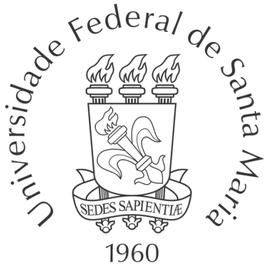
Arquitetura e Organização de Computadores III
UFSM00290

Trabalho de Construção de Circuito para Cálculo de Raiz Quadrada

Vinícius Schultz



Universidade Federal de Santa Maria (UFSM)
2025

MUDANÇAS NO CÓDIGO

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 1;
    sum_2  = 2;
    square = 4;
    ready  = 1;
    while(ready == 1){
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2 + 1;
        if(input-square < 0)
            ready = 0;
        else
            ready = 1;
    }
}
```

MUDANÇAS NO CÓDIGO

o teste sempre vai ser verdadeiro na primeira iteração

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 1;
    sum_2  = 2;
    square = 4;
    ready  = 1;
    while(ready == 1){
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2 + 1;
        if(input-square < 0)
            ready = 0;
        else
            ready = 1;
    }
}
```

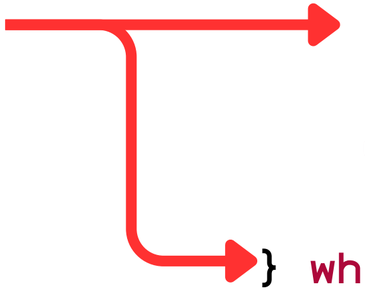
MUDANÇAS NO CÓDIGO

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 1;
    sum_2  = 2;
    square = 4;
    ready  = 1;
    do{
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2 + 1;
        if(input-square < 0)
            ready = 0;
        else
            ready = 1;
    } while(ready == 1)
}
```

MUDANÇAS NO CÓDIGO

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 1;
    sum_2  = 2;
    square = 4;
    ready  = 1;
    do{
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2 + 1;
        if(input-square < 0)
            ready = 0;
        else
            ready = 1;
    } while(ready == 1)
}
```

o teste do while é dependente do teste do if



MUDANÇAS NO CÓDIGO

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 1;
    sum_2  = 2;
    square = 4;
    ready  = 1;
    do{
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2 + 1;
    } while(!(input-square < 0))
    ready = 0;
}
```

MUDANÇAS NO CÓDIGO

atribuições repetidas

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 1;
    sum_2  = 2;
    square = 4;
    ready  = 1;
    do{
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2 + 1;
    } while(!(input-square < 0))
    ready = 0;
}
```

MUDANÇAS NO CÓDIGO

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 2;
    sum_2  = 4;
    square = 9;
    ready  = 1;
    while(!(input-square < 0)){
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2 + 1;
    }
    ready = 0;
}
```

MUDANÇAS NO CÓDIGO

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 2;
    sum_2  = 4;
    square = 9;
    ready  = 1;
    while(!(input-square < 0)){
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2 + 1;
    }
    ready = 0;
}
```

sum_2 + 2
sum_2 + 1

MUDANÇAS NO CÓDIGO

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 2;
    sum_2  = 5;
    square = 9;
    ready  = 1;
    while(!(input-square < 0)){
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2;
    }
    ready = 0;
}
```

MUDANÇAS NO CÓDIGO

BUGS?

Para raízes de 0 a 3 a resposta retornada é 2;

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 2;
    sum_2  = 5;
    square = 9;
    ready  = 1;
    while(!(input-square < 0)){
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2;
    }
    ready = 0;
}
```

MUDANÇAS NO CÓDIGO

SOLUÇÃO:

“Desfazer” os cálculos até root=0;

```
int main(){
    int root, sum_2, square, ready, input=49;
    root   = 0;
    sum_2  = 1;
    square = 1;
    ready  = 1;
    while(!(input-square < 0)){
        root   = root+1;
        sum_2  = sum_2+2;
        square = square + sum_2;
    }
    ready = 0;
}
```

MUDANÇAS NO CÓDIGO

COMPORTAMENTO DAS VARIÁVEIS

root	sum_2	square
0	1	1
1	3	4
2	5	9
3	7	16
4	9	25
5	11	36
6	13	49
7	15	64
Incrementa 1 a cada ciclo	Sequência dos números ímpares	Sequência das raízes perfeitas

