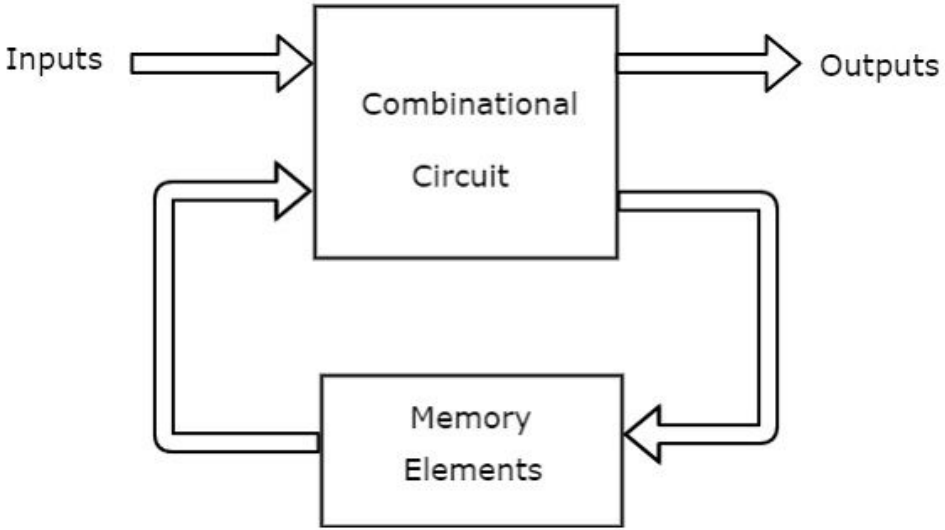
MANIPULATION OF CLOCKED SEQUENTIAL CIRCUITS

Arthur Takeshi e Renan Cunha

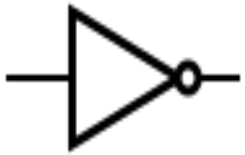
Simulação de Circuitos Sequenciais com Prolog



SWI Prolog

Portas Lógicas

```
% not(E, S)
not(0, 1).
not(1, 0).
```

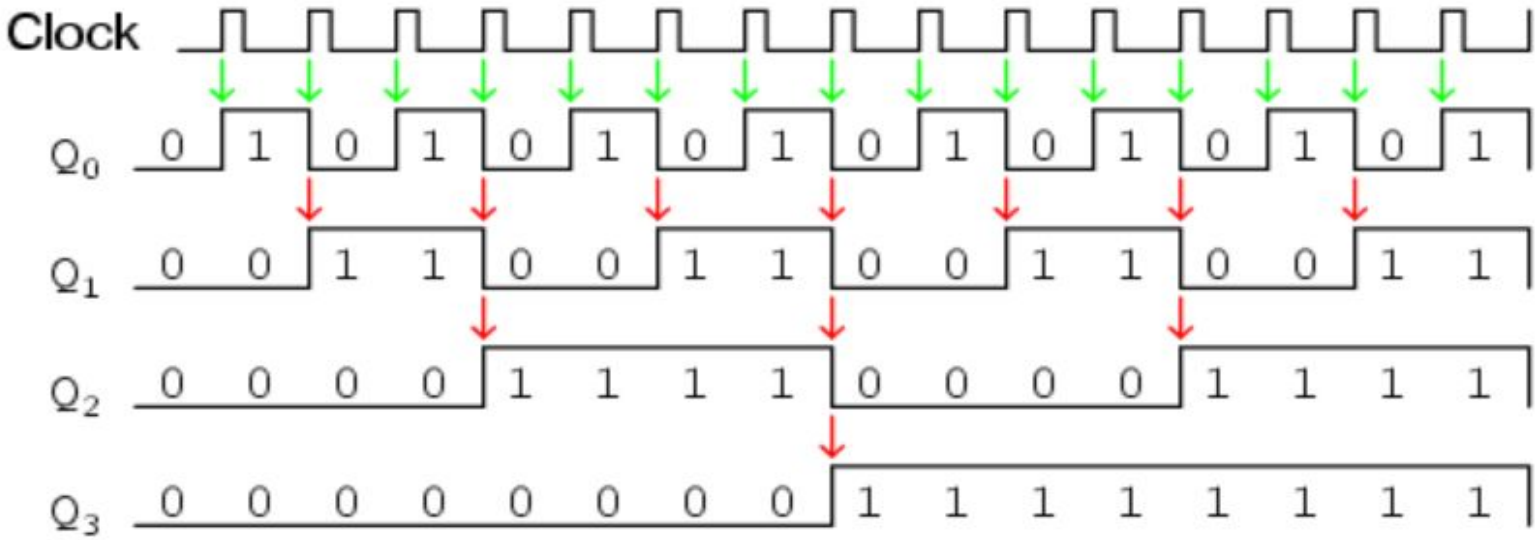


```
% xor(E1, E2, S).
xor(0, 0, 0).
xor(0, 1, 1).
xor(1, 0, 1).
xor(1, 1, 0).
```

```
% or(E1, E2, S)
or(0, 0, 0).
or(0, 1, 1).
or(1, 0, 1).
or(1, 1, 1).
```

```
% and(E1, E2, S)
and(0, 0, 0).
and(0, 1, 0).
and(1, 0, 0).
and(1, 1, 1).
```

