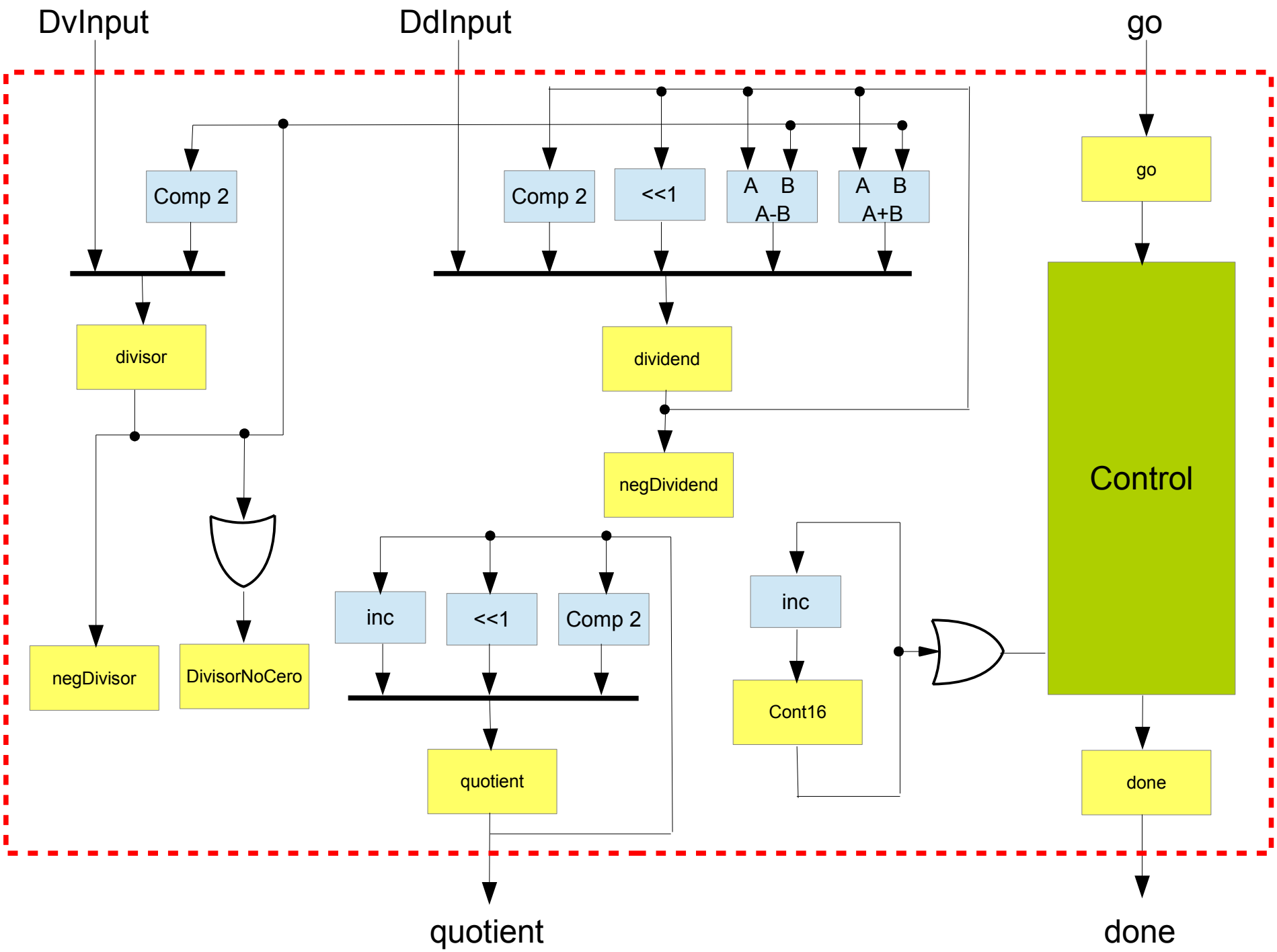
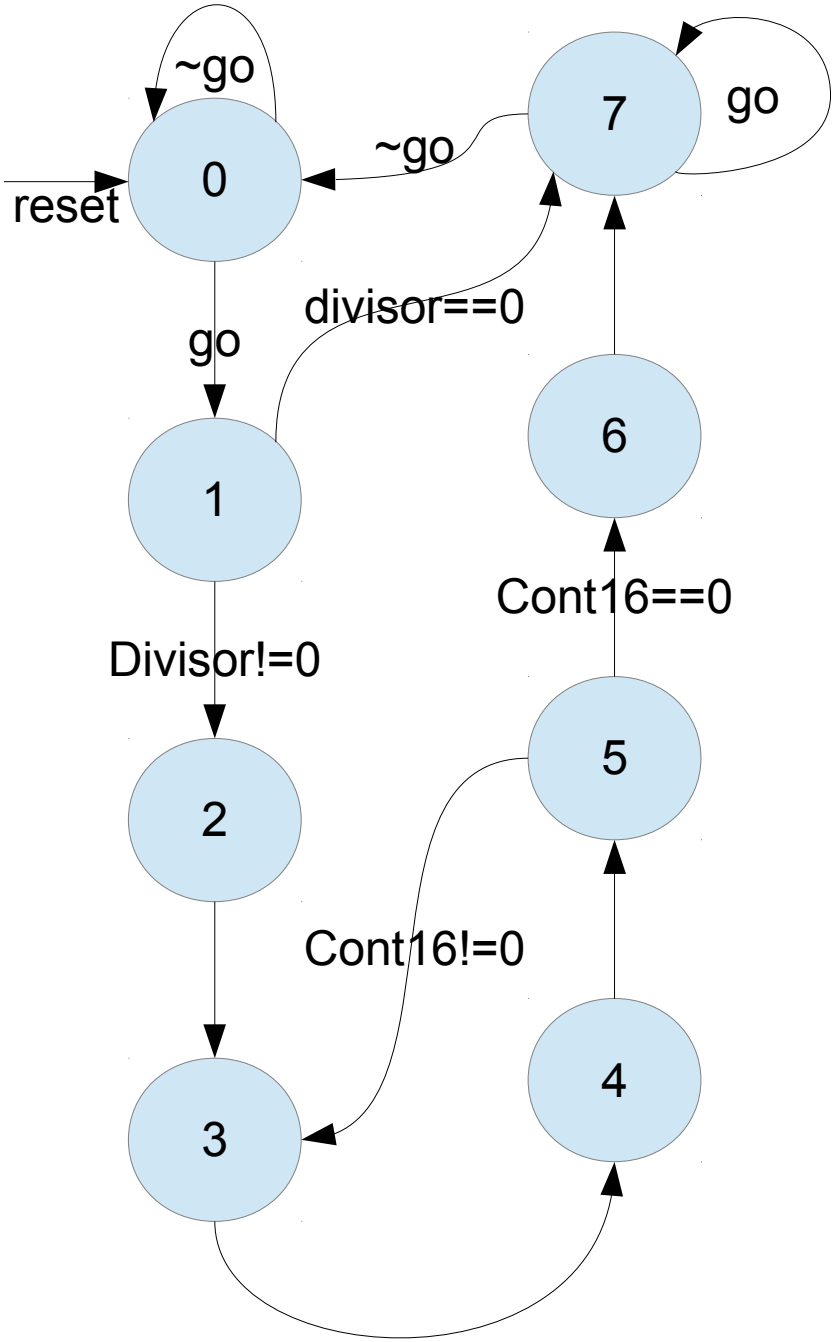
Módulo Divisor