MUDANÇAS NO CÓDIGO

COMPORTAMENTO DAS VARIÁVEIS

root	sum_2	square
0	1 = 0.2 + 1	1
1	3 = 1.2 + 1	4
2	5 = 2.2 + 1	9
3	7 = 3.2 + 1	16
4	9 = 4.2 + 1	25
5	11 = 5.2 + 1	36
6	13 = 6.2 + 1	49
7	15 = 7.2 + 1	64
Incrementa 1 a cada ciclo	Sequência dos números ímpares	Sequência das raízes perfeitas

MUDANÇAS NO CÓDIGO

COMPORTAMENTO DAS VARIÁVEIS

root	sum_2	square
0	1 = 0.2 + 1	1
1	3 = 1.2 + 1	4
2	5 = 2.2 + 1	9
3	7 = 3.2 + 1	16
4	9 = 4.2 + 1	25
5	11 = 5.2 + 1	36
6	13 = 6.2 + 1	49
7	15 = 7.2 + 1	64
Incrementa 1 a cada ciclo	Sequência dos números ímpares	Sequência das raízes perfeitas

MUDANÇAS NO CÓDIGO

COMPORTAMENTO DAS VARIÁVEIS

root	sum_2	square
0	1 = 0.2 + 1	1
1	3 = 1.2 + 1	4
2	5 = 2.2 + 1	9
3	7 = 3.2 + 1	16
4	9 = 4.2 + 1	25
5	11 = 5.2 + 1	36
6	13 = 6.2 + 1	49
7	15 = 7.2 + 1	64
Incrementa 1 a cada ciclo	Sequência dos números ímpares	Sequência das raízes perfeitas

```
9 + 7
9 + 3.2 + 1
square + root.2 + 1
square + root<<1 + 1
```

MUDANÇAS NO CÓDIGO

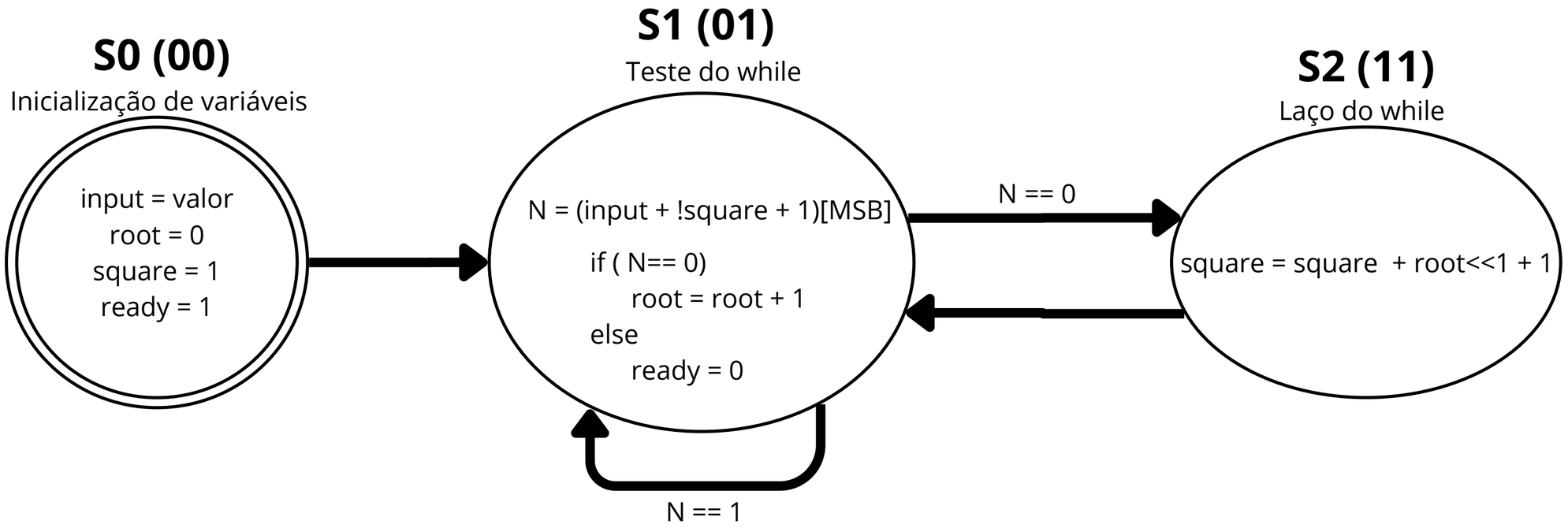
CÓDIGO ORIGINAL

```
int main(){
    int input  = 49;
    int root   = 1;
    int sum_2  = 2;
    int square = 4;
    int ready  = 1;

    while(ready == 1){
        root   = root + 1;
        sum_2  = sum_2 + 2;
        square = square + sum_2 + 1;
        if(input-square < 0)
            ready = 0;
        else
            ready = 1;
    }
}
```

