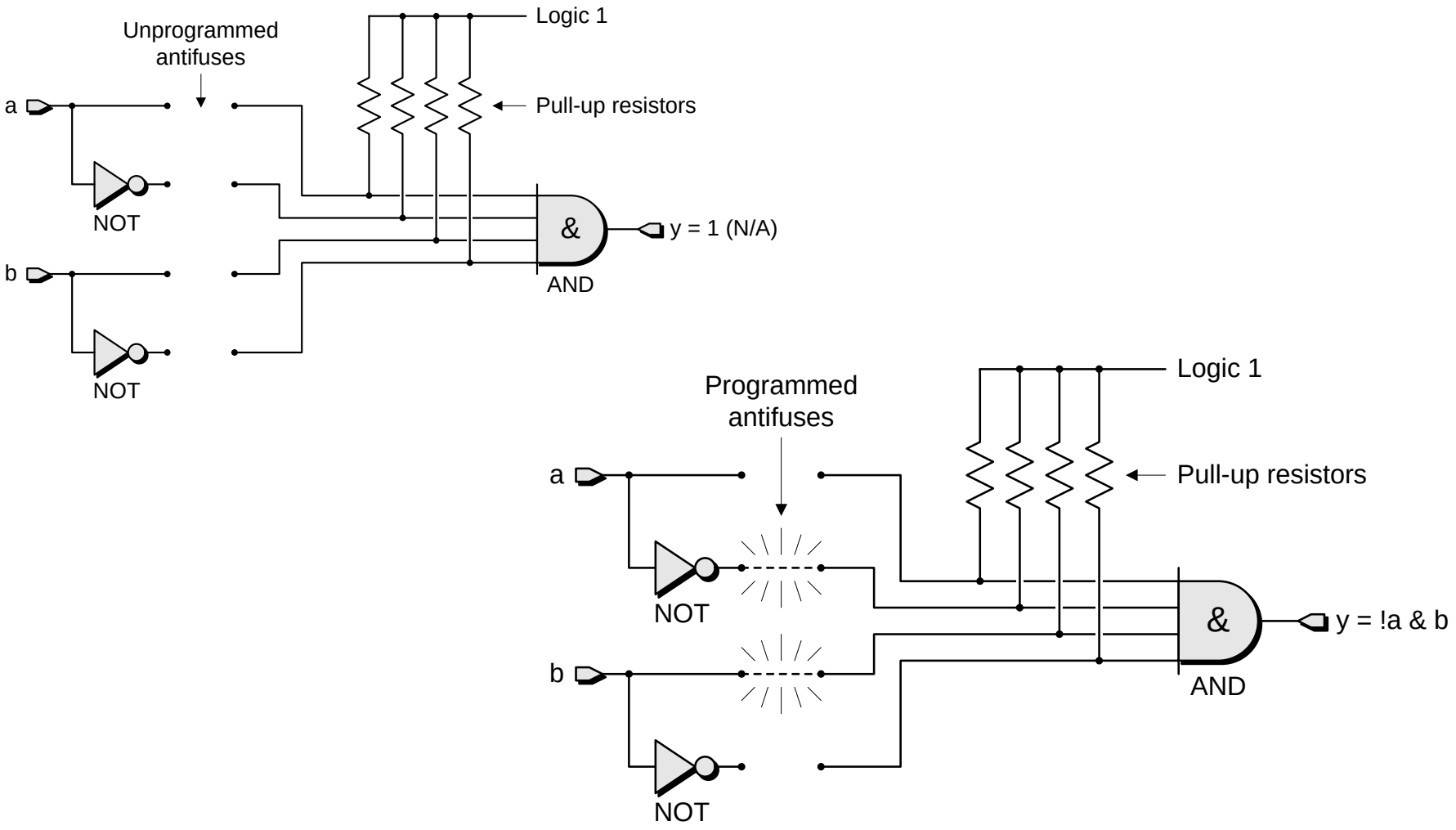
Fusable link (sikringer)