Versión Pipeline

Randolph Steinvorth

Escuela de Ingeniería Eléctrica
Universidad de Costa Rica

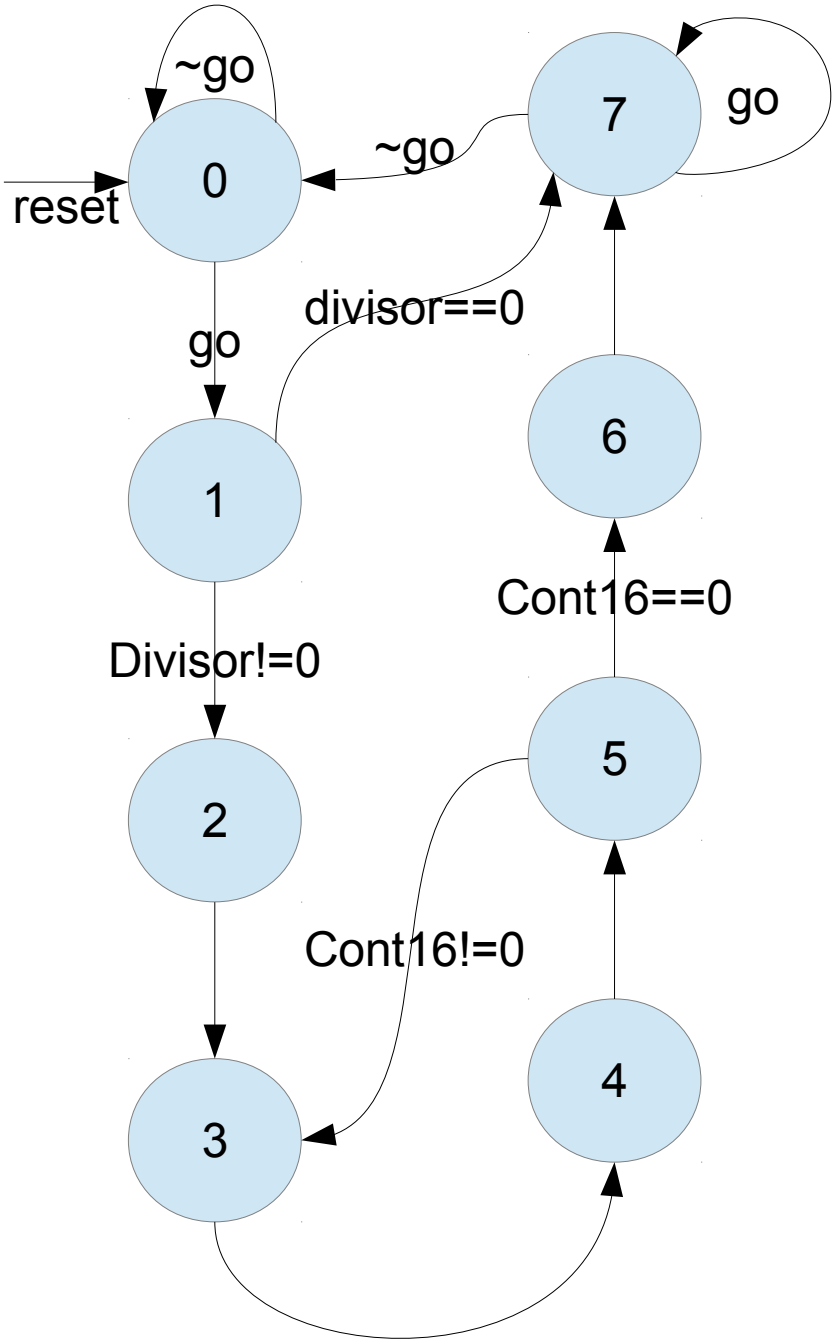

DvInput
DdInput
go
Comp 2
Comp 2
<<1
A B
A-B
A B
A+B
go
divisor
dividend
Control
negDividend
negDivisor
DivisorNoCero
inc
<<1
Comp 2
inc
Cont16
quotient
done
quotient
done




```
ESTADO_0: begin
        done <= 1;
        if (go) begin
                divisor <= dvInput;
                dividend <= ddInput;
                quotient <= 0;
                ProximoEst <= ESTADO_1;
        end
        else
                ProximoEst <= ESTADO_0;
end
```

```
ESTADO_1: begin
        done <= 0;
        if (divisor) begin
                negDivisor <= divisor[`DvLen];
                negDividend <= dividend[`DdLen];
                ProximoEst <= ESTADO_2;
        end
        else
                ProximoEst <= ESTADO_7;
end
```

```
ESTADO_2: begin
        if (negDivisor)
                divisor <= - divisor;
        if (negDividend)
                dividend <= - dividend;
        Cont16 <= 0;
        ProximoEst <= ESTADO_3;
end
```