CÓDIGO REESCRITO

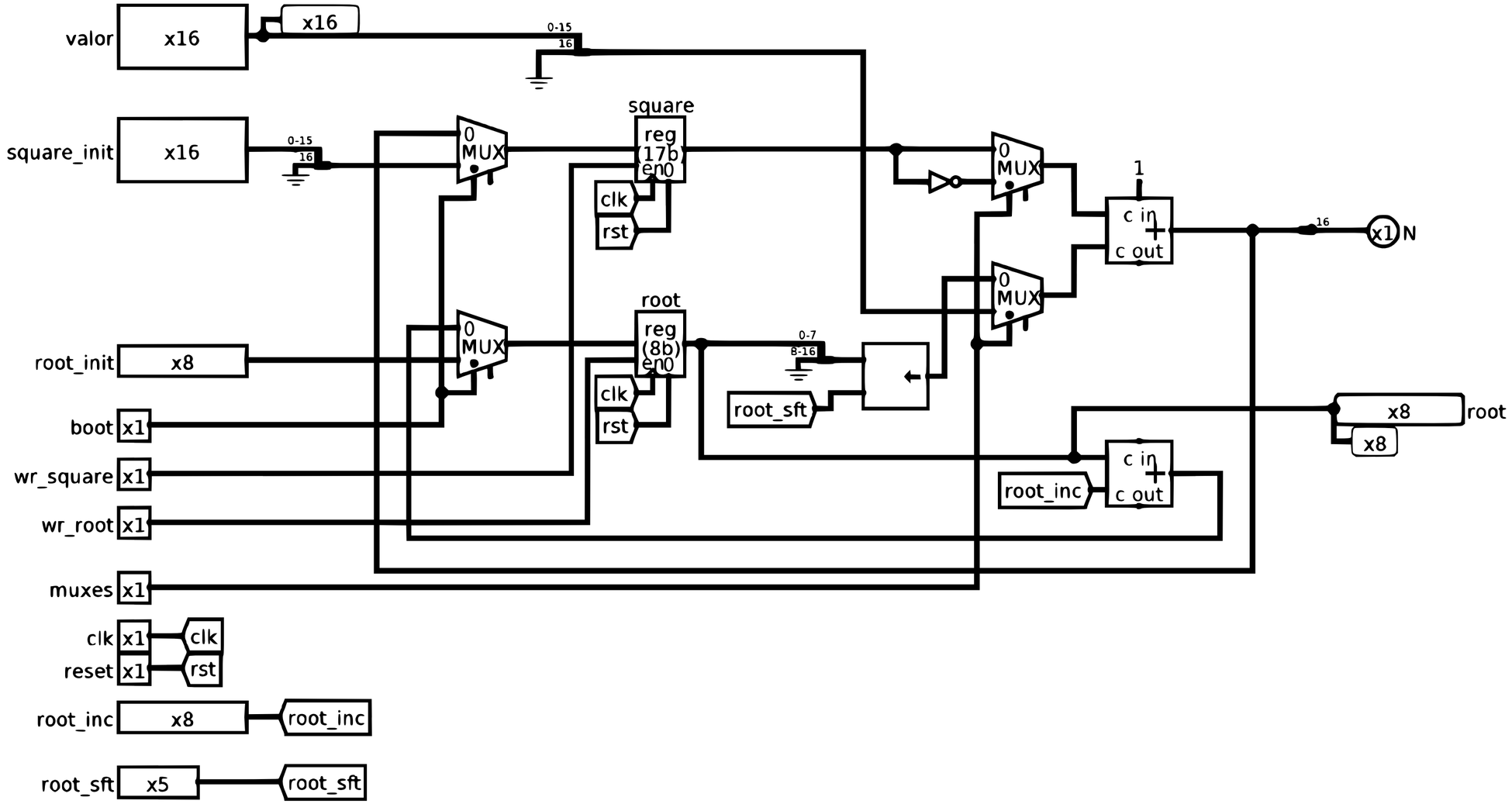
```
int main(){
    int input  = 49;
    int root   = 0;
    int square = 1;
    int ready  = 1;

    while(!(input-square < 0)){
        root   = root+1;
        square = square + (root<<1) + 1;
    }
    ready = 0;
}
```