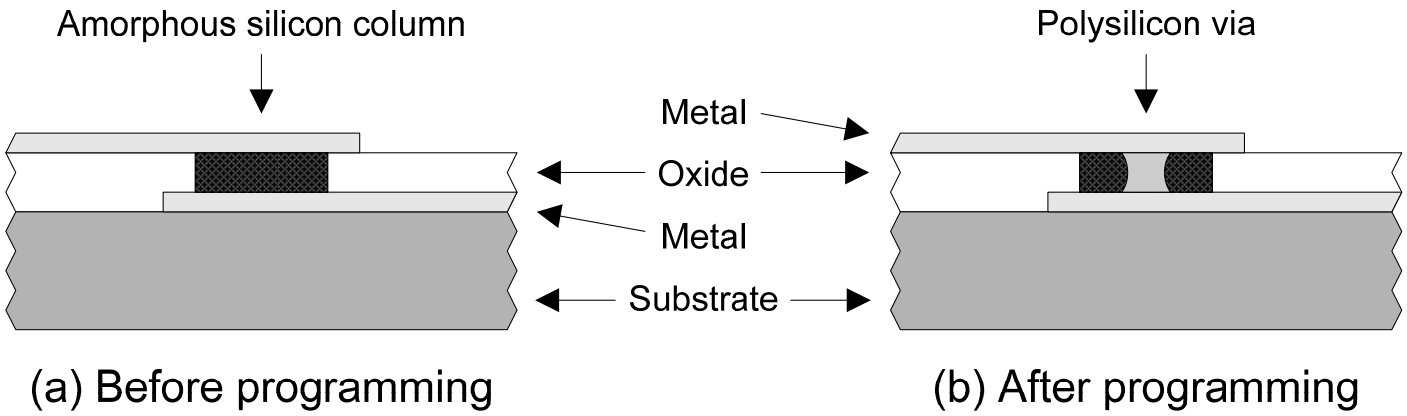
Antifuse

- Prinsipp:
  - Konfigurasjon lagres i FPGA ved at det lages kortslutninger ved bruk av høy spenning.
- Fordeler
  - Lav impedans når sikring er ’on’ (liten forsinkelse)
  - Lavt strømforbruk
  - Kompakt teknologi (tar lite plass)
  - Ekstra pålitelig teknologi (relativt strålingsimmune)
- Ulemper
  - Må programmeres i en egen programmeringsenhet
  - Høy programmeringsspenning og -strøm
  - Permanent programmering (kan kun programmeres en gang)

Antifuse

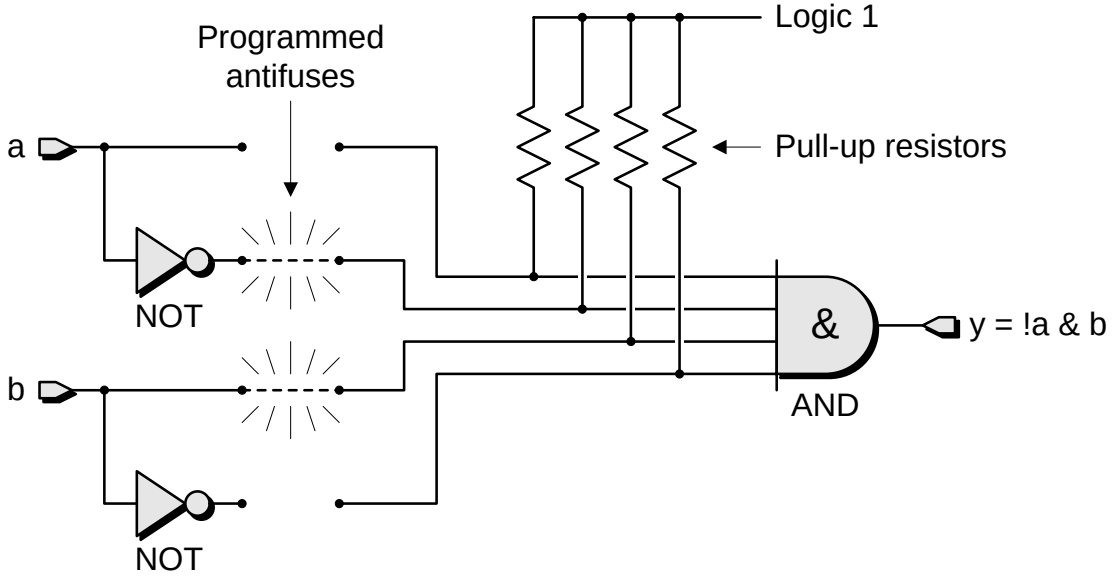

Unprogrammed
antifuses
Logic 1
Pull-up resistors
a
NOT
b
NOT
&
AND
y = 1 (N/A)
Programmed
antifuses
Logic 1
Pull-up resistors
a
NOT
b
NOT
&
AND
y = !a & b

