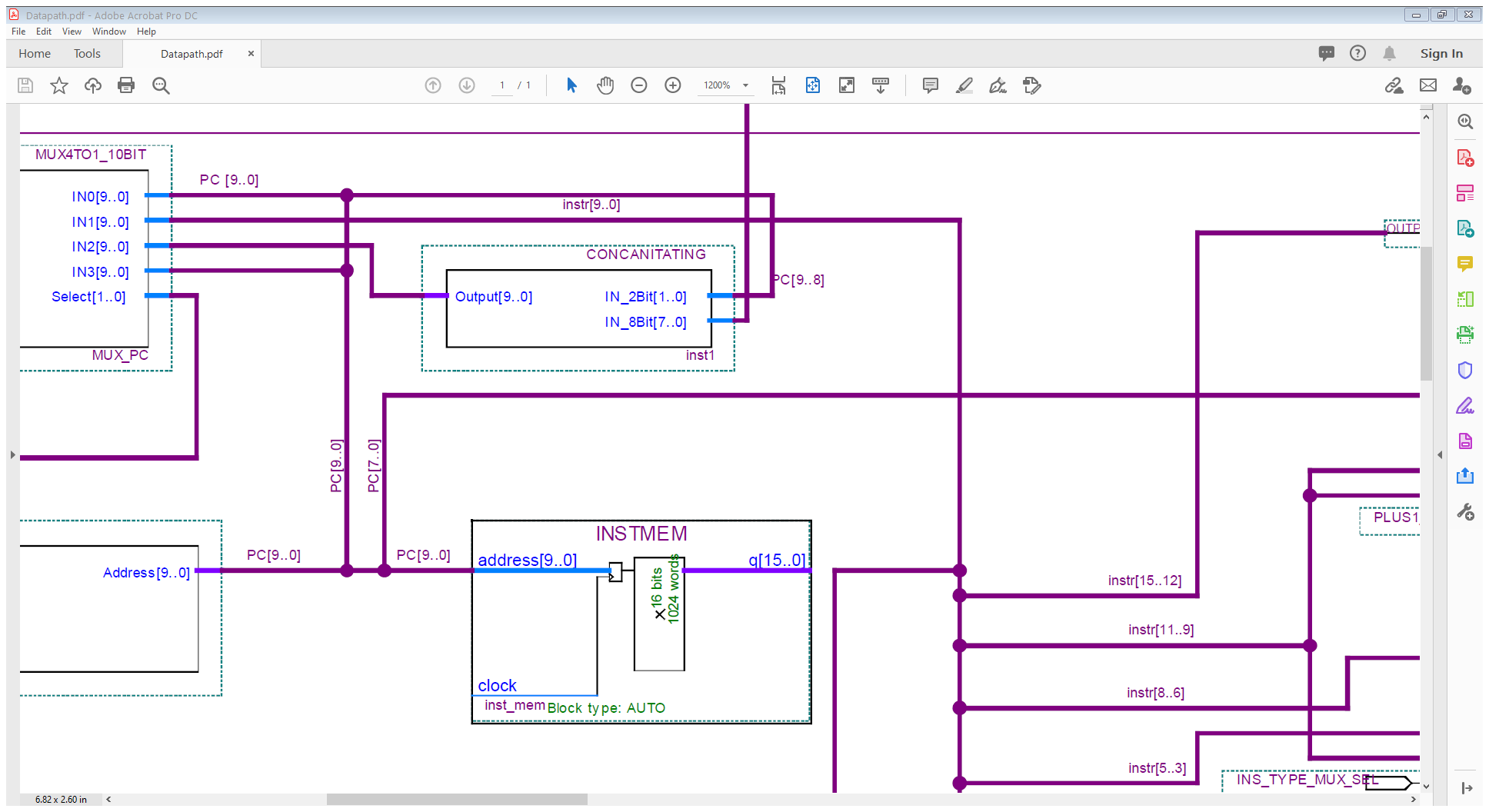
# New sequential and combinational blocks

## CONCANITATING



module CONCANITATING(IN\_2Bit, IN\_8Bit, Output);

input [1:0] IN\_2Bit;

input [7:0] IN\_8Bit;

