Digital Low-Power: World Domination

WS Audiology
Oticon

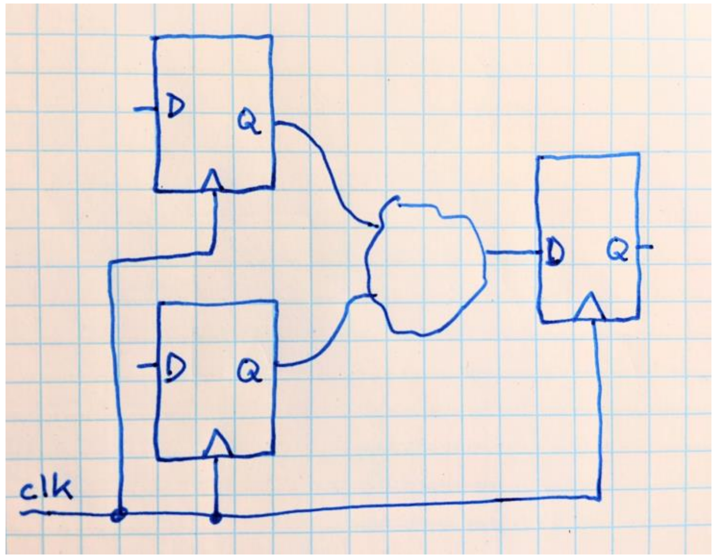
RTL - Register Transfer Logic

RTL written in Verilog or VHDL

```
always_ff @(posedge clk)
  in_reg <= in;

always_comb begin
  temp = in_reg[1] + in_reg[2];
  out_nxt = temp * in_reg[3];
end

always_ff @(posedge clk)
  out <= out_nxt;
```

Synthesis

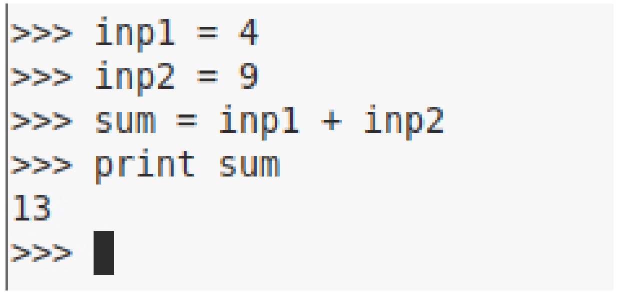
Gates



Beware: synthesis allowed to mangle/restructure logic cloud. Often not important, but occasionally it is!

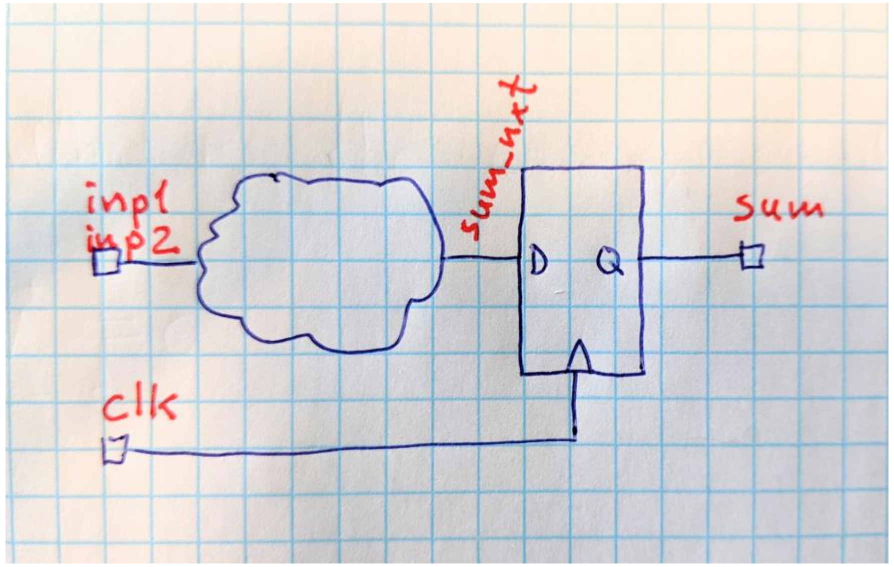
Let's Design

Higher level model or description e.g., Word, Matlab or Python:



```
>>> inp1 = 4
>>> inp2 = 9
>>> sum = inp1 + inp2
>>> print sum
13
>>> 
```

Mental picture:

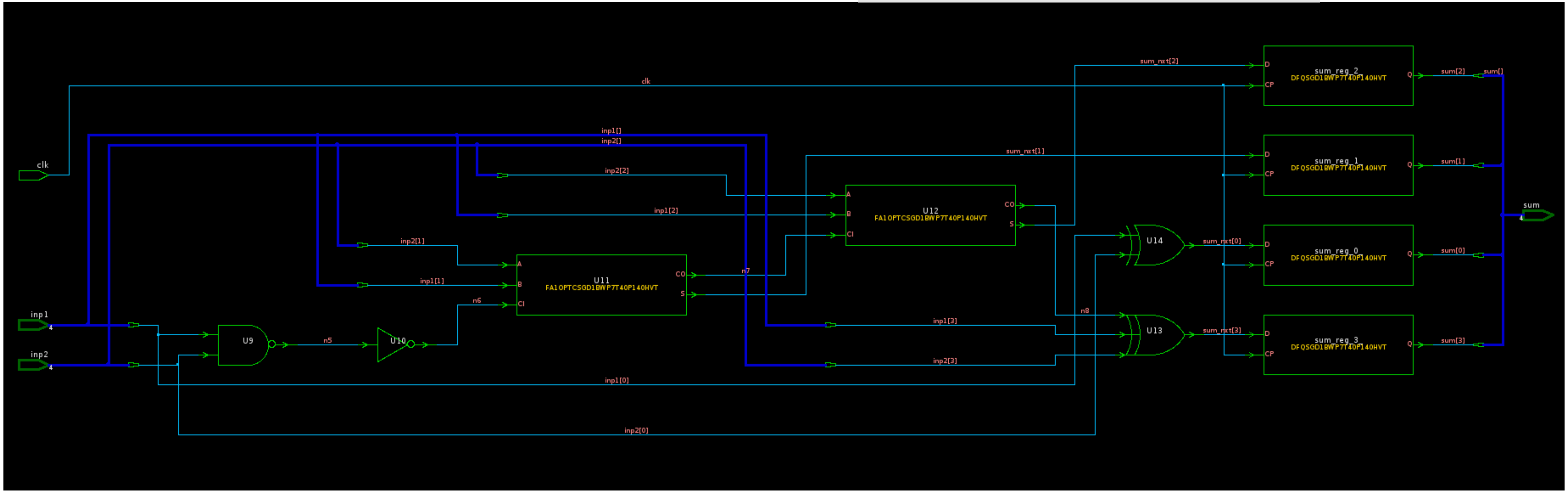


RTL:

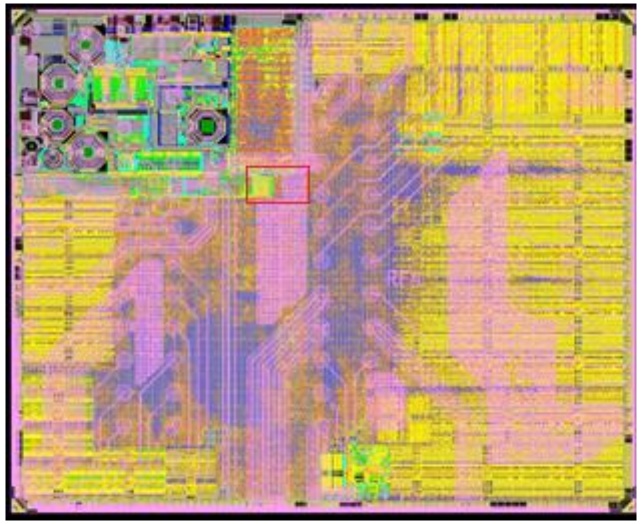
```
module conquer_world (clk, inp1, inp2, sum);
  input clk;
  input [3:0] inp1, inp2;
  output logic [3:0] sum;

  logic [3:0] sum_nxt;
  always_comb
    sum_nxt = inp1 + inp2;

  always_ff @(posedge clk)
    sum <= sum_nxt;

endmodule
```