Clock




```
%C é a lista com os pulsos de clock.
predicado1(C).
?-predicado1([0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1])
```

Flip Flop D

D Flip-flop

Symbol

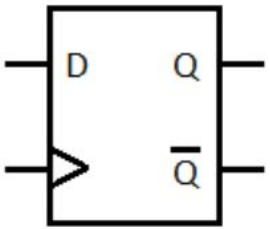
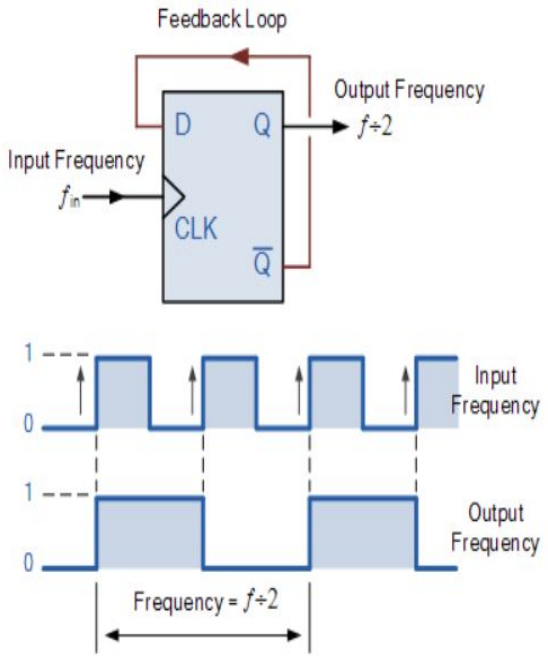


Table of truth:

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

```
% dff(Entrada, Pulso, Estado Atual, Prox. Estado)
dff(_, 0, Q, Q).
dff(D, 1 ,_, D).
```

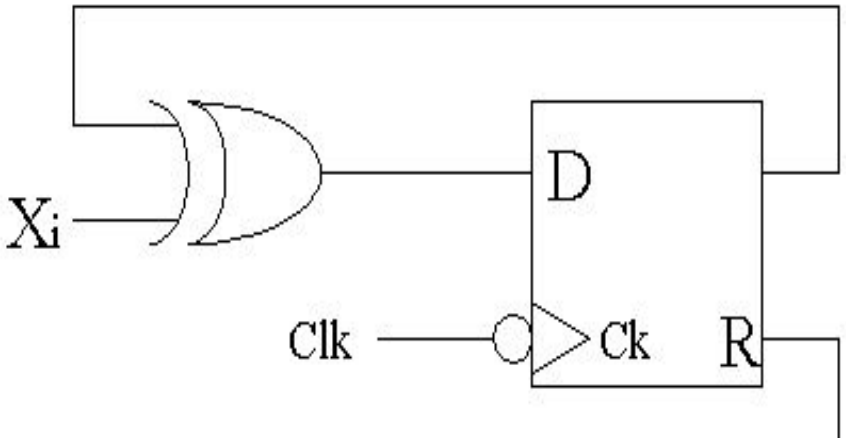
Divisor de Frequência




```
%div(Pulso, Estado Inicial, Saída)
div(C,Q,Z) :- not(Q,D), dff(D, C, Q, Z).

%divide(Clock, Estado Inicial, Lista de Saída)
divide([],_,[]).
divide([P|Ps], S, [Q|Qs]) :-
        div(P, S, Q),
        divide(Ps, Q, Qs).

?- divide([1,1,1,1,1,1], 0, Q).
Q = [1, 0, 1, 0, 1, 0]
?- divide([0, 1, 0, 0, 1, 1, 0, 0], 0, Q).
Q = [0, 1, 1, 1, 0, 1, 1, 1].
```

