

Operadores de asignación en Verilog

`assign` ('continuous assignment')

- Sintaxis

```
assign variable = [delay] expression;
```
- Consiste en la asignación continua de un valor a una señal, usualmente tipo wire o similar
- Toma precedencia sobre otras asignaciones
- No se puede utilizar con señales con memoria (ej. *reg*), debido a que estas no necesitan asignación continua
- Se puede realizar de manera *explícita*, incluyendo la palabra clave 'assign', o de manera implícita al momento de declarar una señal:

```
wire a = x & y; //equivale a hacer assign a = x & y
```

`=` ('blocking procedural assignment')

- Sintaxis

```
variable = [delay] expression;
```
- La expresión se evalúa en el momento en el que se encuentra, y se asigna su valor a la variable en ese mismo momento
- En un bloque *begin-end* secuencial, la ejecución de los statements subsiguientes se detiene hasta que se complete la asignación
- No se puede utilizar con señales que no soportan memoria (ej. *wire*)


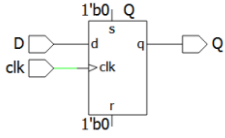
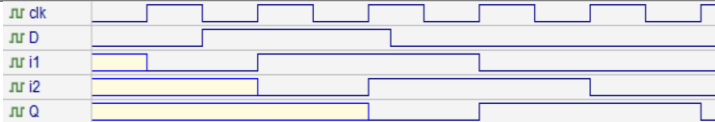
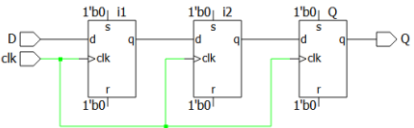
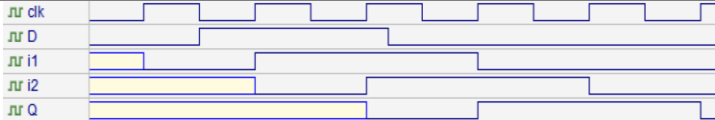
`<=` ('non-blocking procedural assignment')

- Sintaxis

```
variable <= [delay] expression;
```
- La expresión se evalúa en el momento en el que se encuentra, pero se asigna su valor al final del step de tiempo correspondiente
- En un bloque *begin-end* secuencial, la ejecución de los statements subsiguientes no se detiene, y estos se evalúan antes que se complete la asignación
- No se puede utilizar con señales que no soportan memoria (ej. *wire*)

Comparación de operadores = y <=

Considérense los siguientes bloques *always*, con su correspondiente salida de simulación:

Verilog	Resultado	Implementación
<pre>always @(posedge clk) begin i1 = D; i2 = i1; Q = i2; end</pre>	 <p>Se puede ver claramente que las señales se asignan en el momento en que se encuentra el =</p>	
<pre>always @(posedge clk) begin i1 <= D; i2 <= i1; Q <= i2; end</pre>	 <p>El valor se evalúa donde se encuentra el <=, pero la asignación a las señales ocurre al final del bloque begin-end</p>	
<pre>always @(posedge clk) begin Q = i2; i2 = i1; i1 = D; end</pre>	 <p>Igual que en el primer caso, la asignación ocurre en donde se encuentra el =, pero debido a que el orden esta invertido, conseguimos un resultado distinto</p>	