```
module StateMachine(
    input Left,
    input Right,
    input clk,
    output RunGame,
    output Add,
    output Subtract
    );
    wire left, right, INIT_PRESS;
    wire IDLE, LEFT_BREAK, RIGHT_BREAK, ALL_BREAK;
    wire Next_IDLE, Next_LEFT_BREAK, Next_RIGHT_BREAK,
Next_ALL_BREAK, out;
    FDRE #(.INIT(1'b0)) sync1 (.C(clk), .R(1'b0),
.CE(1'b1), .D(Left), .Q(left));
    FDRE #(.INIT(1'b0)) sync2 (.C(clk), .R(1'b0),
.CE(1'b1), .D(Right), .Q(right));

    Edge_Detector btn_press (.clk(clk), .btn(!IDLE),
.out(out));
    FDRE #(.INIT(1'b0)) init_btn (.C(clk), .R(1'b0),
.CE(out), .D(Right), .Q(INIT_PRESS)); //1 if INIT
right, 0 if left



    wire IDLE, LEFT_BREAK, RIGHT_BREAK, ALL_BREAK;
    wire Next_IDLE, Next_LEFT_BREAK, Next_RIGHT_BREAK,
Next_ALL_BREAK;

    assign Next_IDLE = (IDLE & !left & !right) |
(LEFT_BREAK & !left & !right) |
                       (RIGHT_BREAK & !left & !right) |
```

```
(ALL_BREAK & !left & !right);
    FDRE #(.INIT(1'b1)) idle (.C(clk), .R(1'b0),
.CE(1'b1), .D(Next_IDLE), .Q(IDLE));

    assign Next_LEFT_BREAK = (LEFT_BREAK & left &
!right) | (IDLE & left & !right) |
                              (RIGHT_BREAK & left &
!right) | (ALL_BREAK & left & !right);
    FDRE #(.INIT(1'b0)) left_break (.C(clk), .R(1'b0),
.CE(1'b1), .D(Next_LEFT_BREAK), .Q(LEFT_BREAK));

    assign Next_RIGHT_BREAK = (LEFT_BREAK & !left &
right) | (IDLE & !left & right) |
                               (RIGHT_BREAK & !left &
right) | (ALL_BREAK & !left & right);
    FDRE #(.INIT(1'b0)) right_break (.C(clk), .R(1'b0),
.CE(1'b1), .D(Next_RIGHT_BREAK), .Q(RIGHT_BREAK));

    assign Next_ALL_BREAK = (LEFT_BREAK & left & right)
| (IDLE & left & right) |
                              (RIGHT_BREAK & left &
right) | (ALL_BREAK & left & right);
    FDRE #(.INIT(1'b0)) all_break (.C(clk), .R(1'b0),
.CE(1'b1), .D(Next_ALL_BREAK), .Q(ALL_BREAK));

    assign RunGame = !IDLE;
    assign Add = LEFT_BREAK & !left & !right &
INIT_PRESS;
    assign Subtract = RIGHT_BREAK & !left & !right &
!INIT_PRESS;
endmodule
```