Sequential Parity Checker

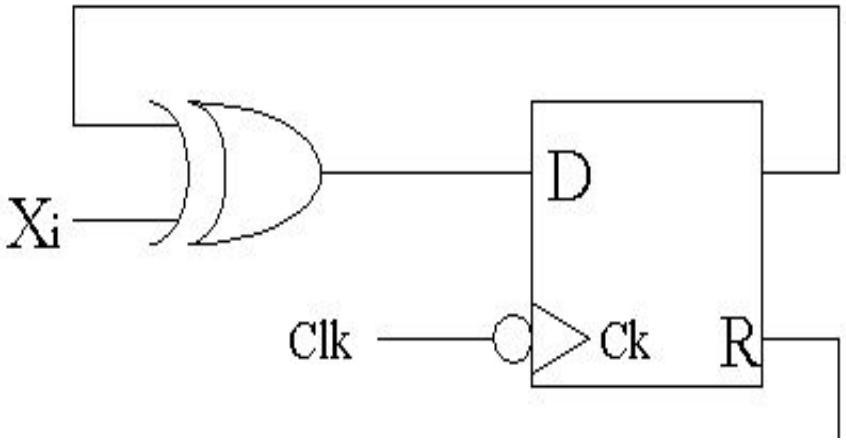
- Diz se a quantidade de 1s recebidos é impar ou não a cada clock.




```
% Clock, Entrada, Estado, P. Estado
par(Clock, X, Z, Z1) :-
        xor(X, Z, T),
        dff(T, Clock, Z, Z1).
```

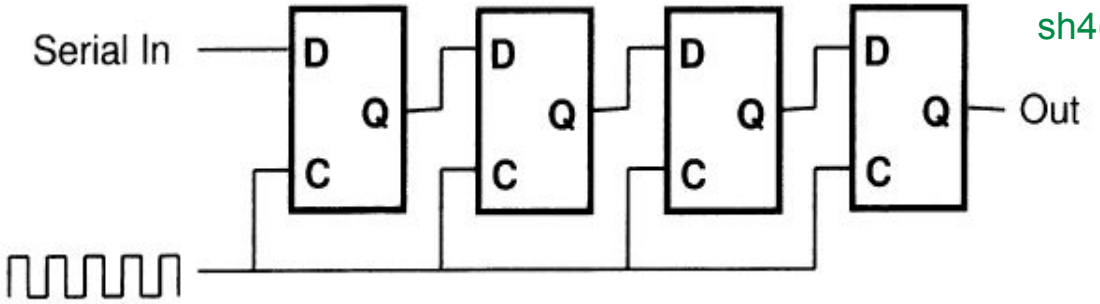
```
% Lista de todas as variáveis
parity([],_,_,[]).
parity([C|Cs],[S|Ss],N,[Z|L]) :-
        par(C,S,N,Z),
        parity(Cs,Ss,Z,L).
```

Sequential Parity Checker




```
?- parity([1,1,1,1,1,1], [1,0,0,1,1,0], 0, Q).
Q = [1, 1, 1,0, 1, 1].
```