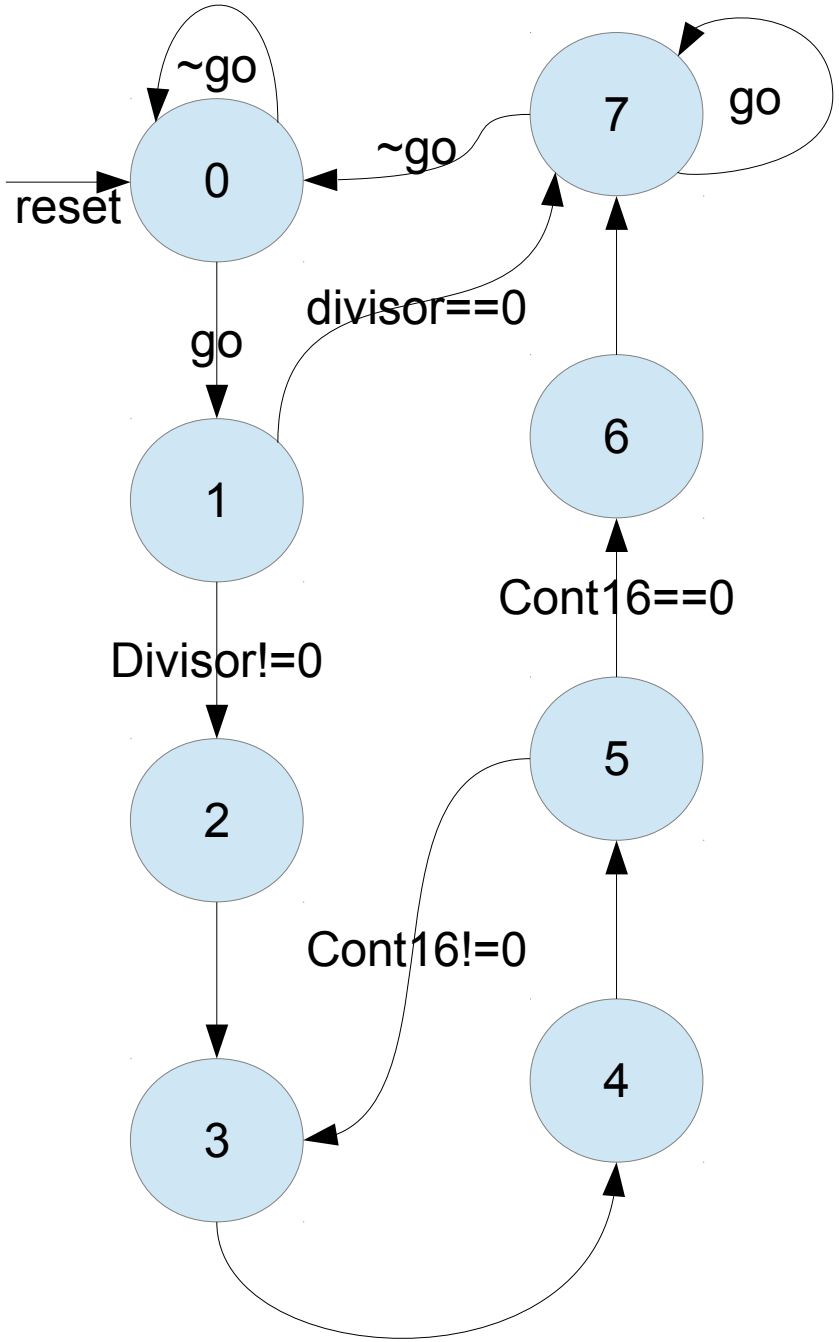



```
ESTADO_3: begin
        quotient <= quotient << 1;
        dividend <= dividend << 1;
        Cont16 <= Cont16 - 1;
        ProximoEst <= ESTADO_4;
end
```

