Programació funcional

Albert Rubio

Especialitat de Computació
Grau en Enginyeria Informàtica

FIB

Pla de la sessió

- Testing amb QuickCheck
- Modelat amb llenguatges funcionals
- Cas d'estudi: modelant de circuits

Testing amb QuickCheck

Fer testing definint propietats.

Per exemple al `quicksort`

- idempotent
- ordenat
- mínim
- màxim

Usem `quickCheck` o `verboseCheck`

Veure exemples a `quick.hs`

Modelat amb llenguatges funcionals

- Fàcil definir models simbòlics (usant `data`)
- Fàcil definir propietats per pattern matching.
- Fàcil definir regles.
- Obtenim especificacions executables

Exemple: especificació de circuits digitals.
Llibreria `Lava` de Haskell per descriure Hardware.

Es pot veure com un llenguatge especific d'un domini (DSL).

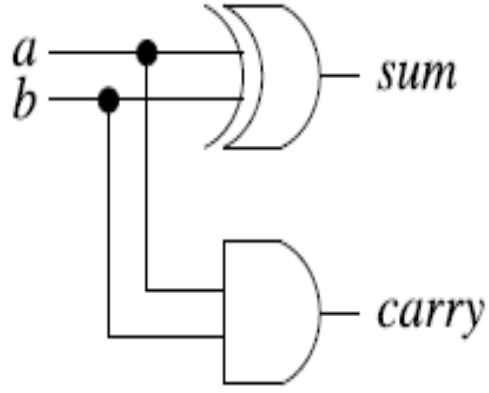
Modelat amb llenguatges funcionals

Exemples:

```
type Bit = Bool
type Signal = [Bit]
high, low :: Signal
high = True : high
low = False : low
--bit and
band :: Bit -> Bit -> Bit
band a b = a && b
--bit xor
bxor :: Bit -> Bit -> Bit
bxor a b = a /= b
--bit or
--bit not ...
```

Modelat amb llenguatges funcionals

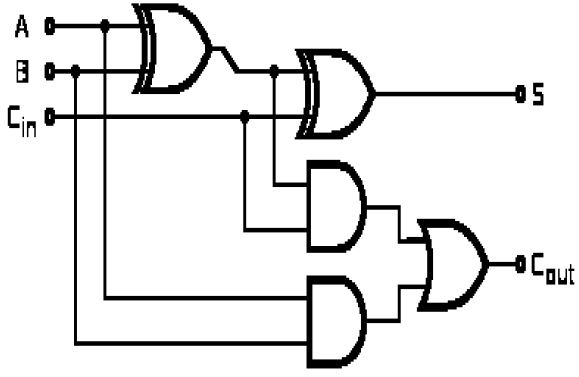
Half Adder




```
halfAdd :: Bit -> Bit -> (Bit, Bit)
halfAdd a b = (sum, carry)
   where sum = a ‘bxor‘ b
         carry = a ‘band‘ b
```

Modelat amb llenguatges funcionals

Full Adder

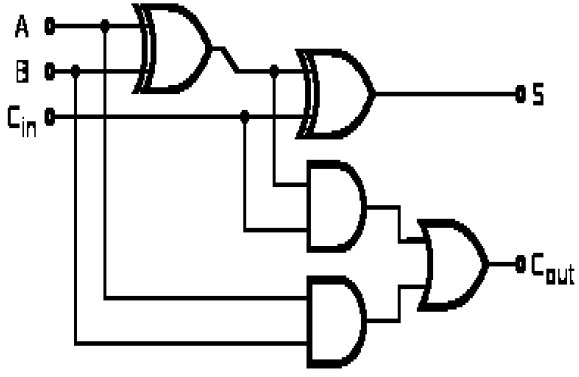



```
fullAdd :: Bit -> Bit -> Bit -> (Bit, Bit)

???
```