Four-Stage Shift Register

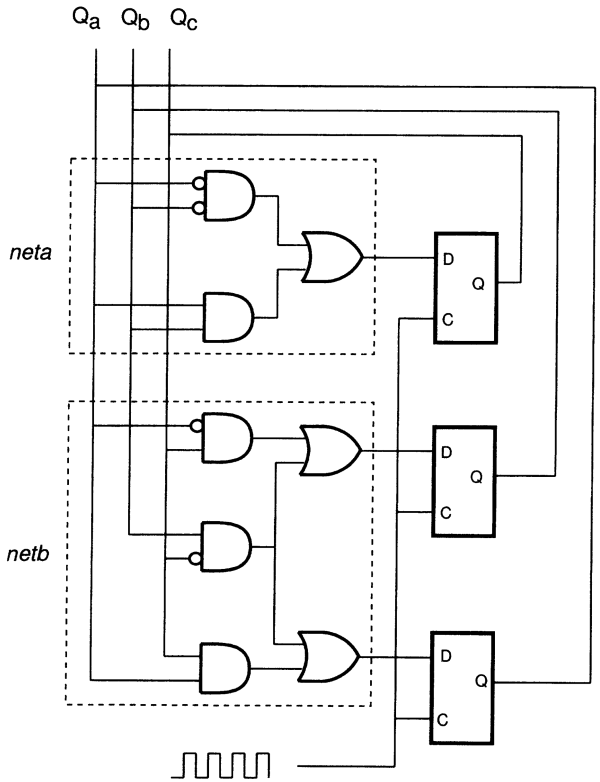



```
%sh4(pulso, Entrada, estados, prox. estados)
sh4(C,D,s(Q1,Q2,Q3,Q4),s(N1,N2,N3,N4)):-
        dff(D,C,Q1,N1),
        dff(Q1,C,Q2,N2),
        dff(Q2,C,Q3,N3),
        dff(Q3,C,Q4,N4).
```

```
shifter([],_,_,[]).
shifter([C|Cs],[S|Ss],A,[Q|L]):-
        sh4(C,S,A,N),
        N = s(_,_,_,Q),
        shifter(Cs,Ss,N,L).
```

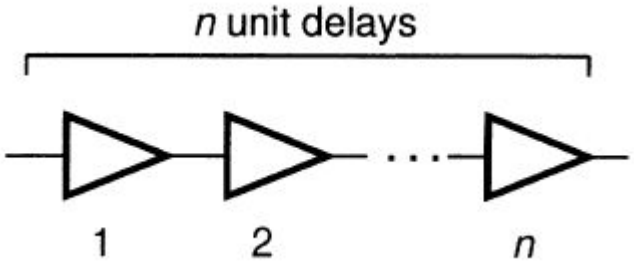
```
%?- Example: shifter([1,1,1,1,1,1,1,1,1,1], [1,0,0,1,1,0,0,1,1,1], s(0,0,0,0), L).
L = [0, 0, 0, 1, 0, 0, 1, 1, 0, 0]
```

Gray Code Counter




```
neta(A,B,Q):- and(A,B,T1), not(A,NA), not(B,NB),
        and(NA,NB,T2), or(T1,T2,Q).

netb(A,B,C,Q1,Q2):- and(A,C,T1), not(C,NC), and(B,NC,T2),
        not(A,NA), and(NA,C,T3), or(T1,T2,Q1), or(T2,T3,Q2).

gcc(C,s(Qa,Qb,Qc),s(Za,Zb,Zc)):-netb(Qa,Qb,Qc,D1,D2),
        neta(Qa,Qb,D3), dff(C,D1,Qa,Za), dff(C,D2,Qb,Zb),
        dff(C,D3,Qc,Zc).

testgcc([],_,[]).
testgcc([C|Cs],S,[N|Ns]):- gcc(C,S,N), testgcc(Cs,N,Ns).

?-testgee([1,1,1,1,1,1,1,1,1], s(O,O,O), Q).
Q=[s(0,0,1),s(0,1,1),s(0,1,0),s(1,1,0),s(1,1,1),
s(1,0, 1),s(1,0,0), s(0,0,0),s(0,0,1)]
```

Cascaded Components - Atrasa o clock em N unidades de tempo




```
% unit(Entrada, Estado Atual, Proximo Estado)
unit(0,0,0). unit(1,0,1). unit(0,1,0). unit(1,1,1).

% delay(Entrada, Estado Atual, Saida, Proximo Estado)
delay(A,[],A,[]).
delay(A,[S|Ss],Q,[Z|Zs]):- unit(A,S,Z), delay(S,Ss,Q,Zs).

%test(Clock, N-lista de estados iniciais, Lista de saída de pulsos)
test([],_,[]).
test([P|Ps],S,[Q|Qs]):-
    delay(P,S,Q,Z), test(Ps,Z,Qs).

%Example: test([1,1,0,0,1,1,0,0],[0,0,0],Q).
Q = [0, 0, 0, 1, 1, 0, 0, 1]
```

The End