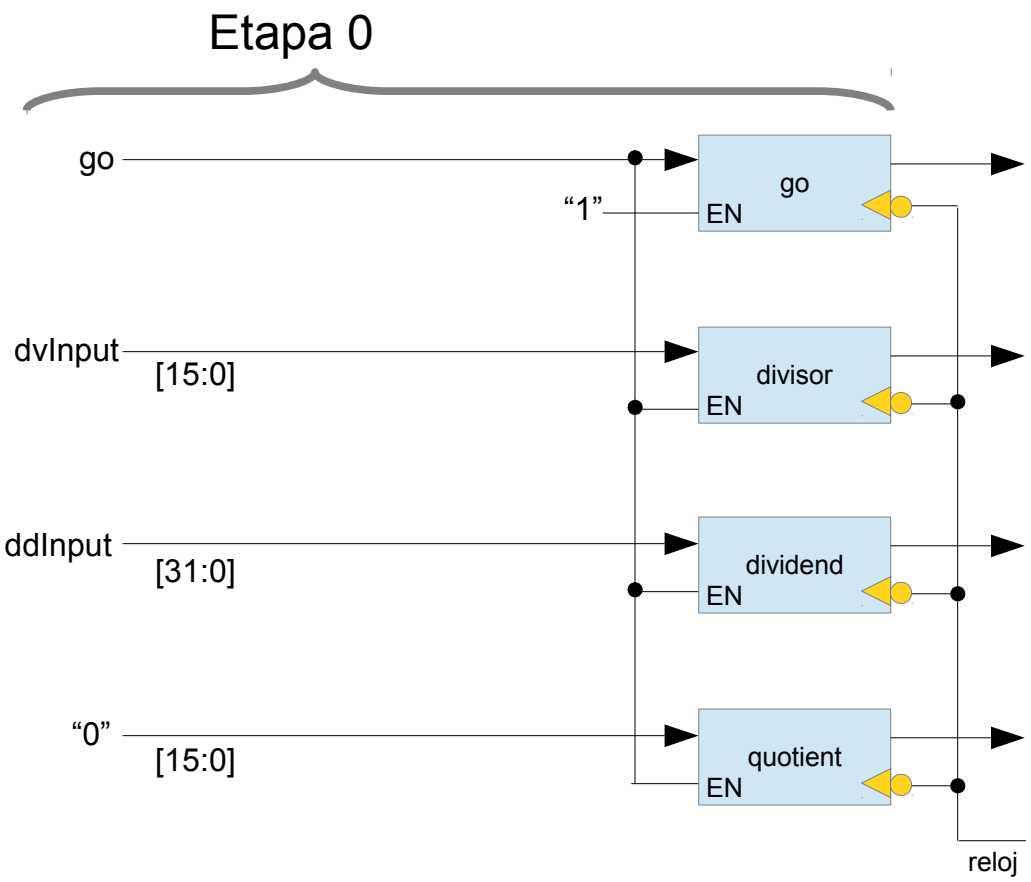
```
ESTADO_4: begin
        dividend[`DdLen:`HiDdMin] <=
            dividend[`DdLen:`HiDdMin] - divisor;
        ProximoEst <= ESTADO_5;
end
```

```
ESTADO_5: begin
  if (! dividend [`DdLen])
    quotient <= quotient + 1;
  else
    dividend[`DdLen:`HiDdMin] <=
      dividend[`DdLen:`HiDdMin] + divisor;
  if (Cont16)
    ProximoEst <= ESTADO_3;
  else
    ProximoEst <= ESTADO_6;
end
```

```
ESTADO_6: begin
  if (negDivisor != negDividend)
    quotient <= - quotient;
  ProximoEst <= ESTADO_7;
end
```




```
ESTADO_7: begin
  if (go)
    ProximoEst <= ESTADO_7;
  else
    ProximoEst <= ESTADO_0;
  end
```

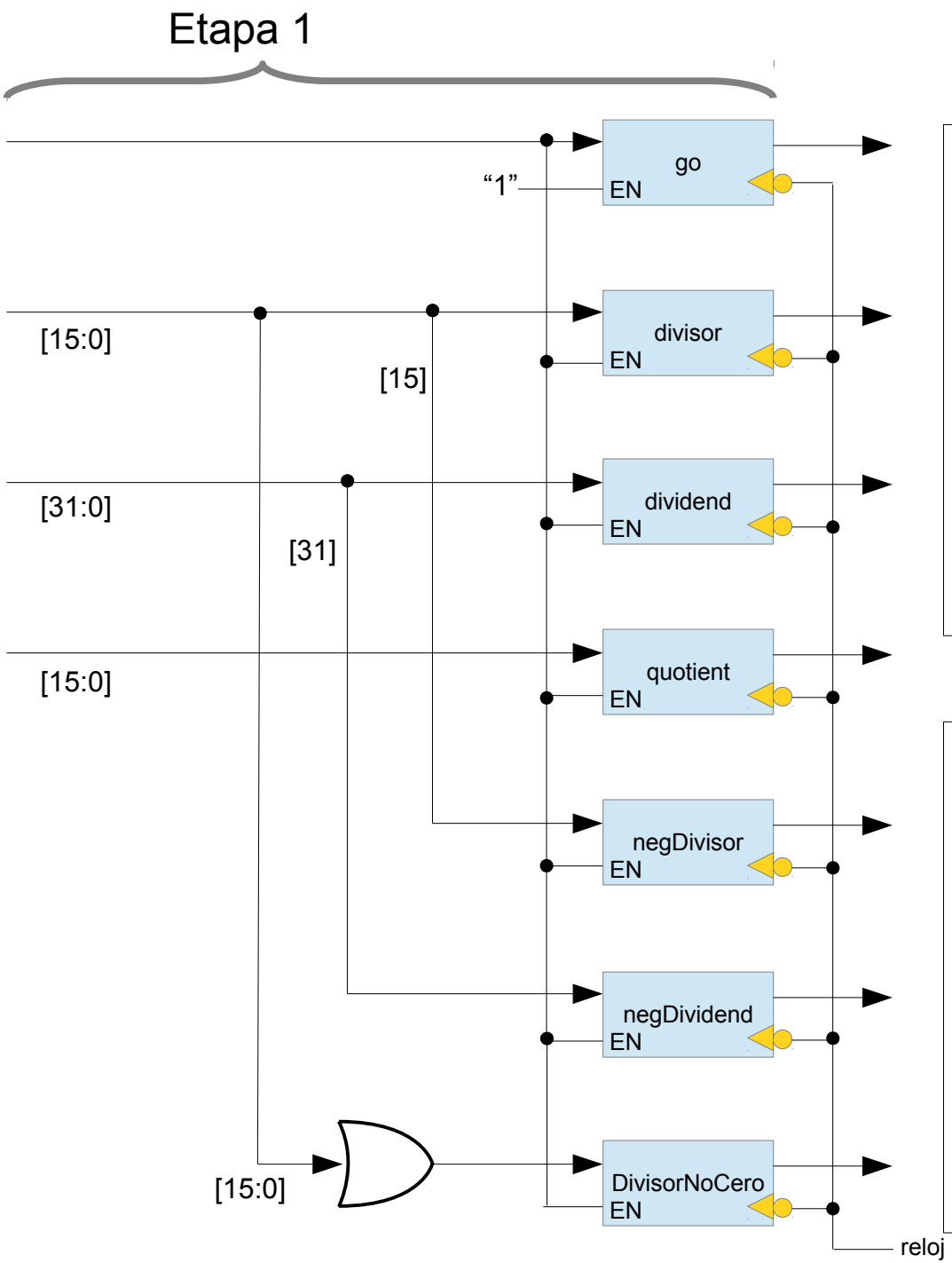



```
ESTADO_0: begin
        done <= 1;
        if (go) begin
                divisor <= dvInput;
                dividend <= ddInput;
                quotient <= 0;
                ProximoEst <= ESTADO_1;
        end
        else
                ProximoEst <= ESTADO_0;
end
```

Notas:

(1) El "done" no se va a usar pues el pipeline siempre está listo para recibir datos en cada flanco negativo del reloj.