Antifuse



(a) Before programming

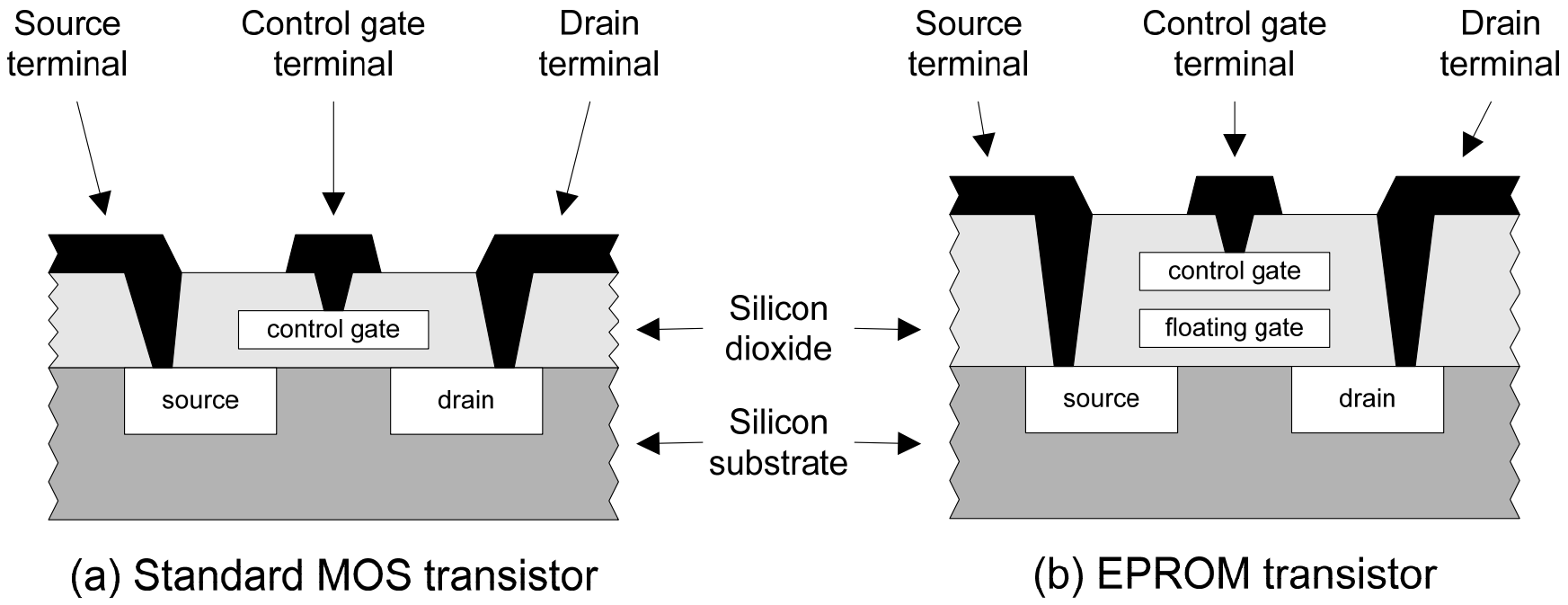
(b) After programming



Floating-gate ((E)EPROM/FLASH)

- Prinsipp:
  - Konfigurasjon lagres i krets i form av offset på gate-inngangene til pass-transistorer.
- 3 typer programmering/sletting:
  - UV-lys (EPROM)
  - Høy spenning (EEPROM)
  - Logisk spenning (FLASH)
- Fordeler
  - Kan reprogrammeres
  - Permanent, men re-programerbar
  - Mer kompakt teknologi enn SRAM
- Ulemper
  - Tidkrevende programmering

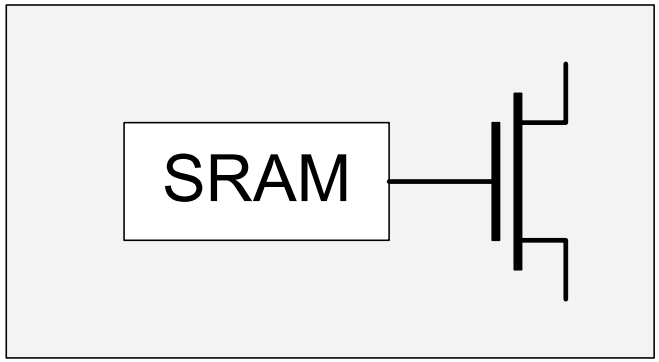
EPROM



(a) Standard MOS transistor

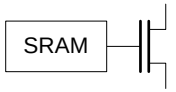
(b) EPROM transistor

SRAM



- Prinsipp:
  - SRAM-minne inne i FPGA kretsens konfigurasjon
- Fordeler
  - Kan programmeres uendeling mange ganger
  - Kan lett deles mellom flere design
  - Trenger ikke spesiell prosess
- Ulemper
  - Plassoverhead (SRAM-celle med 5 transistorer)
  - Flyktig minne (må lagre konfigurasjonen i eksternt permanent minne)