Modelat amb llenguatges funcionals

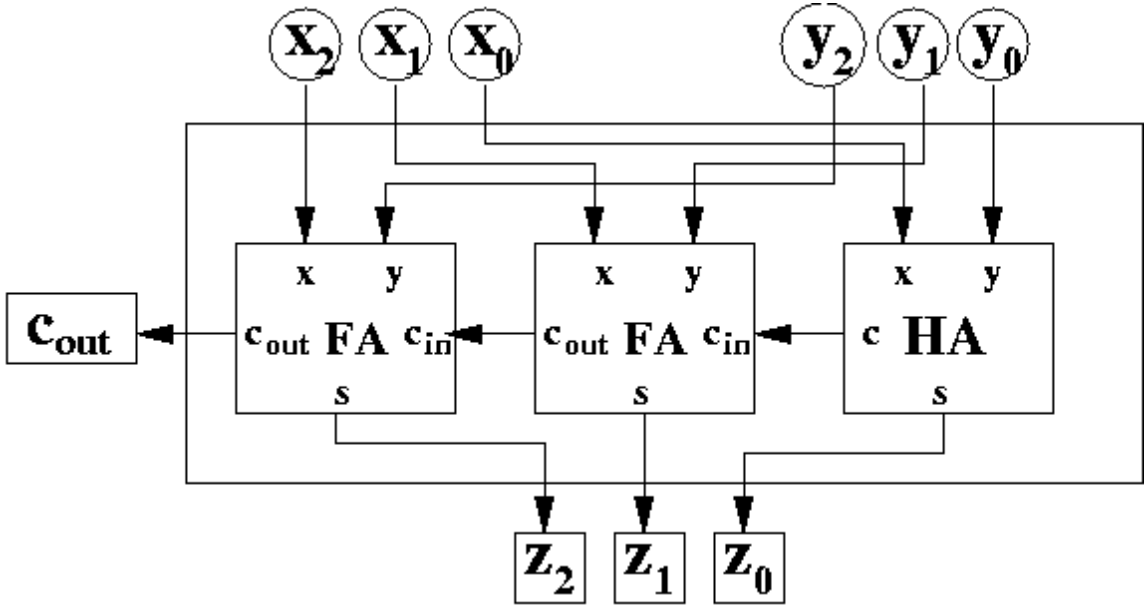
Full Adder




```
fullAdd :: Bit -> Bit -> Bit -> (Bit, Bit)
fullAdd a b ci = (s, co)
   where (s1,c1)=halfAdd a b
         (s,c2)=halfAdd s1 ci
         co = c1 ‘band‘ c2
```

Modelat amb llenguatges funcionals

Ripple Carry parallel Adder

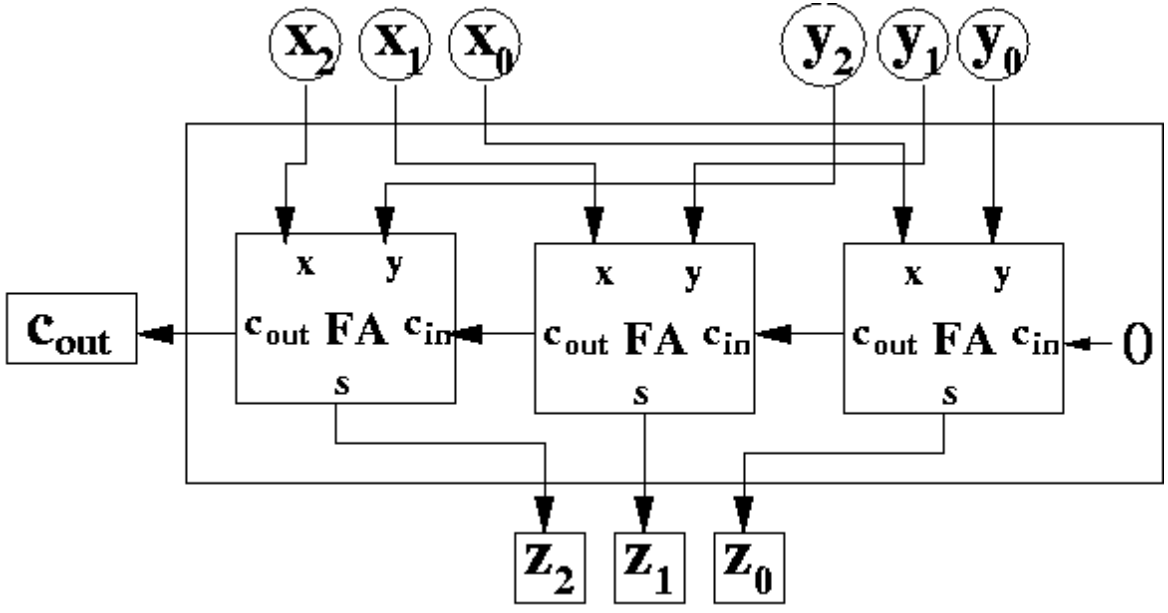



```
rcpAdder3 :: (Bit,Bit,Bit) -> (Bit,Bit,Bit) -> ((Bit,Bit,Bit), Bit)

???
```