(2) El "go" se utiliza para validar las entradas "dvInput" y "ddInput". Es decir, el "go" indica que los datos en esas dos entradas son válidos.

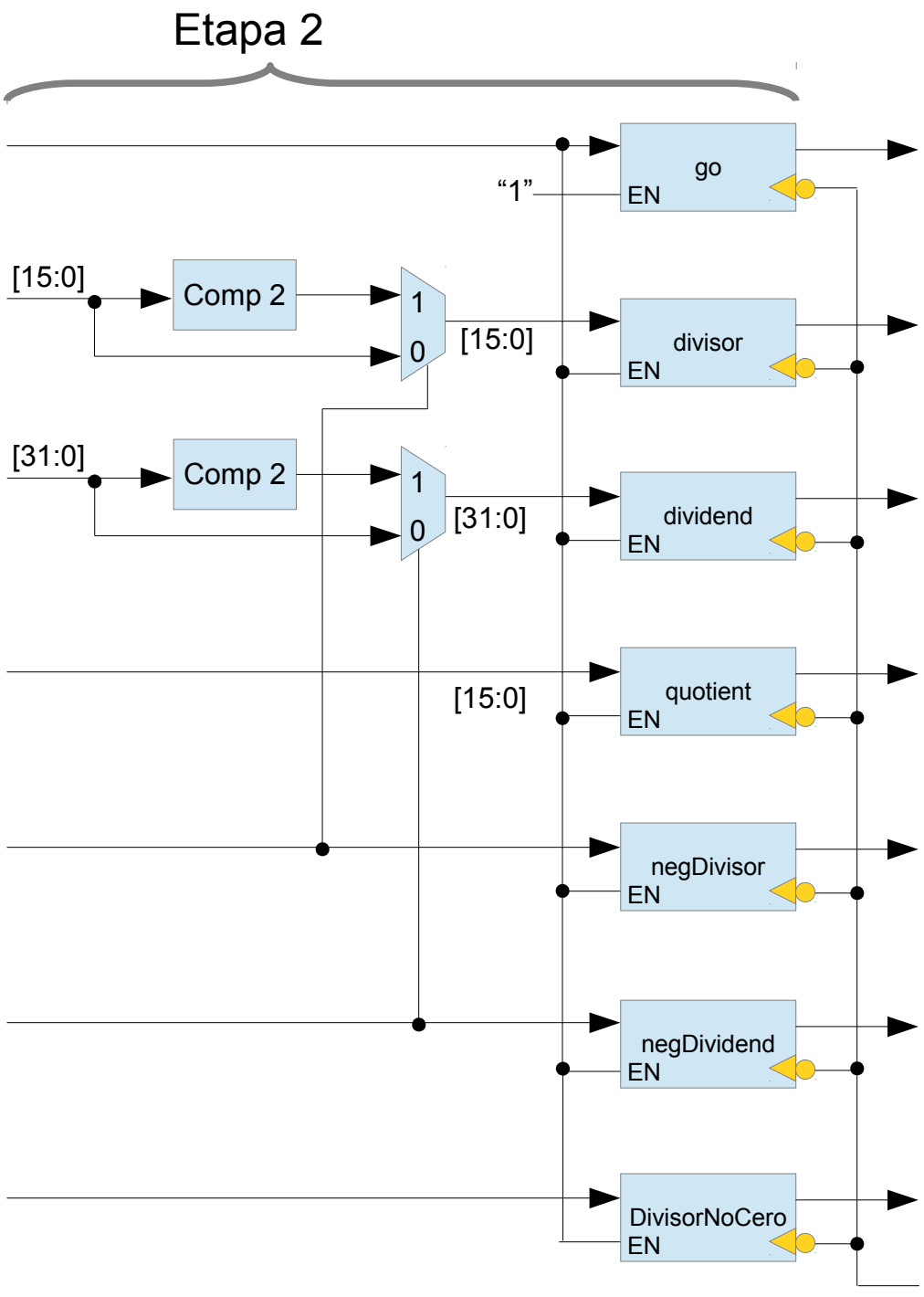



```
ESTADO_1: begin
        done <= 0;
        if (divisor) begin
                negDivisor <= divisor[`DvLen];
                negDividend <= dividend[`DdLen];
                ProximoEst <= ESTADO_2;
        end
        else
                ProximoEst <= ESTADO_7;
end
```

Notas:

(1) El "done" no se va a usar pues el pipeline siempre está listo para recibir datos en cada flanco negativo del reloj.

(2) La condición (divisor) se usa mayormente para evitar procesar una división entre cero. En el caso del pipeline, no se salta a ningún paso posterior, sino que se indica que se tiene una condición de división entre cero negando la señal "DivisorNoCero".