Programmeringsteknologier for programmerbar logikk

Technology	Symbol	Predominantly associated with ...
Fusible-link		
Antifuse		
EPROM		
E^2PROM/ FLASH		
SRAM		

- Hvilke prog. logikk familier anvender typisk teknologiene?

Programmeringsteknologier

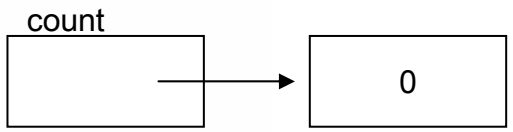
Feature	SRAM	Antifuse	E2PROM / FLASH
Technology node	State-of-the-art	One or more generations behind	One or more generations behind
Reprogrammable	Yes (in system)	No	Yes (in-system or offline)
Reprogramming speed (inc. erasing)	Fast	----	3x slower than SRAM
Volatile (must be programmed on power-up)	Yes	No	No (but can be if required)
Requires external configuration file	Yes	No	No
Good for prototyping	Yes (very good)	No	Yes (reasonable)
Instant-on	No	Yes	Yes
IP Security	Acceptable (especially when using bitstream encryption)	Very Good	Very Good
Size of configuration cell	Large (six transistors)	Very small	Medium-small (two transistors)
Power consumption	Medium	Low	Medium
Rad Hard	No	Yes	Not really

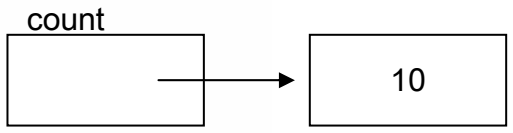
Access datatyper

- Benyttes der størrelsen av data ikke er kjent på forhånd
  - Dynamisk allokering av data
  - Definerer pekere til data
- Benyttes for å lage en kompleks sammenheng mellom datatyper
  - Scalare og composite datatyper ikke tilstrekkelig
  - F.eks. lenka lister
- Benyttes i modellering/testbenker

Deklarering og allokering

```
process is

   --Deklarasjon av pekertype til datatypen natural
   type natural_ptr is access natural;

   --Deklarasjon av peker
   variable count : natural_ptr;

begin
   --allokerer et nytt natural object og
   --count settes til å peke på det
   count := new natural;

   --tilordning av verdi til objectet
   count.all := 10;

   --allokering, initialisering av peker
   --og tilodning av verdi
   count := new natural'(10);
   wait;
end process;
```





Bruk av record datatyper

```
process is
  --Deklarasjon av record datatype
  type stimulus_record is record
      stimulus_time : time;
      stimulus_value : bit_vector(0 to 3);
    end record stimulus_record;

  --Deklarasjon av pekertype til stimulus_record
  type stimulus_ptr is access stimulus_record;

  --Deklarasjon av peker til stimulus_record
  variable bus_stimulus : stimulus_ptr;
begin

  --Allokering av nytt stimulus_record object
  --tilordning av peker og verdi
  bus_stimulus := new stimulus_record'( 20 ns, B"0011" );

  --Tilordning av ny verdi
  bus_stimulus.all := (20 ns, B"0010");
  --Tilordning av enkeltelement
  bus_stimulus.stimulus_time := 10 ns;
  wait;
end process;
```

Bruk array datatyper

```
process is

  type time_array is array (positive range <>) of time;
  type time_array_ptr is access time_array;
  variable activation_times : time_array_ptr;

begin

  --Allokerer et nytt time_array object på tre elementer
  activation_times := new time_array'(10 us, 15 us, 40 us);
  --Allokerer et nytt timearray object på to object i tillegg til det
  --eksisterer fra før
  activation_times := new time_array'( activation_times.all
                                       & time_array'(70 us, 100 us) );
  --Allokerer nytt time object med 10 elementer
  activation_times := new time_array(1 to 10);

  wait;
end process;
```

