**Custom Computing Assessed Coursework**

* 1. Rsh Definition:  
       
     rsh = <a,<b,z>> $wire <<a,b>,z>.  
       
     Rebecca code for P1:

INCLUDE "prelude.rby".

n = 4.

P1 = (Q1)^n ; (fork)^~1.

Q1 = snd fork ; rsh ; [add, (DI 0)^~1].

current = P1.

Diagram of P1:

Add

D-1

Add

D-1

Add

D-1

Add

D-1

Simulation:

re " x1 ; x2 ; x3 ; x4 ; x5 ; x6"

Simulation start :

0 - <x1,0> ~ ((((x1 + 0) + 0) + 0) + 0)

1 - <x2,0> ~ ((((x2 + 0) + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0))

2 - <x3,0> ~ ((((x3 + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0)) + ((((x2 + 0) + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0))

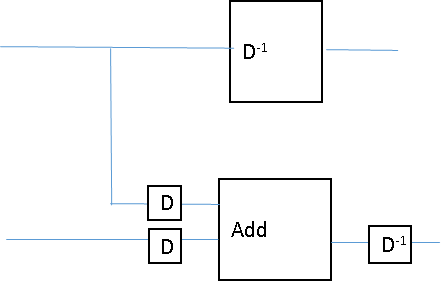
3 - <x4,0> ~ ((((x4 + 0) + ((((x1 + 0) + 0) + 0) + 0)) + ((((x2 + 0) + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0))) + (

4 - <x5,((((x1 + 0) + 0) + 0) + 0)> ~ ((((x5 + ((((x1 + 0) + 0) + 0) + 0)) + ((((x2 + 0) + 0) + 0) + ((((x1 + 0) ) + ((((x4 + 0) + ((((x1 + 0) + 0) + 0) + 0)) + ((((x2 + 0) + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0))) + ((((x3 + 0) +

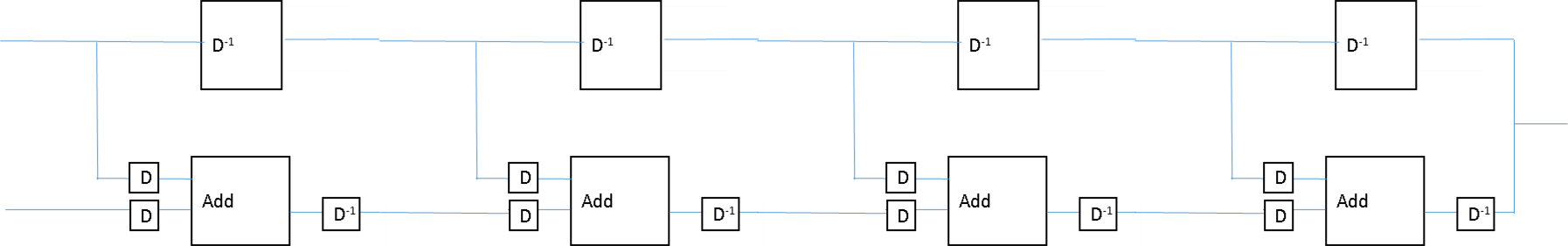
5 - <x6,((((x2 + 0) + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0))> ~ ((((x6 + ((((x2 + 0) + 0) + 0) + ((((x1 + 0) + 0) (((x4 + 0) + ((((x1 + 0) + 0) + 0) + 0)) + ((((x2 + 0) + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0))) + ((((x3 + 0) + 0) + 0)) + ((((x2 + 0) + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0))) + ((((x3 + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0)) + ((((x2 (x1 + 0) + 0) + 0) + 0))) + ((((x3 + 0) + 0) + ((((x1 + 0) + 0) + 0) + 0)) + ((((x2 + 0) + 0) + 0) + ((((x1 + 0) + 0)

* 1. We begin by placing delays at the domain and antidelays at the range of the adders, making Q2, which is described by