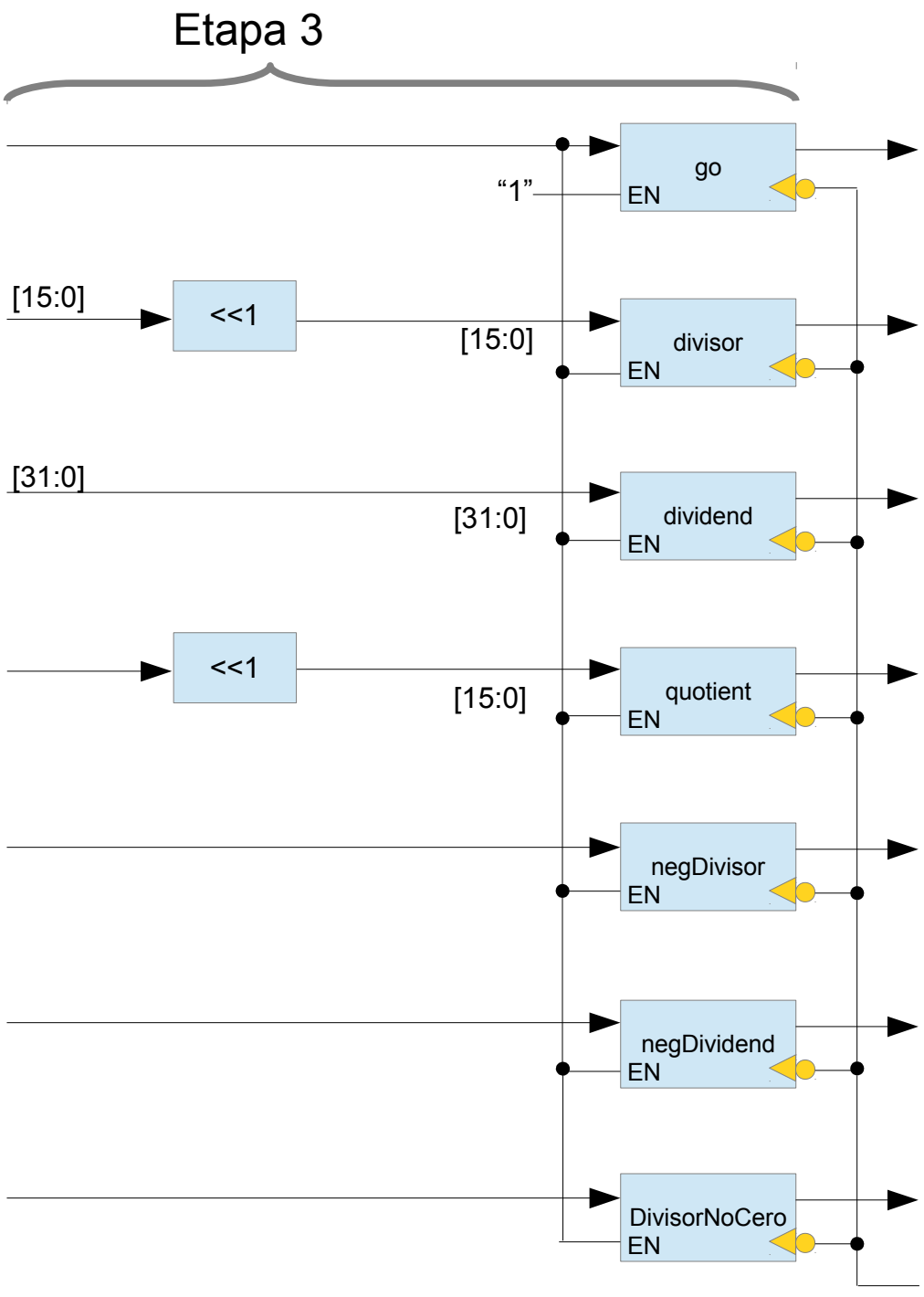



```
ESTADO_2: begin

        if (negDivisor)
                divisor <= - divisor;

        if (negDividend)
                dividend <= - dividend;

        Cont16 <= 0;

        ProximoEst <= ESTADO_3;
end
```

Notas:

(1) Las condiciones "negDivisor" y "negDividend" se usan para seleccionar si el contenido del los registros "divisor" y "dividend" se complementan a 2 para que ambos sean positivos.

(2) No se incluye un registro "Cont16" pues las iteraciones se harán repitiendo un número de etapas en el "pipeline" igual al número de iteraciones que tiene el algoritmo. Esto implica que se duplicaría la circuitería del diseño según el número de iteraciones.

