4Bit Vedic-Wallace Tree Code

```
// Half Adder
module ha(sum, carry, a, b);
  input a, b;
  output sum, carry;
  assign sum = a ^ b;
  assign carry = a & b;
endmodule

// Full Adder
module fa(sum, carry, a, b, cin);
  input a, b, cin;
  output sum, carry;

  assign sum = a ^ b ^ cin;
  assign carry = (a & b) | (b & cin) | (a & cin);
endmodule

// Wallace Tree Multiplier
module wallace(prod, a, b);
  input [3:0] a, b;
  output [7:0] prod;

  wire [3:0] p0, p1, p2, p3;
  wire s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11;
  wire c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11;

  // Generate partial products
  assign p0 = a & {4{b[0]}};
  assign p1 = a & {4{b[1]}};
  assign p2 = a & {4{b[2]}};
  assign p3 = a & {4{b[3]}};

  // Wallace Tree Reduction
  ha h1(s0, c0, p0[1], p1[0]);
  fa f1(s1, c1, p0[2], p1[1], p2[0]);
  fa f2(s2, c2, p0[3], p1[2], p2[1]);
  fa f3(s3, c3, p1[3], p2[2], 1'b0);

  fa f4(s4, c4, s1, c0, 1'b0);
  fa f5(s5, c5, s2, c1, p3[0]);
  fa f6(s6, c6, s3, c2, p3[1]);
  fa f7(s7, c7, p2[3], c3, p3[2]);
  fa f8(s8, c8, s5, c4, 1'b0);
  fa f9(s9, c9, s6, c8, c5);
  fa f10(s10, c10, s7, c6, c9);
  fa f11(s11, c11, p3[3], c7, c10);

  // Final product assignment
  assign prod[0] = p0[0];
  assign prod[1] = s0;
  assign prod[2] = s4;
  assign prod[3] = s8;
  assign prod[4] = s9;
  assign prod[5] = s10;
  assign prod[6] = s11;
  assign prod[7] = c11;
endmodule
```