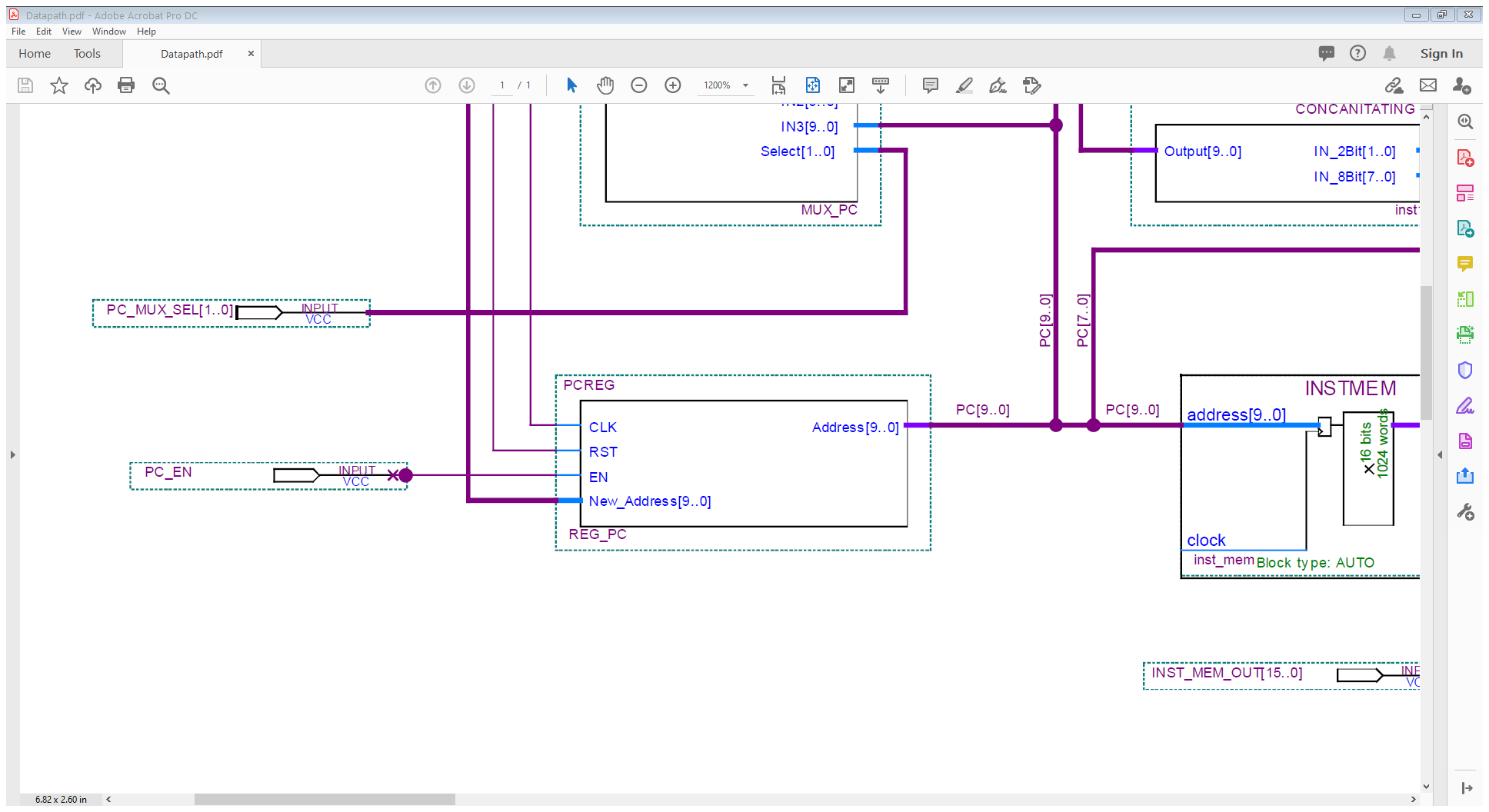
output reg [9:0] Output;

always @ \*

Output <= {IN\_2Bit[1:0], IN\_8Bit[7:0]};

Endmodule

## PCREG



module PCREG(CLK, RST, EN, New\_Address, Address);

input CLK, EN, RST;

input [9:0] New\_Address;

output reg [9:0] Address;

always @(posedge CLK)

begin

if (RST)