MÁQUINA DE ESTADOS

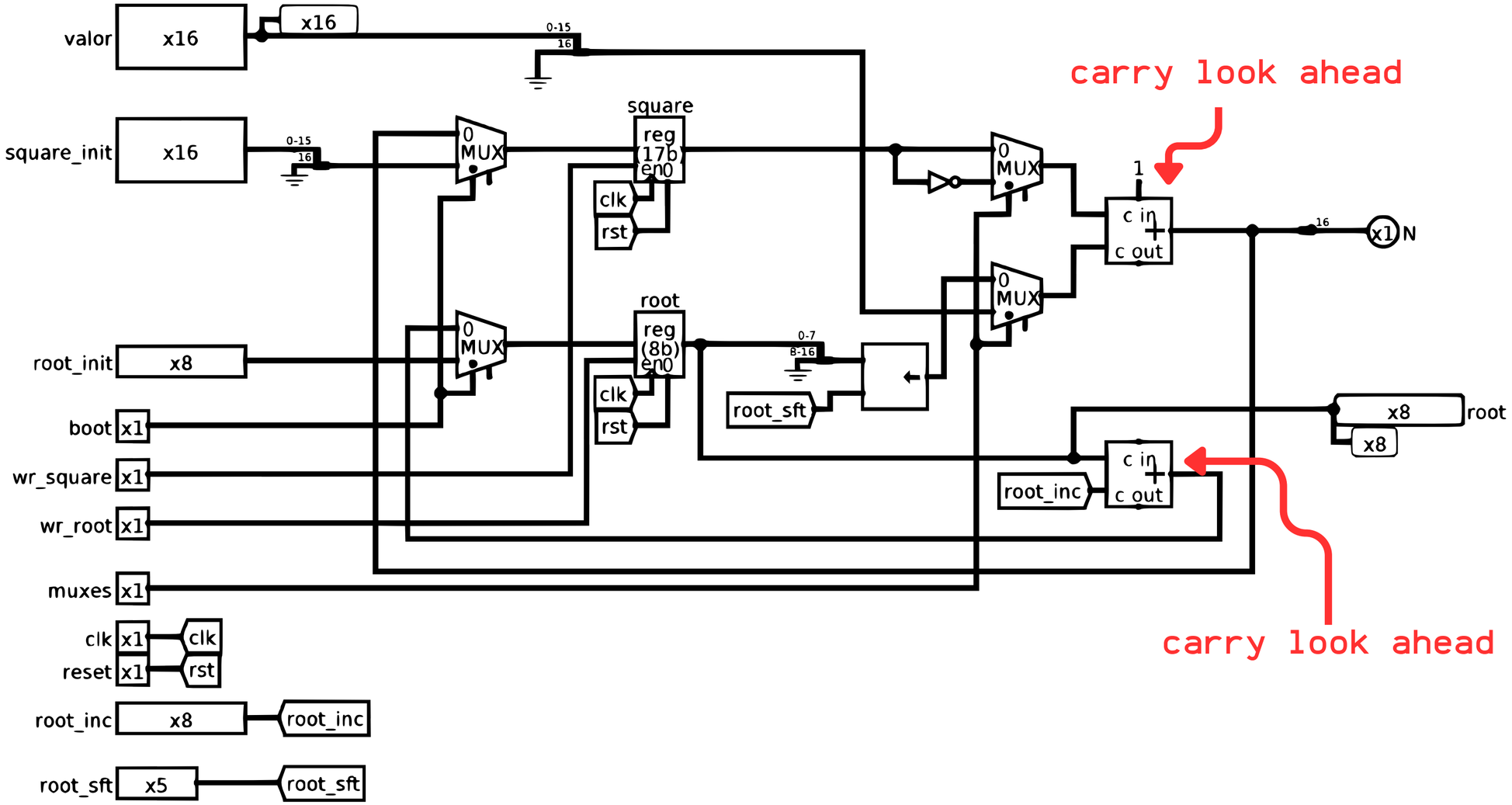



```
int main(){
    int input  = 49;
    int root   = 0;
    int square = 1;
    int ready  = 1;

    while(!(input-square < 0)){
        root   = root+1;
        square = square + (root<<1) + 1;
    }
    ready = 0;
}
```