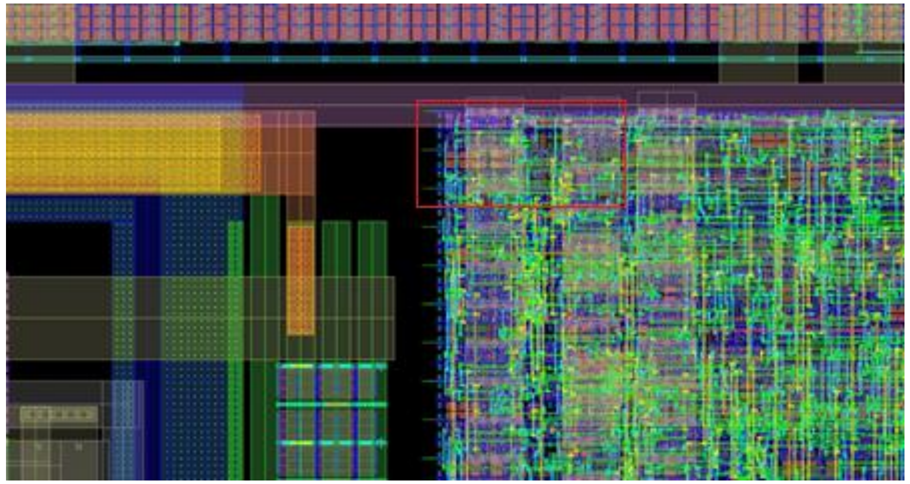
Synthesis - converting RTL to gates

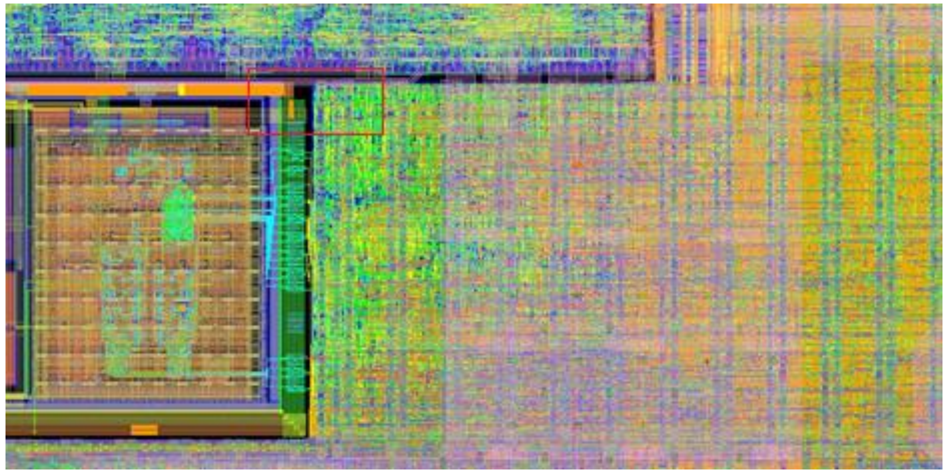
```
module conquer_world (clk, inp1, inp2, sum);
  input clk;
  input [3:0] inp1, inp2;
  output logic [3:0] sum;

  logic [3:0] sum_nxt;
  always_comb
    sum_nxt = inp1 + inp2;

  always_ff @(posedge clk)
    sum <= sum_nxt;

endmodule
```