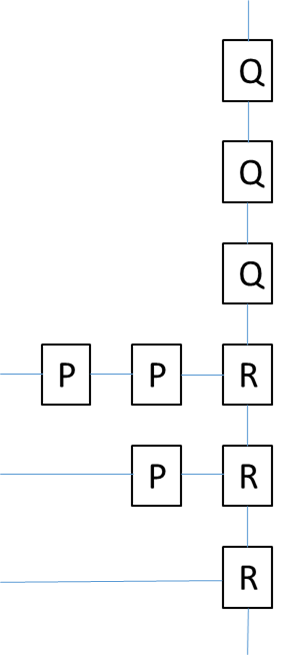
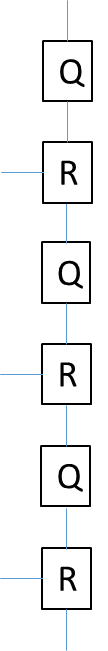
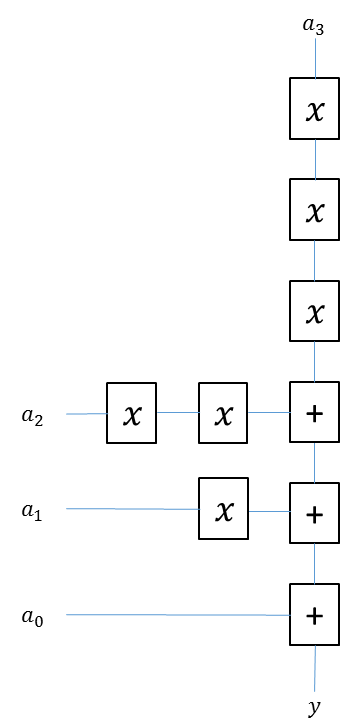
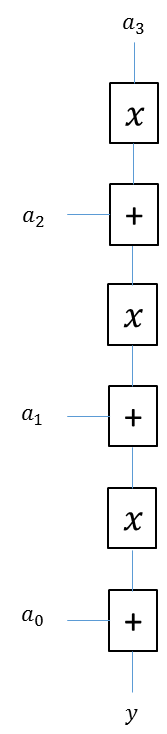
Q2 = snd fork ; rsh ; [[D,D] ; add ; D-1 , D-1].  
  
and looks like:



Making P2 look like this when n = 4:



We can transform ([D,D] ; add ; D-1)n into ([D,D] ; add)n ; D-n

* 1. Basic Step (n = 1):   
     Assumptive Step (n = k):   
     Inductive Step (n = k+1):
  2. Base Case: Inductive Case:   
       
     Base Case:   
     Inductive Case:
  3.    
       
       
       
       
     
  4. When the polynomial equation is modelled using P, Q, and R as above, both P and Q are blocks that multiply by and R is an adder. The design will look like:  
       
     When optimised using Horner’s Rule, it looks like   
        
     multx = pi1^~1 ; snd "x" ; mult.

poly = rdr 3 (snd multx ; add).

* 1. **Failure**: a deviation from a design specification  
     **Error**: a failure that results in an incorrect signal value  
     **Fault**: an error that manifests as an incorrect logical result
  2. Area of die:   
     (Gross) Number of dice on 20cm wafer:   
     (Gross) Number of dice on 30cm wafer:   
     Defect density from years 1 to 5   
     Yield for each year   
     Number of good dice   
     Cost per die for each year

**Cost per Die per Year for 20cm Wafers**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Year 1** | **Year 2** | **Year 3** | **Year 4** | **Year 5** | **Total Cost** |
| **Yield** |  |  |  |  |  |  |
|  | 60.157 | 73.182 | 89.028 | 108.305 | 131.757 |  |
| **Wafer cost** | 5000 | 4500 | 4000 | 3500 | 3000 |  |
| **Cost Per Die** | 83.116 | 61.490 | 44.929 | 32.316 | 22.769 | 244.62 |

**Cost per Die per Year for 30cm Wafers**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Year 1** | **Year 2** | **Year 3** | **Year 4** | **Year 5** | **Total** |
| **Yield** |  |  |  |  |  |  |
|  | 135.353 | 164.661 | 200.314 | 243.687 | 296.452 |  |
| **Wafer cost** | 10000 | 9000 | 8000 | 7000 | 6000 |  |
| **Cost Per Die** | 73.881 | 54.658 | 39.937 | 28.725 | 20.239 | 217.44 |

Therefore, the 30cm production process should be chosen, as it is cheaper.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **20cm Die** | | **30cm Die** | |
| **Wafer Cost** | 5000 | | 8000 | |
| **Gross Number of Dice, N** | 136.59 | | 307.33 | |
| **Defect Densities** |  |  |  |  |
| **Yield ()** | 0.631 | 0.317 | 0.631 | 0.317 |
|  | 94.677 | 43.299 | 213.02 | 97.424 |
| **Cost Per Die** | 52.811 | 115.48 | 37.554 | 82.116 |