Address <= 10'b0;

else

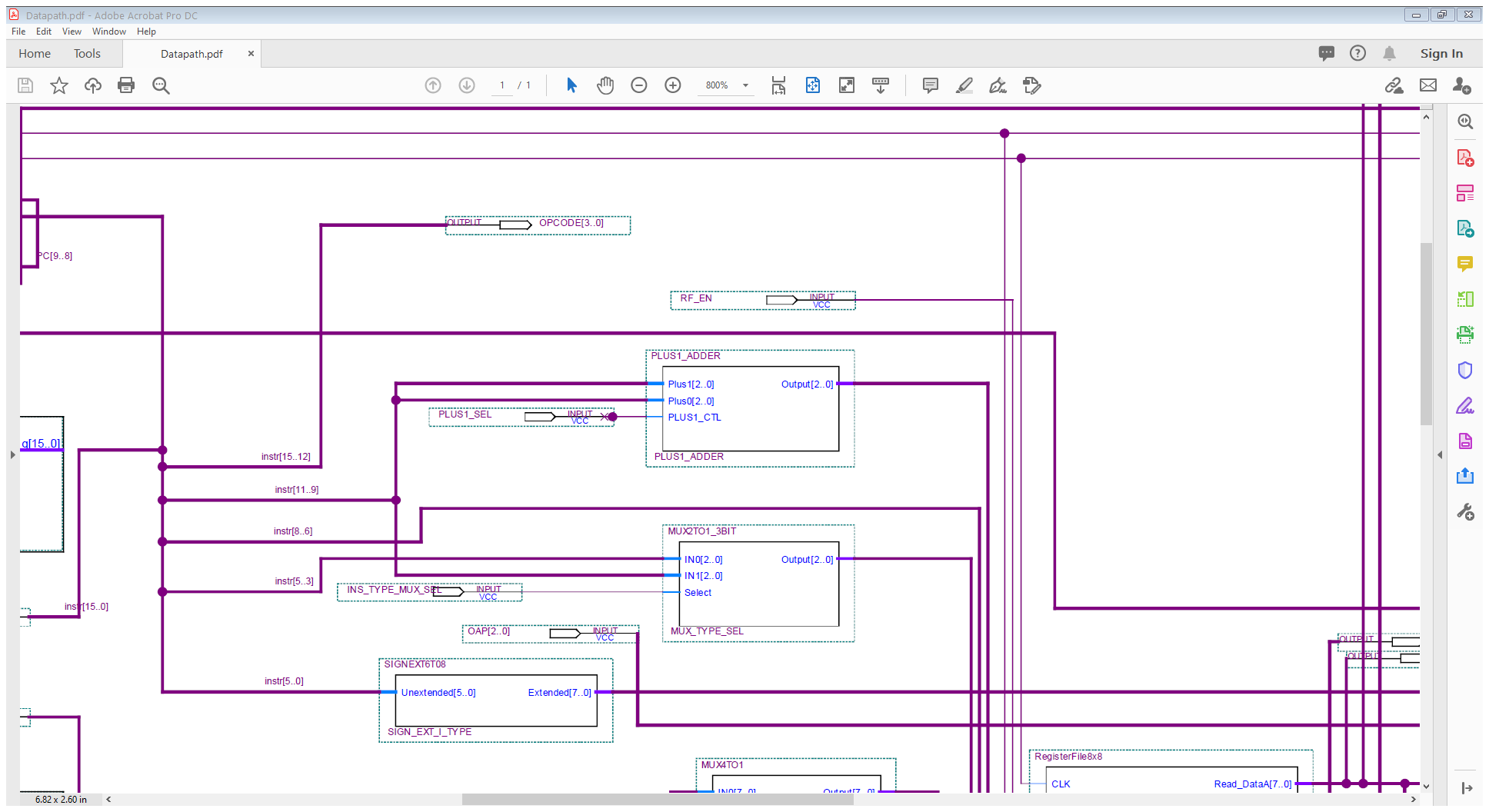
if (EN)

Address <= New\_Address + 1;

end

endmodule

## PLUS1\_ADDER



module PLUS1\_ADDER(

input [2:0] Plus1,

input [2:0] Plus0,

input PLUS1\_CTL,

output reg [2:0] Output

);

always@ \*

begin

case (PLUS1\_CTL)

1'b0 : Output = Plus0;