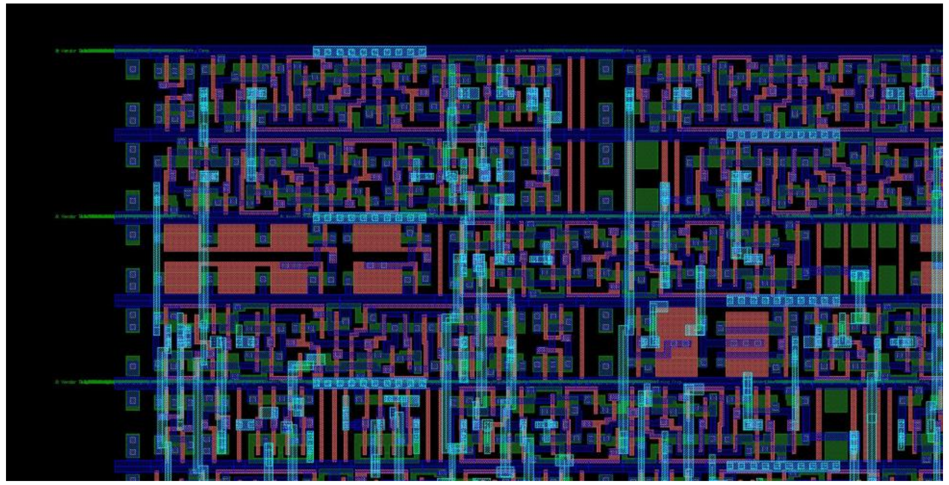


Place Route of Gates

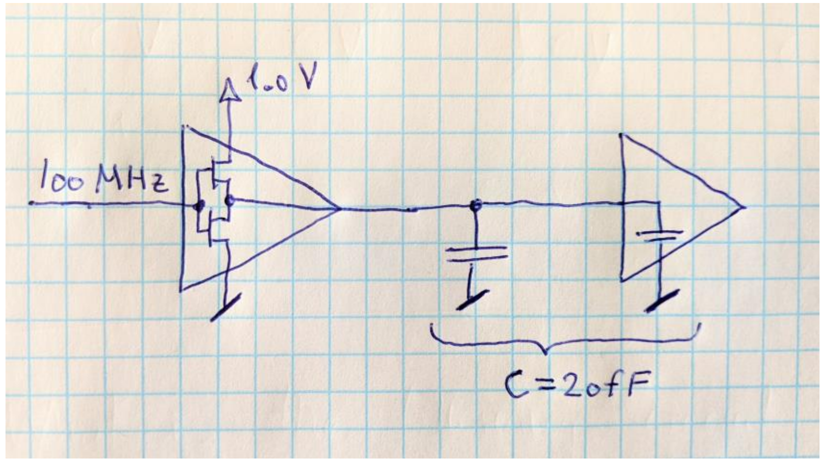








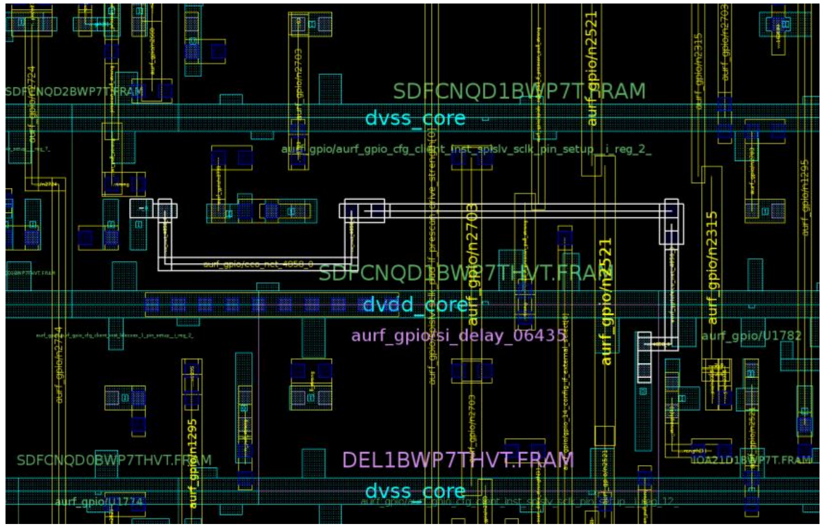
Low-Power Basics



$$E = \frac{1}{2} C V^2$$

Q: What is the energy on the capacitor ?

A: 1/2 * 20fF * (1V)^2 = 10 fJ



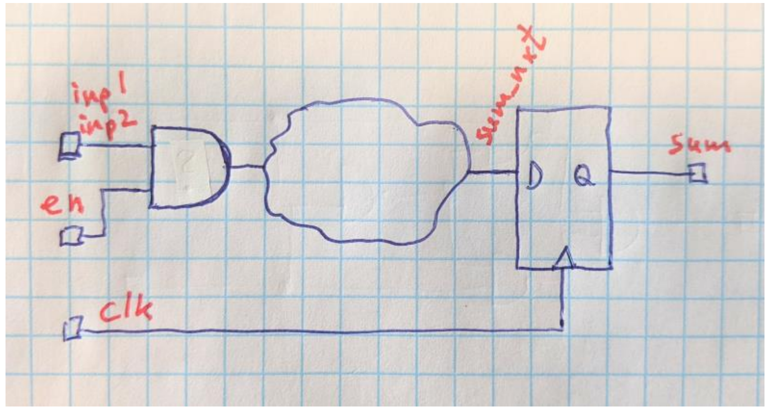
$$P = f \cdot E \qquad P = f \cdot C \cdot V^2$$

Q: How much power does it use ?

