Q2.

(a)

//Dataflow

```
module pg(input S, R, output Q, Qn);
assign Q = ~(Qn | R);
assign Qn = ~(Q | S);
endmodule
```

//gate

```
module pg(input S, R, output Q, Qn);
nor(Q, Qn, R);
nor(Qn, Q, S);
endmodule
```

//Behavioral

```
module pg(input s, r, output reg Q, Qn);
always @ * begin
if(s == 0 & r == 0) begin
Q = Q;
Qn = Qn;
end
else if(s == 1 & r == 0) begin
Q = 1;
Qn = 0;
end
else if(s == 0 & r == 1) begin
Q = 0;
Qn = 1;
end
end
endmodule
```

```
module pg_tb;
wire Q, Qn;
reg s, r;
pg inst0(s, r, Q, Qn);
initial begin
s=0;r=1;
#100 s=0;r=0;
#100 s=1;r=0;
#100 s=0;r=0;
#100 s=0;r=1;
#100 s=0;r=0;
#100 s=1;r=1;
#50 $stop;
end
endmodule
```

(b)

//Dataflow

```
module pg(input s, r, clk, output Q, Qn);
assign ns = ~(clk & s);
assign nr = ~(clk & r);
assign Q = ~(ns & Qn);
assign Qn = ~(nr & Q);
endmodule
```

//Gate

```
module pg(input s, r, clk, output Q, Qn);
nand(ns, clk, s);
nand(nr, clk, r);
nand(Q, ns, Qn);
nand(Qn, nr, Q);
endmodule
```

//Behavioral

```
module pg(input s, r, clk, output reg Q, Qn);
always @ * begin
if(~clk) begin
Q = Q;
Qn = Qn;
end
if(clk) begin
if(s == 0 & r == 0) begin
Q = Q;
Qn = Qn;
end
if(s == 0 & r == 1) begin
Q = 0;
Qn = 1;
end
if(s == 1 & r == 0) begin
Q = 1;
Qn = 0;
end
end
end
endmodule

module pg_tb;
wire Q, Qn;
reg s, r, clk;
pg inst0(s, r, clk, Q, Qn);
initial #200 $finish;
initial begin clk = 1;forever #5 clk = ~clk; end
initial begin
```

```
s=0;r=1;
#10 s=0;r=0;
#10 s=1;r=0;
#10 s=0;r=0;
#10 s=0;r=1;
#10 s=0;r=0;
#10 s=1;r=1;
#50 $stop;
end
endmodule
```