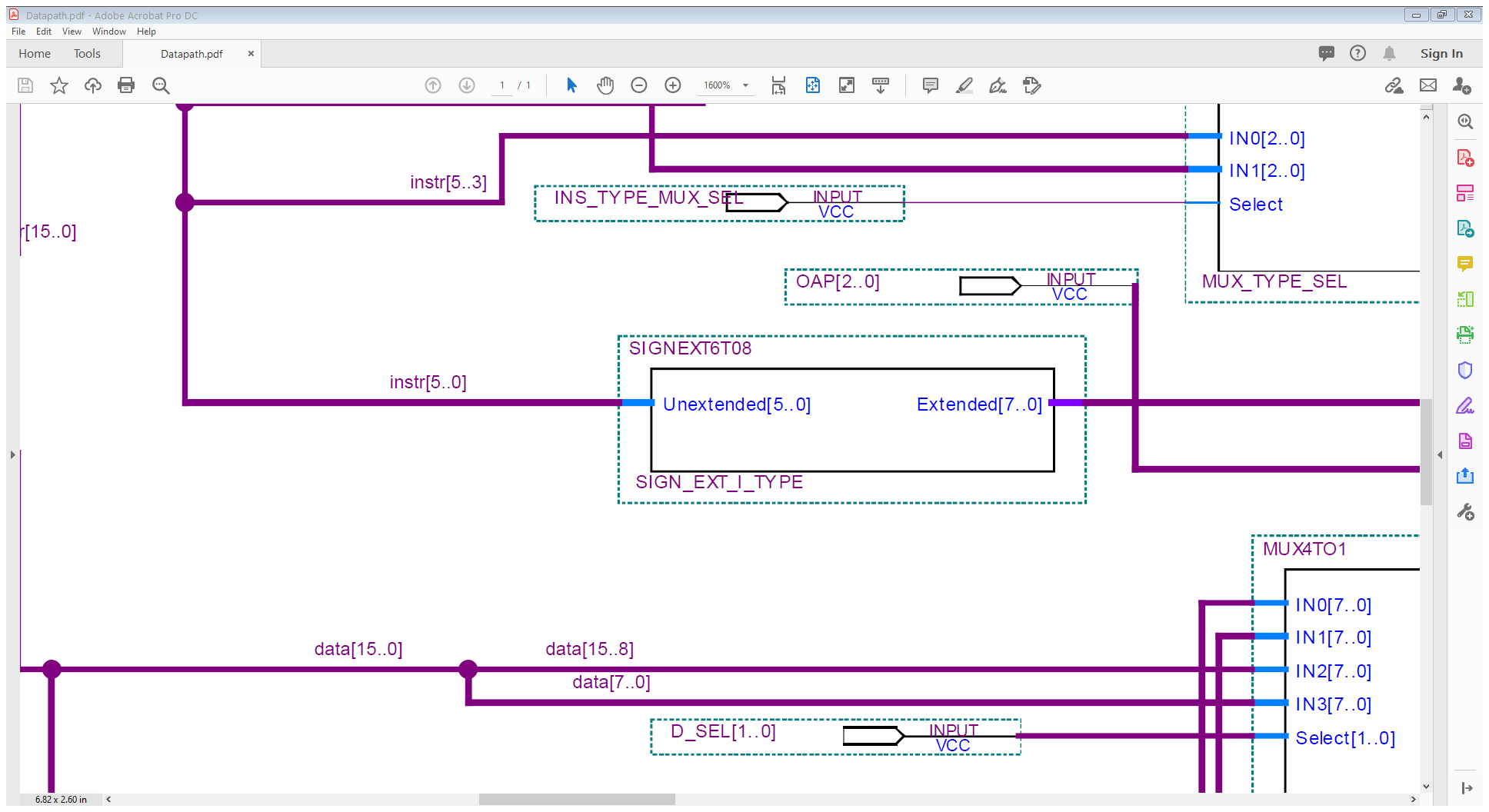
1'b1 : Output = Plus1 + 1;

endcase

end

endmodule

## SIGN\_EXTENSION



module SIGNEXT6T08(

input [5:0] Unextended,

output reg [7:0] Extended

);