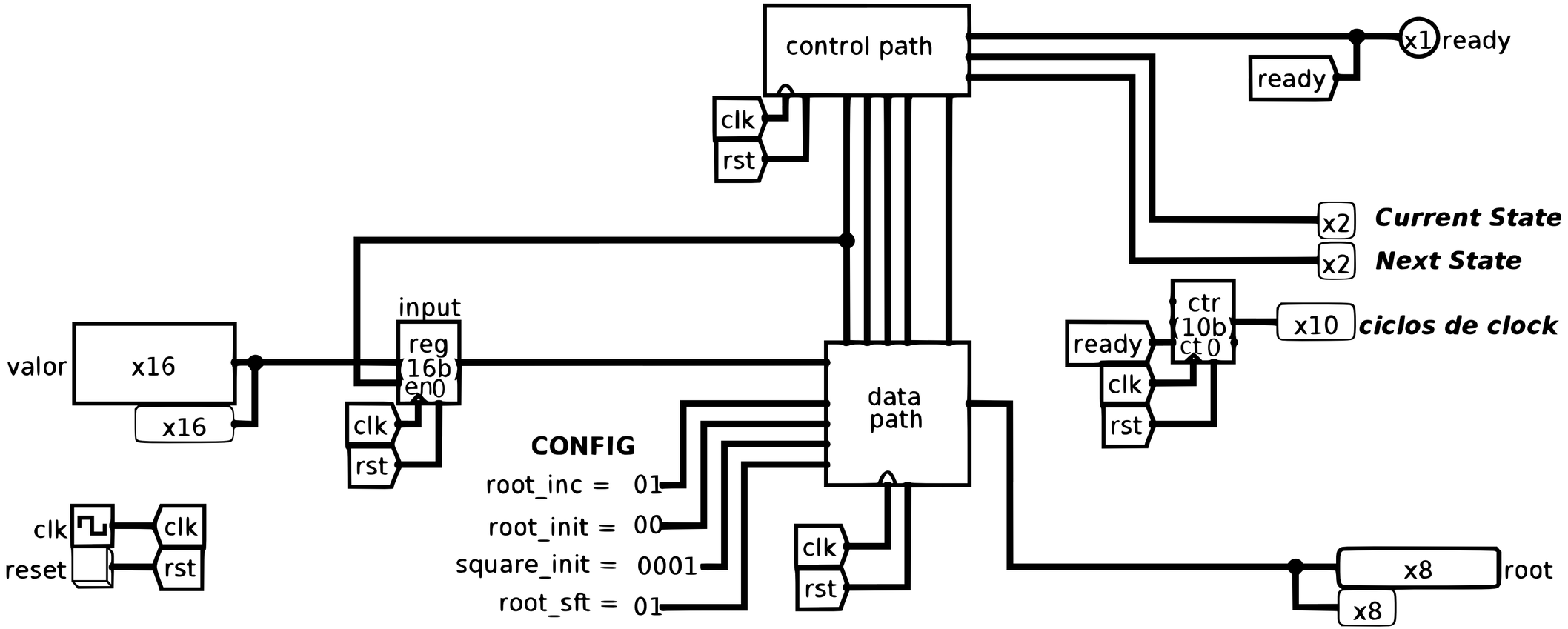
DATA PATH


valor
x16
x16
0-15
16
square_init
x16
0-15
16
0
MUX
square
reg
(17b)
en0
clk
rst
0
MUX
1
c in
c out
16
x1
N
root_init
x8
boot
x1
wr_square
x1
wr_root
x1
muxes
x1
0
MUX
root
reg
(8b)
en0
clk
rst
0-7
8-16
root_sft
0
MUX
x8
root
x8
c in
c out
root_inc
clk
x1
clk
reset
x1
rst
root_inc
x8
root_inc
root_sft
x5
root_sft

