Behavioural modelling represents digital circuits at a functional and algorithmic level. It is used mostly to describe sequential circuits, but can also be used to describe combinational circuits.

```verilog
// Behavioral description of two-to-one-line multiplexer

module mux_2x1_beh (m_out, A, B, select);
output reg m_out;
input A, B, select;

always @(A or B or select)
  if (select == 1) m_out = A;
  else m_out = B;
endmodule
```

Behavioural descriptions use the keyword always, followed by an optional event control expression and a list of procedural assignment statements.

The target output of a procedural assignment statement must be of the reg data type. Contrary to the wire data type, whereby the target output of an assignment may be continuously updated, a reg data type retains its value until a new value is assigned.
(a) Logic diagram

(b) Block diagram

MUX

$Y$

$I_0$

$I_1$

$S$

$0$

$1$
Behaviour declared by the keyword `initial` is called single-pass behaviour and specifies a single statement or a block statement (i.e., a list of statements enclosed by either a

```
begin . . . end
```

or a

```
fork . . . join
```

keyword pair).

A single-pass behaviour expires after the associated statement executes.

In practice, designers use single-pass behaviour primarily to prescribe stimulus signals in a test bench, never to model the behaviour of a circuit, because synthesis tools do not accept descriptions that use the initial statement.
The always keyword declares a cyclic behaviour.

Both types of behaviours begin executing when the simulator launches at time $t = 0$.

The initial behaviour expires after its statement executes; the always behaviour executes and re-executes indefinitely, until the simulation is stopped.

A module may contain an arbitrary number of initial or always behavioural statements.

They execute concurrently with respect to each other, starting at time $0$, and may interact through common variables.
// Behavioral description of four-to-one line multiplexer

// Verilog 2001, 2005 port syntax

module mux_4x1_beh
(output reg m_out,
 input   in_0, in_1, in_2, in_3,
 input   [1: 0] select
);

always @ (in_0, in_1, in_2, in_3, select)      // Verilog 2001, 2005 syntax
  case (select)
    2'b00:   m_out = in_0;
    2'b01:   m_out = in_1;
    2'b10:   m_out = in_2;
    2'b11:   m_out = in_3;
  endcase
endmodule