Modelat amb llenguatges funcionals

Ripple Carry parallel Adder

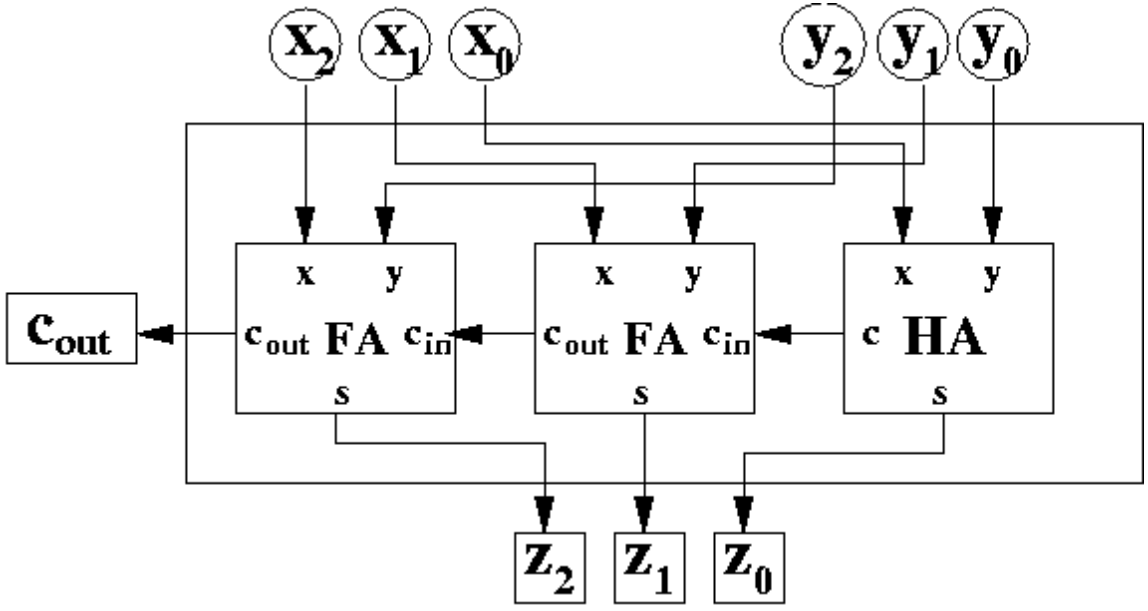



```
rcpAdder3 :: (Bit,Bit,Bit) -> (Bit,Bit,Bit) -> ((Bit,Bit,Bit), Bit)
```

???

Modelat amb llenguatges funcionals

Ripple Carry parallel Adder



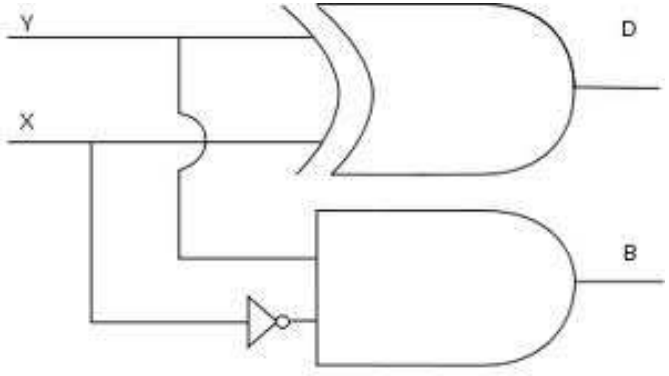

```
rcpAdder3 :: (Bit,Bit,Bit) -> (Bit,Bit,Bit) -> ((Bit,Bit,Bit), Bit)
rcpAdder3 (x0,x1,x2) (y0,y1,y2) = ((z0,z1,z2),co)
   where (z0,c0)=halfAdd x0 y0
         (z1,c1)=fullAdd x1 y1 c0
         (z2,co)=fullAdd x2 y2 c1
```

Modelat amb llenguatges funcionals

Per evitar un retard inacceptable s'introdueix el
Lookahead Carry adder
que usa le **Lookahead Carry generator** (amb n-carries)