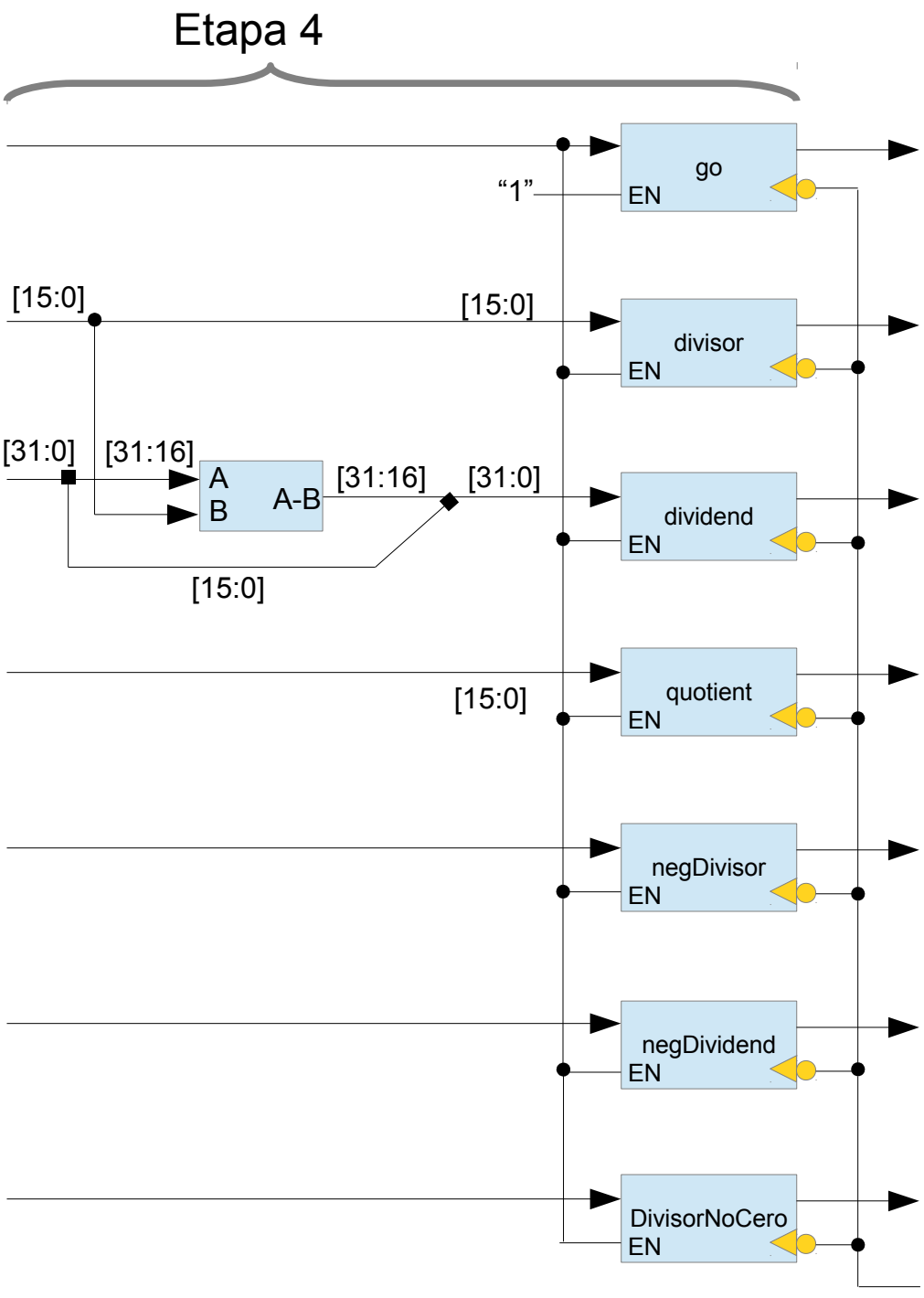



```
ESTADO_3: begin
        quotient <= quotient << 1;
        dividend <= dividend << 1;
        Cont16 <= Cont16 - 1;
        ProximoEst <= ESTADO_4;
end
```

Notas:

(1) Únicamente es necesario desplazar hacia la izquierda (<<1) los contenidos de los registros "quotient" y "dividend". Se introduce un "0" al bit menos significativo de ambos registros en el desplazamiento.

(2) La referencia al registro "Cont16" no es necesaria como se explicó anteriormente.

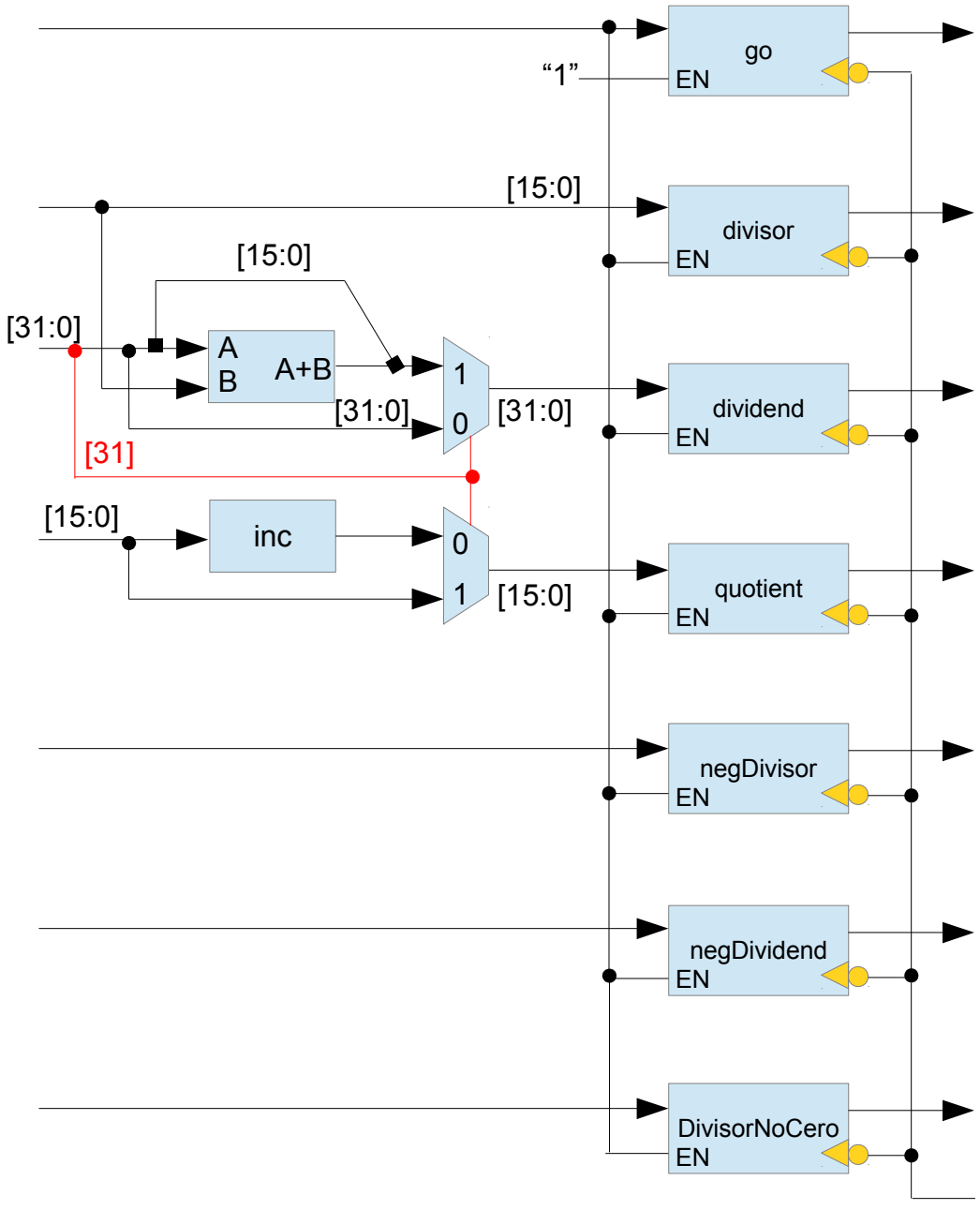



```
ESTADO_4: begin

        dividend[`DdLen:`HiDdMin] <=
            dividend[`DdLen:`HiDdMin] - divisor;

        ProximoEst <= ESTADO_5;
end
```

Notas:

(1) A la palabra superior del registro "dividend", es decir, de los bits [31:16], se le resta el contenido del registro "divisor".