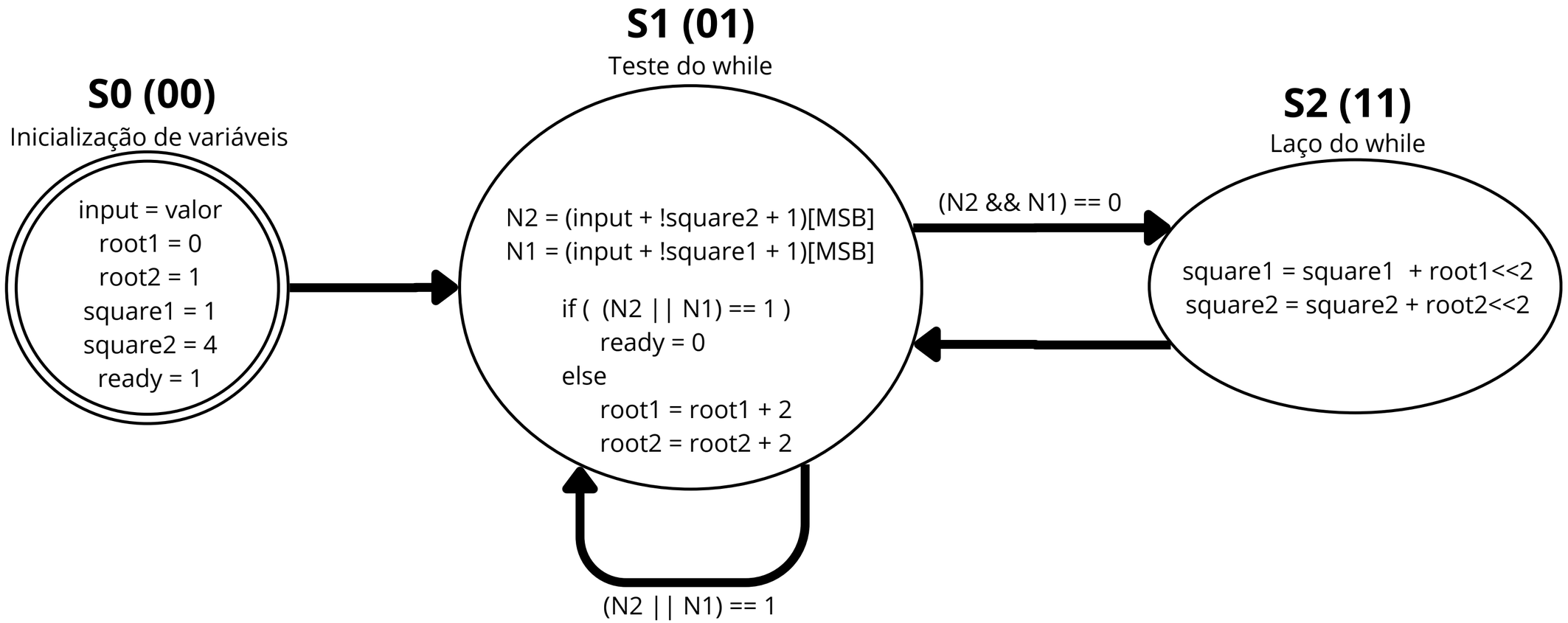
DATA PATH


valor
x16
x16
0-15
16
square_init
x16
0-15
16
0
MUX
square
reg
(17b)
en0
clk
rst
0
MUX
carry look ahead
1
c in
c out
16
x1
N
root_init
x8
boot
x1
wr_square
x1
wr_root
x1
muxes
x1
clk
x1
clk
reset
x1
rst
root_inc
x8
root_inc
root_sft
x5
root_sft
0
MUX
root
reg
(8b)
en0
clk
rst
0-7
8-16
root_sft
0
MUX
c in
c out
root_inc
x8
root
x8
carry look ahead