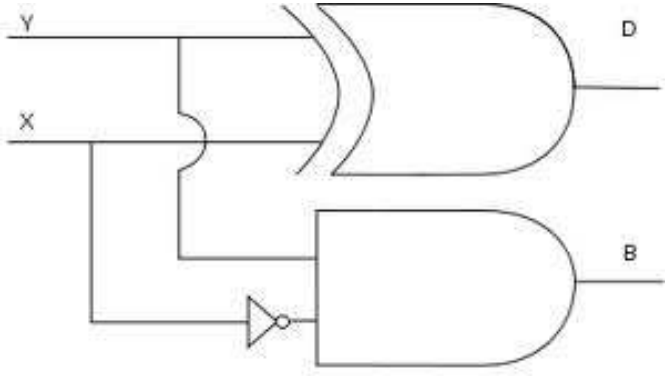
Modelat amb llenguatges funcionals

Half substractor



Modelat amb llenguatges funcionals

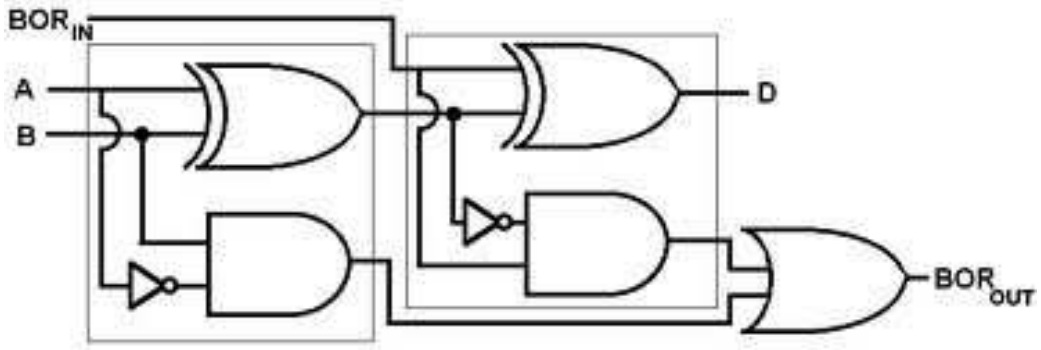
Half substractor



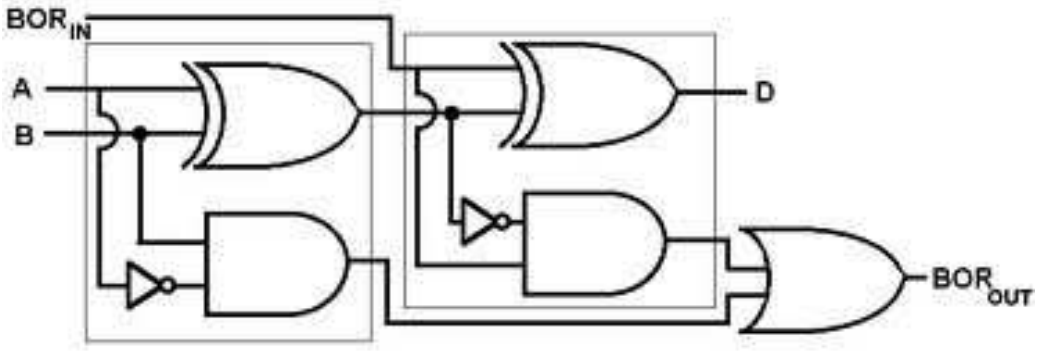

```
halfSub :: Bit -> Bit -> (Bit, Bit)
halfSub x y = (d, b)
   where d = x ‘bxor‘ y
         b = (bnot x) ‘band‘ y
```

Modelat amb llenguatges funcionals

Full substractor



Modelat amb llenguatges funcionals



Full substractor

```
fullSub :: Bit -> Bit -> Bit -> (Bit, Bit)
fullSub a b bi = (d, bo)
   where (d1,b1)=halfSub a b
         (d,b2)= halfSub bi d1
          bo   = b1 ‘bor‘ b2
```

Modelat amb llenguatges funcionals

Podem usar data per expresar l'estructura dels circuits

```
data sBit = Zero | One

data Circuit = Wire Bit | And2 Circuit Circuit
             | Inv Circuit | Xor2 Circuit Circu
             | HA Circuit Circuit ..

eval::Circuit -> Bool
eval (Wire Zero) = False
eval (Wire One)  = True
eval (And2 c1 c2) = (eval c1) && (eval c2)
eval (Inv c) = not (eval c)
eval (Xor2 c1 c2) = (eval c1) /= (eval c2)
eval (HA c1 c2) = (eval c1) /= (eval c2)
```