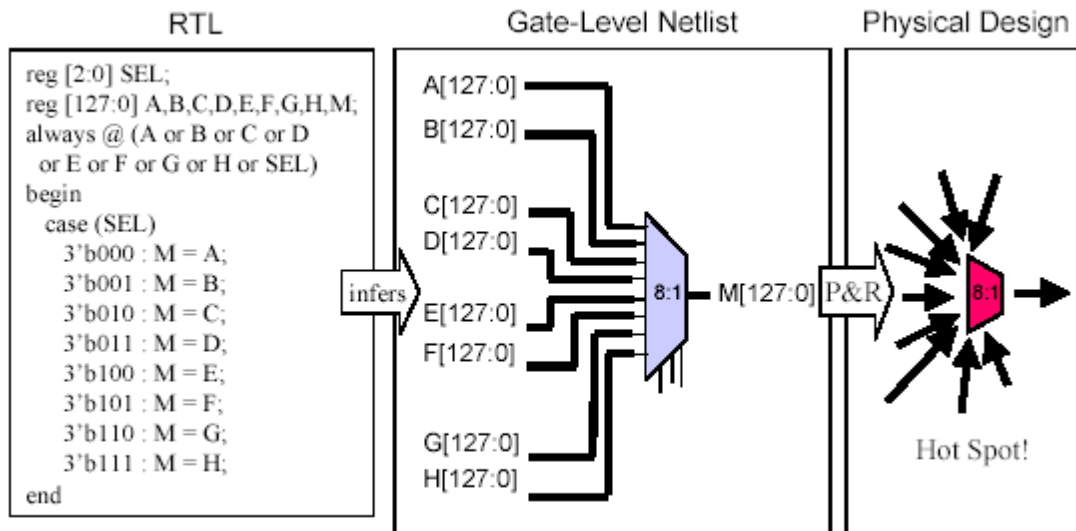


Dealing with design Hot Spots

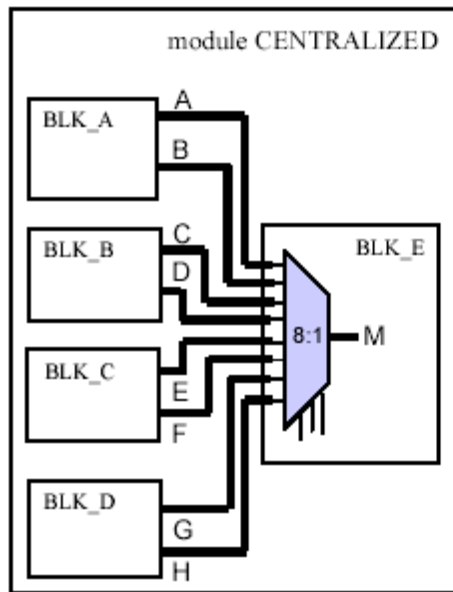
Functional areas of the design that have a high route per area requirement cause Hot Spots. Figure 1 shows a very popular manifestation of the Hot Spot problem in terms of the large mux. Hot spots do not show up in synthesis timing results. They are a function of routing resources and thus they will pass through synthesis without detection. The user has to be aware of the type of circuitry that is described by their RTL code. Proper analysis of the RTL implied routing requirements would detect possible congestion problems with the physical implementation.



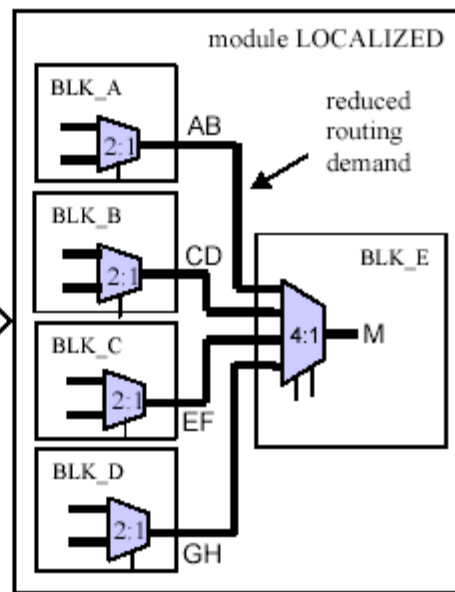
Hot spots do not show up in synthesis timing results. They are a function of routing resources and therefore will pass through synthesis without detection. The user has to be aware of the type of circuitry that is described by their RTL code. Proper analysis of the RTL implied routing requirements would detect possible congestion problems with the physical implementation.

The large mux hot spot has multiple solutions. The first solution utilizes physical hierarchy. Just de-generating a large mux into component parts is not a good solution. It only has the effect of spreading out the hot spot a little. You would end up with a hot spot that looks like a ring. This solution works because it utilizes hierarchy to spread out the routing requirements of the design without impacting the area.

Hot Spot - Centralized Mux



Solution - Localized Muxes



Distribute

Hot Spot - Centralized Mux

```

module CENTRALIZED(...)
...
BLK_A U1 (.CLK(CLK), .RST(RST), .Q1(A), Q2(B));
BLK_B U1 (.CLK(CLK), .RST(RST), .Q1(A), Q2(B));
BLK_C U1 (.CLK(CLK), .RST(RST), .Q1(A), Q2(B));
BLK_D U1 (.CLK(CLK), .RST(RST), .Q1(A), Q2(B));
BLK_E U5 (.I1(A), .I2(B), .I3(C), .I4(D), .I5(E), .I6(F),
.I7(G), .I8(H), .M(Q), .SEL(S), .CLK(CLK), .RST(RST));
endmodule

module BLK_E (...
always@(I1 or I2 or I3 or I4 or I5 or I6 or I7 or I8 or SEL)
case (SEL)
3'b000 : M = I1;
3'b001 : M = I2;
3'b010 : M = I3;
3'b011 : M = I4;
3'b100 : M = I5;
3'b101 : M = I6;
3'b110 : M = I7;
3'b111 : M = I8;
endcase
endmodule
...

```

Solution - Localized Muxes

```

module LOCALIZED(...)
...
wire S0 = S[0];
wire [1:0] S21 = S[2:1];
BLK_A U1 (.CLK(CLK), .RST(RST), .S(S0), Q(AB));
BLK_B U1 (.CLK(CLK), .RST(RST), .S(S0), Q(CD));
BLK_C U1 (.CLK(CLK), .RST(RST), .S(S0), Q(EF));
BLK_D U1 (.CLK(CLK), .RST(RST), .S(S0), Q(GH));
BLK_E U5 (.I1(AB), .I2(CD), .I3(EF), .I4(GH), .M(Q),
.SEL(S21), .CLK(CLK), .RST(RST));
endmodule

module BLK_E (...
always@(I1 or I2 or I3 or I4 or SEL)
case (SEL)
2'b00 : M = I1;
2'b01 : M = I2;
2'b10 : M = I3;
2'b11 : M = I4;
endcase
endmodule

module BLK_A (...
always@(A or B or S)
case (S)
1'b0 : Q = A;
1'b1 : Q = B;
endcase // could have used an if-then-else statement
endmodule
...

```

NOTE: The RTL coding examples are not complete on purpose. They are just meant to show the highlights of the differences between the RTL code examples. The function statements represent combinational logic.