Verilog Assignments

Continuous assignment

Features

1. It is used outside 'always' statements, 'initial' block.
2. primarily used by 'assign" keyword.
3. Left Hand Side should be declared as wires.
4. " = " sign is used.

e.g. ~~Sum = a ^ b~~

```
assign sum = a ^ b;
```

Procedural assignment

1. It is used inside always blocks & other procedural blocks.
2. Two types of procedural assignments.

(i) Blocking Assignment

(ii) Non-Blocking Assignment.

Use of procedural Assignments.

- Combinational logic → Generally use blocking Assignments.
- Sequential logic → Generally use Non-blocking Assignments.

Blocking Assignments.

- This type of assignment must complete before the next line is executed.
- Blocks the flow of the program

i.e. "Execution flow within the procedure is blocked until the assignment is complete.

- Operator is '='.

The following examples would try to swap the bytes but it won't ~~do~~ work because of blocking assignment.

```
always @(posedge clk)
begin
word [15:8] = word [7:0];
word [7:0] = word [15:8];
end
```

or,

```
always @ (posedge clk) begin
word [7:0] = word [15:8];
word [15:8] = word [7:0];
end
```

→ Both can't swap the bytes.

Non-Blocking Assignments.

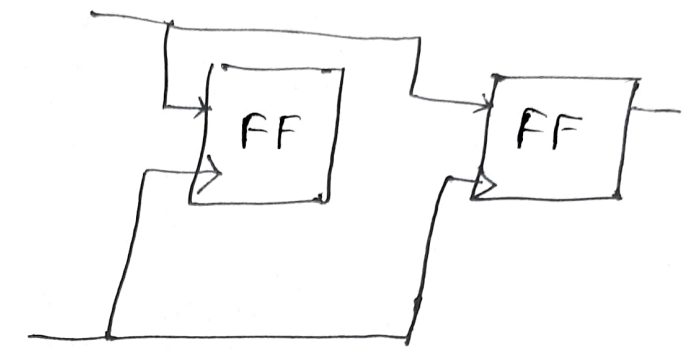
- These are evaluated and assigned in two steps:
  - ⇒ The right hand side (RHS) is evaluated immediately
  - ⇒ The assignment to the left-hand side (LHS) is postponed until other evaluations in the current time step are completed.
- Execution flow within the procedure continues. until a timing control is encountered (flow is not blocked).
- Operator is '<='
- Both the following will swap the upper & lower bytes.

```
always @(posedge clk)
begin
word [15:8] <= word [7:0];
word [7:0] <= word [15:8];
end
```

```
always @(posedge clk)
begin
word [7:0] <= word [15:8];
word [15:8] <= word [7:0];
end
```

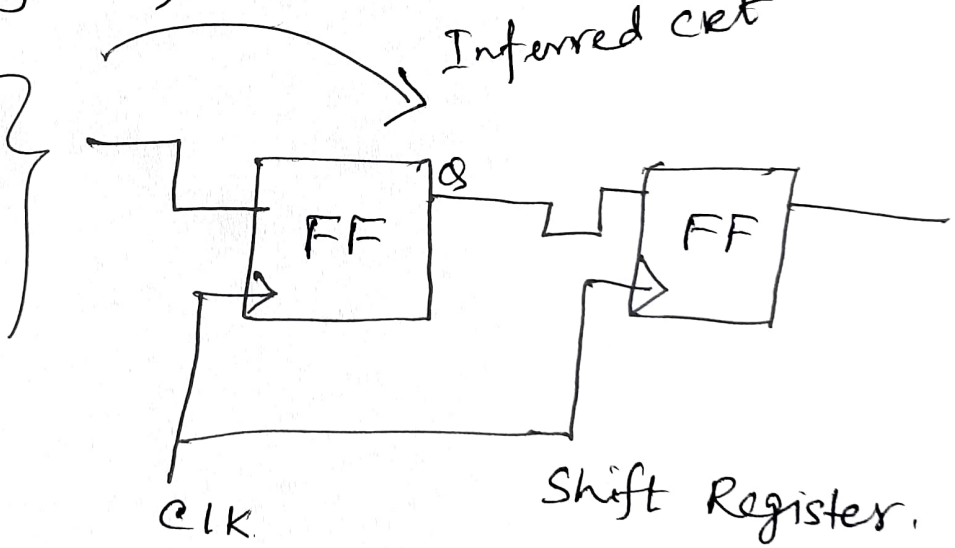
Implications of Sequential Procedural Assignments.

```
always @(posedge clk).
begin
    y1 = in;
    y2 = y1;
end.
```



It would generate Parallel Flops

```
always @ (posedge clk)
begin
    y1 <= in;
    y2 <= y1;
end.
```



Shift Register.