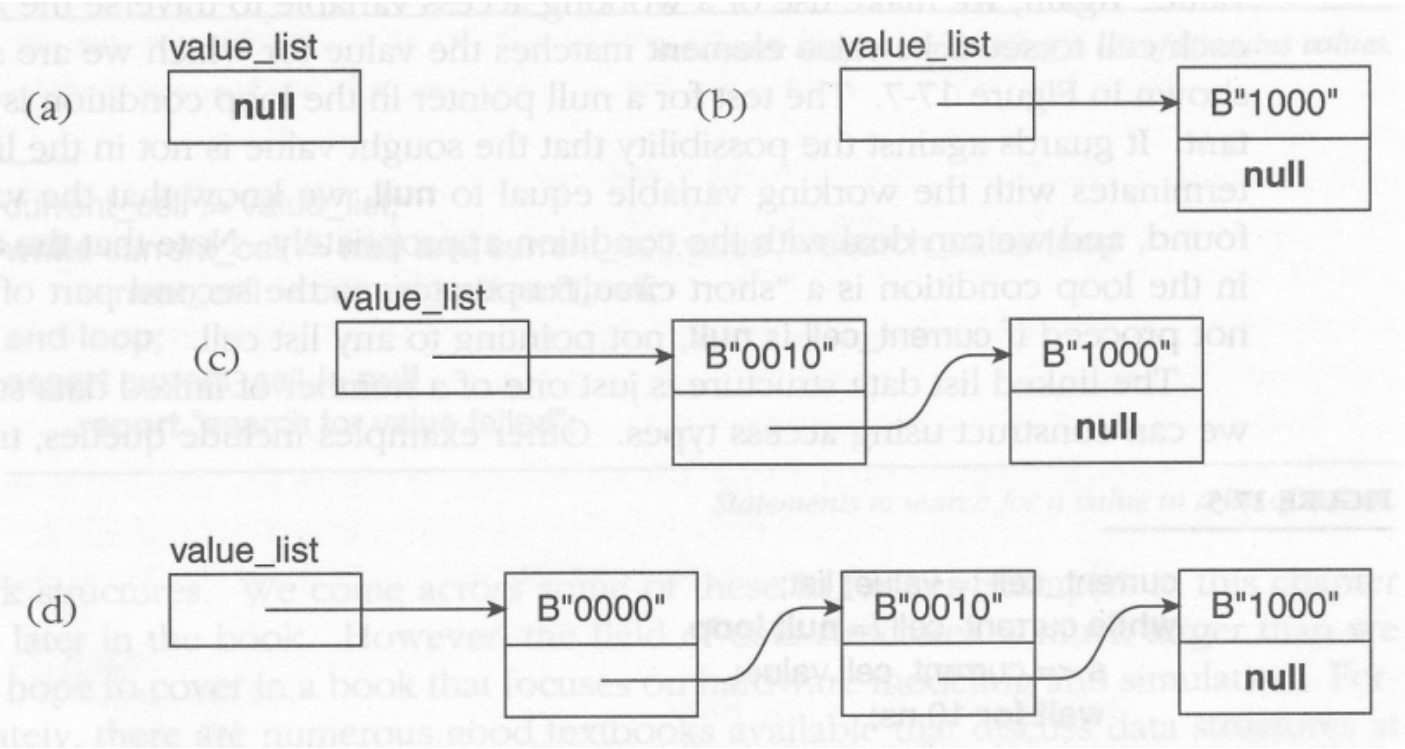
Lenka lister

```
process is

  --ikke komplett typedefinisjon, bare navn interessant nå
  type value_cell;

  --deklarasjon av pekertype
  type value_ptr is access value_cell;

  --komplett typedefinisjon
  --med definisjon av peker til neste celle
  type value_cell is record
      value : bit_vector(0 to 3);
      next_cell : value_ptr;
    end record value_cell;

  variable value_list : value_ptr;--(a)

begin
```

Lenka lister

```
begin
  --Her skal value_list inneholde en null peker
  if value_list /= null then
    report "value_list /= null";
  end if;

  --Bygger opp listen
  value_list := new value_cell'( B"1000", value_list );--(b)
  value_list := new value_cell'( B"0010", value_list );--(c)
  value_list := new value_cell'( B"0000", value_list );--(d)
  wait;
end process;
```