always@ \*

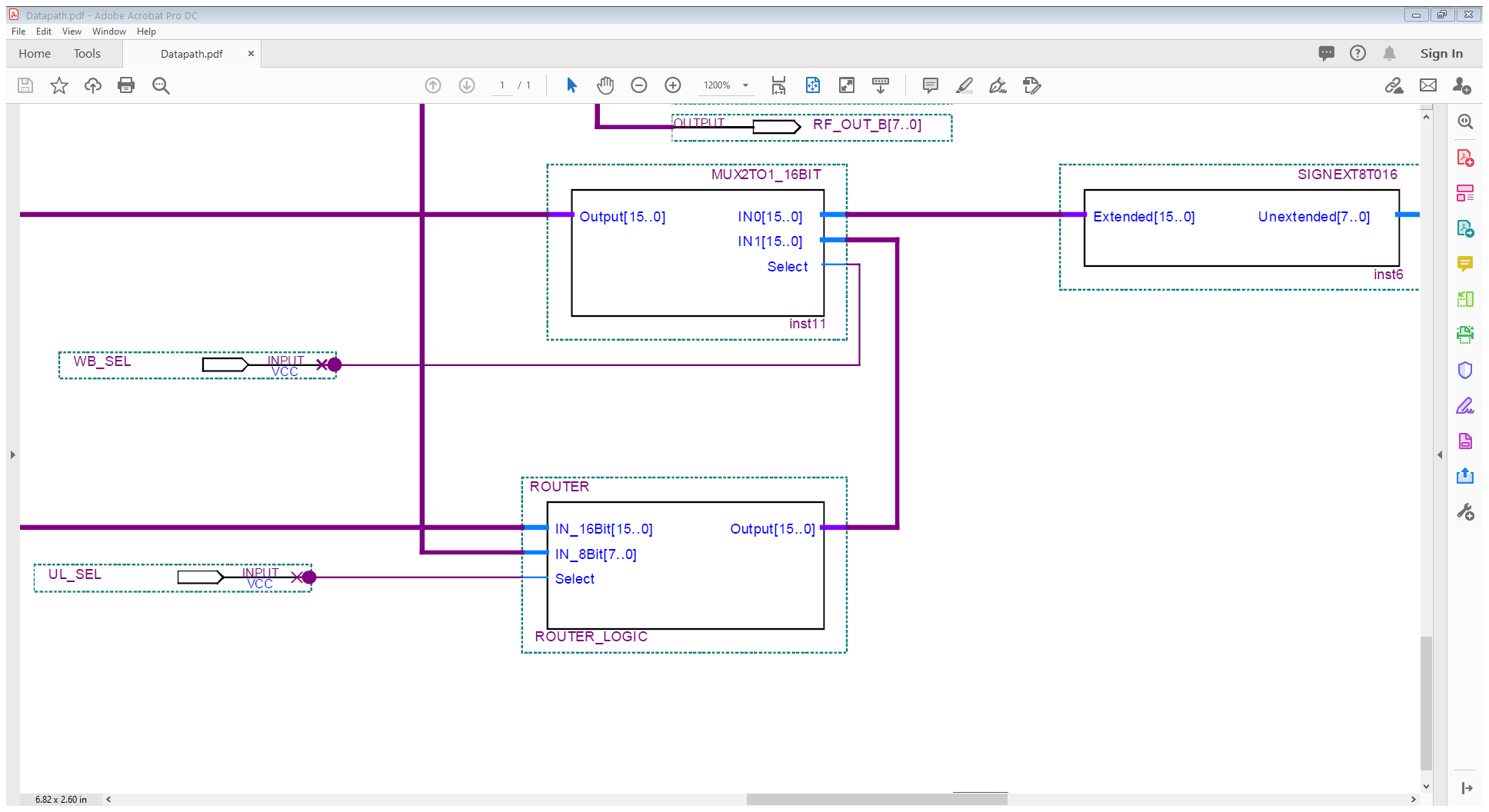
begin

Extended[7:0] <= { {2{Unextended[5]}}, Unextended[5:0] };

end

endmodule

## ROUTER



module ROUTER(IN\_16Bit, IN\_8Bit, Select, Output);

input [15:0] IN\_16Bit;

input [7:0] IN\_8Bit;

input Select ;

output reg [15:0] Output;

always @ \*

case (Select)

1'b0 : Output <= {IN\_16Bit[15:8], IN\_8Bit[7:0]};

1'b1 : Output <= {IN\_8Bit[7:0], IN\_16Bit[7:0]};

endcase

endmodule