Etapa 5

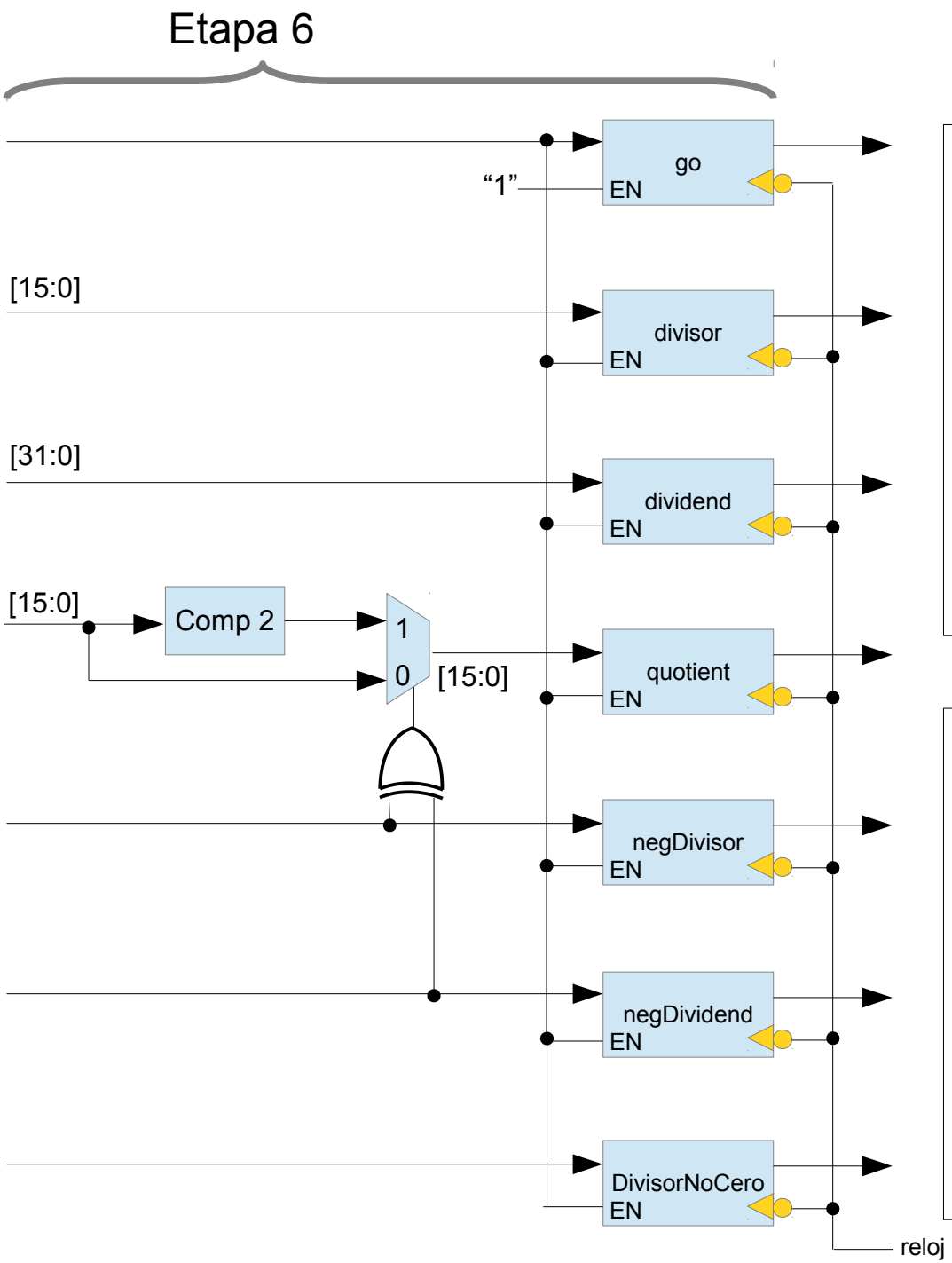



```
ESTADO_5: begin
  if (! dividend [`DdLen])
    quotient <= quotient + 1;
  else
    dividend[`DdLen:`HiDdMin] <=
      dividend[`DdLen:`HiDdMin] + divisor;

  if (Cont16)
    ProximoEst <= ESTADO_3;
  else
    ProximoEst <= ESTADO_6;
end
```

Notas:

(1) La condición "! dividend [`DdLen]" en verdadero indica que de la resta que se hizo de "dividend" en la etapa anterior, no se generó un llevo y por lo tanto se puede incrementar "quotient" para contabilizar la resta. De lo contrario se debe restituir el valor anterior de "dividend" y no se incrementa "quotient".

(2) La condición de "Cont15" distinto de cero aquí no es relevante pues las iteraciones del lazo se hacen repitiendo las etapas 3, 4, y 5 del pipeline 16 veces.




```
ESTADO_6: begin

  if (negDivisor != negDividend)
    quotient <= - quotient;

  ProximoEst <= ESTADO_7;

end
```

Notas:

(1) Si los signos del divisor y el dividendo difieren, entonces el cociente es negativo. De lo contrario es positivo.

(2) Esta es la última etapa del pipeline.