Lenka lister



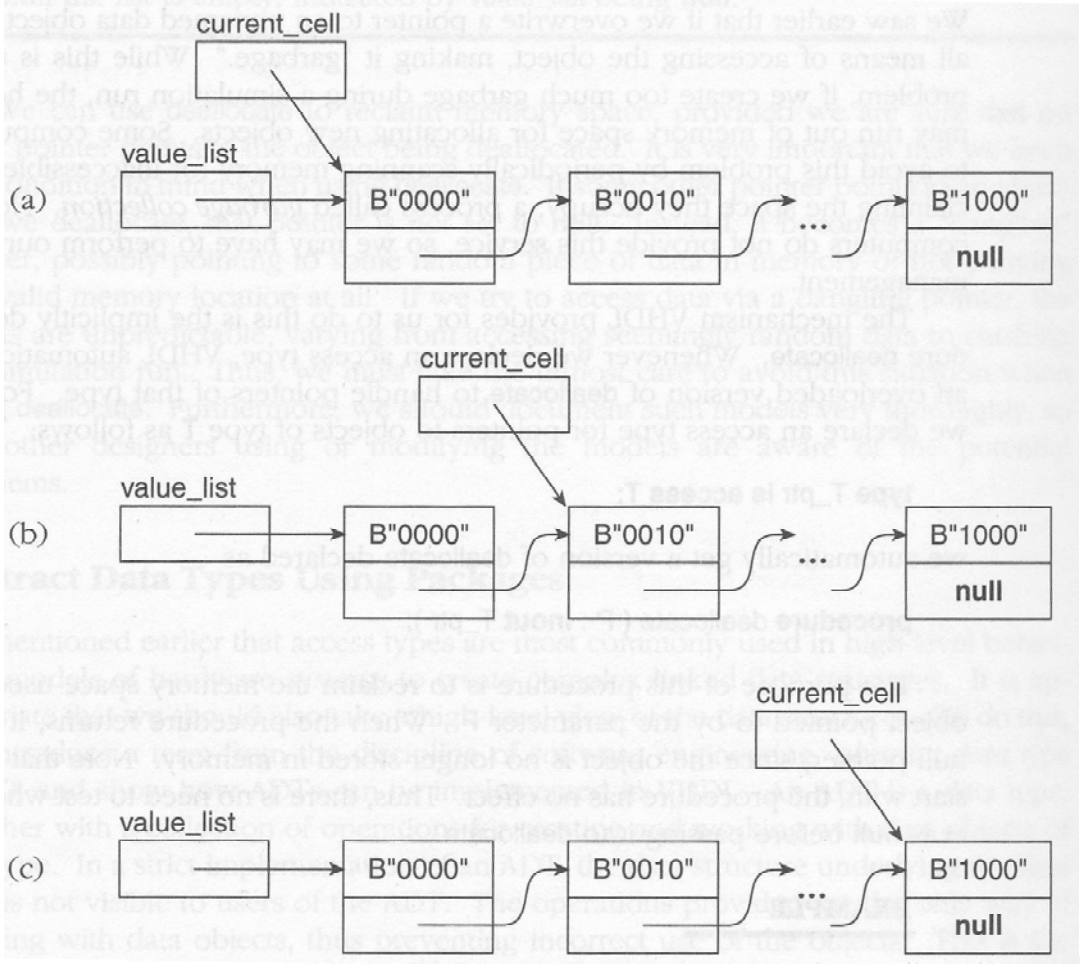
Lage stimuli ved lenka lister

```
  variable value_list, current_cell : value_ptr;

begin
  value_list := new value_cell'( B"1000", value_list );
  value_list := new value_cell'( B"0010", value_list );
  value_list := new value_cell'( B"0000", value_list );

  current_cell := value_list;
  while current_cell /= null loop
    s <= current_cell.value;
    wait for 10 ns;
    current_cell := current_cell.next_cell;
  end loop;

  wait;
end process;
```

Søking i lenka lister

```
current_cell := value_list;
while current_cell /= null
    and current_cell.value /= search_value loop
  current_cell := current_cell.next_cell;
end loop;
assert current_cell /= null
  report "search for value failed";
```

Søking i lenka lister



De-allokering

- Når vi definerer en accesstype får vi automatisk dannet en procedyre *deallocate*

```
type T is (t1, t2, t3);

type T_ptr is access T;

--deallocate dannes automatisk
--trenger ikke deklareres
procedure deallocate ( P : inout T_ptr );

procedure deallocate ( P : inout T_ptr ) is
begin
  null;
end procedure deallocate;
```

De-allokering

```
--Sletting av enkeltelement
cell_to_be_deleted := value_list;
value_list := value_list.next_cell;
deallocate(cell_to_be_deleted);

--Sletting av alle elementer
while value_list /= null loop
  cell_to_be_deleted := value_list;
  value_list := value_list.next_cell;
  deallocate(cell_to_be_deleted);
end loop;
```