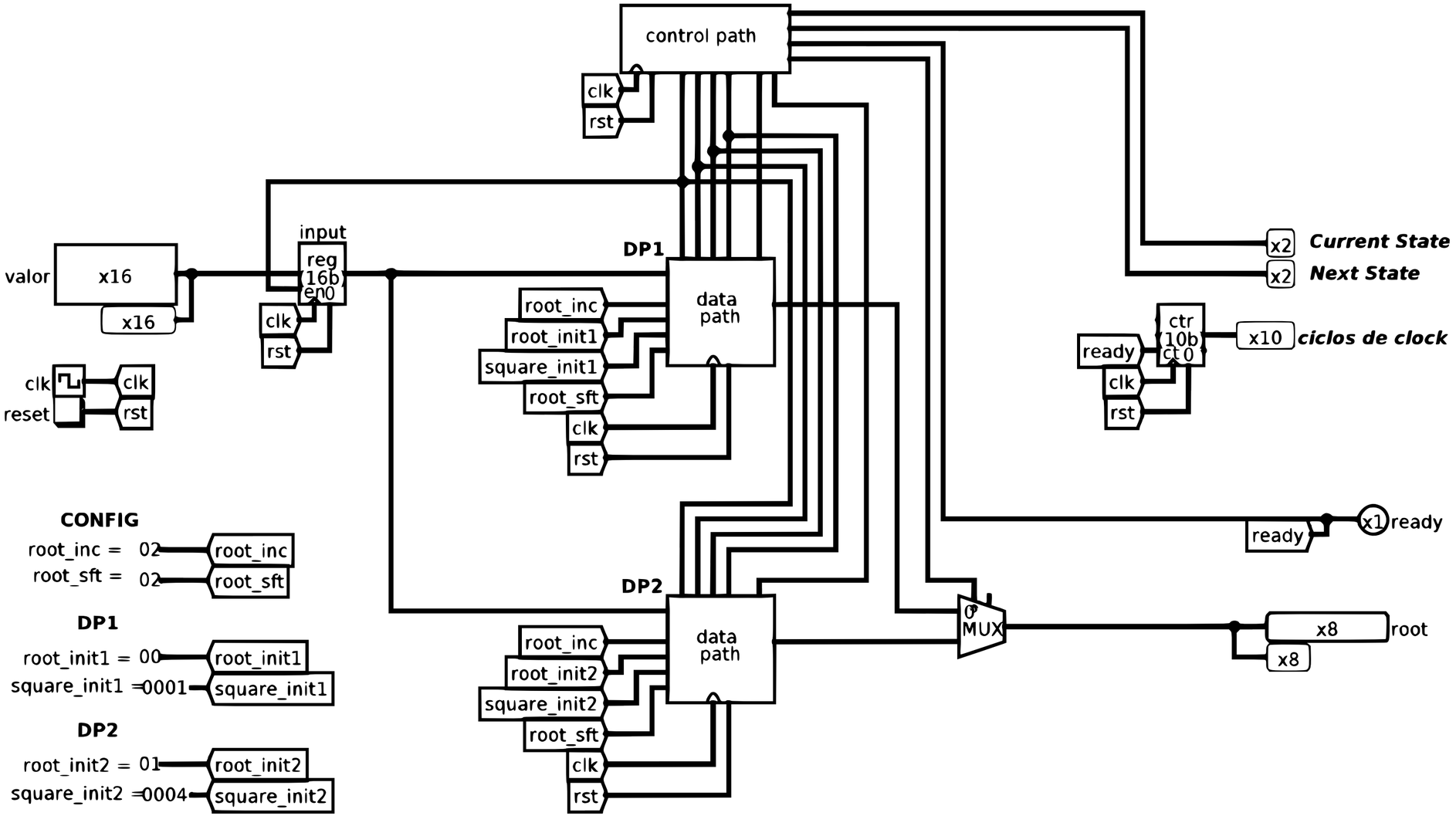
TOPO


control path
clk
rst
x1
ready
ready
x2
Current State
x2
Next State
ctr
(10b)
ct 0
x10
ciclos de clock
ready
clk
rst
input
reg
(16b)
en0
valor
x16
x16
clk
rst
clk
reset
clk
rst
CONFIG
root_inc = 01
root_init = 00
square_init = 0001
root_sft = 01
data
path
clk
rst
x8
root
x8

MÁQUINA DE ESTADOS



DATA PATH


control path
clk
rst
valor
x16
x16
input
reg
16b
en 0
clk
rst
clk
reset
clk
rst
DP1
data
path
root_inc
root_init1
square_init1
root_sft
clk
rst
x2
Current State
x2
Next State
ctr
10b
ct 0
ready
clk
rst
x10
ciclos de clock
ready
x1
ready
CONFIG
root_inc = 02
root_inc
root_sft = 02
root_sft
DP1
root_init1 = 00
root_init1
square_init1 = 0001
square_init1
DP2
root_init2 = 01
root_init2
square_init2 = 0004
square_init2
DP2
data
path
root_inc
root_init2
square_init2
root_sft
clk
rst
0
MUX
x8
root
x8

RESULTADOS

- Hardware Utilizado

	16 bits	17 bits	8 bits
Somadores	ø	2	2
Multiplexadores	ø	6	3
Registradores	1	2	2

- Ciclos necessários para calcular √65535:

255

RESULTADOS

- Caminho Crítico