A: 100 MHz * 2* 10 fJ = 2 uW

Low-Power Design - Prevent Toggling

Mental model



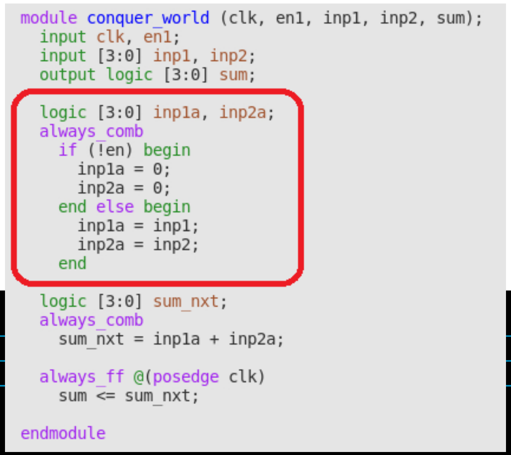
Original RTL

```
module conquer_world (clk, inp1, inp2, sum);
  input clk;
  input [3:0] inp1, inp2;
  output logic [3:0] sum;

  logic [3:0] sum_nxt;
  always_comb
    sum_nxt = inp1 + inp2;

  always_ff @(posedge clk)
    sum <= sum_nxt;

endmodule
```

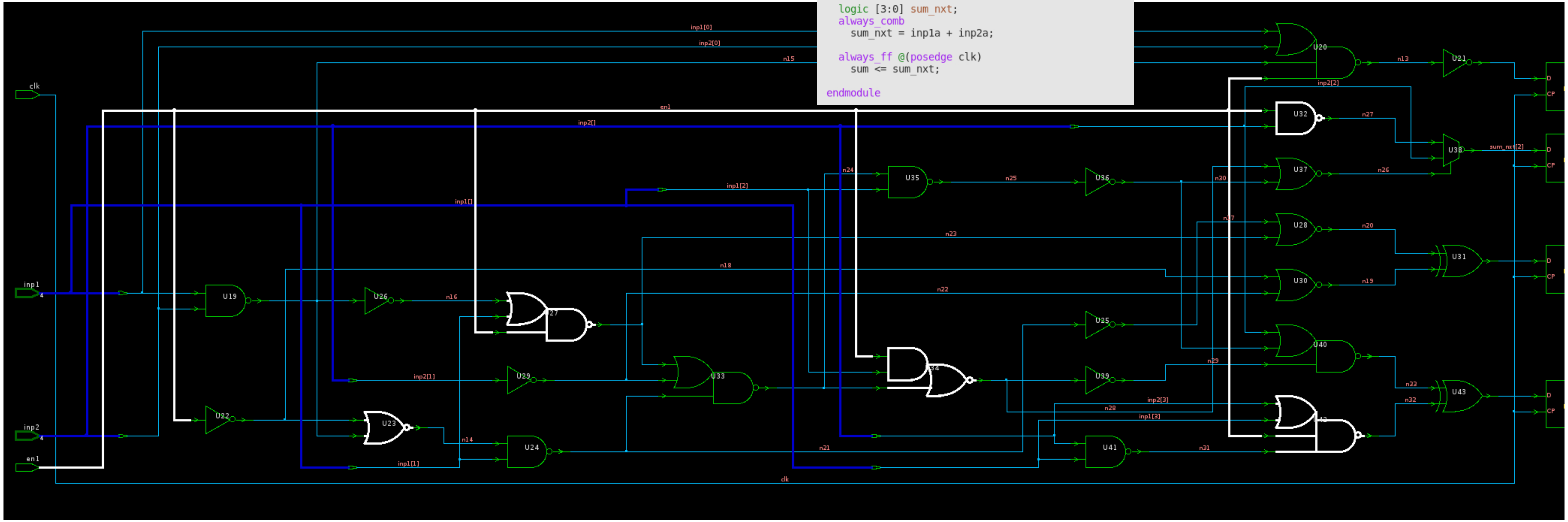
Low-Power RTL

```
module conquer_world (clk, en1, inp1, inp2, sum);
  input clk, en1;
  input [3:0] inp1, inp2;
  output logic [3:0] sum;

  logic [3:0] inp1a, inp2a;
  always_comb
    if (!en) begin
      inp1a = 0;
      inp2a = 0;
    end else begin
      inp1a = inp1;
      inp2a = inp2;
    end

  logic [3:0] sum_nxt;
  always_comb
    sum_nxt = inp1a + inp2a;

  always_ff @(posedge clk)
    sum <= sum_nxt;

endmodule
```

Low-Power Design - Prevent Toggling



```
module conquer_world (clk, en1, inp1, inp2, sum);
  input clk, en1;
  input [3:0] inp1, inp2;
  output logic [3:0] sum;

  logic [3:0] inp1a, inp2a;
  always_comb
    if (!en) begin
      inp1a = 0;
      inp2a = 0;
    end else begin
      inp1a = inp1;
      inp2a = inp2;
    end

  logic [3:0] sum_nxt;
  always_comb
    sum_nxt = inp1a + inp2a;

  always_ff @(posedge clk)
    sum <= sum_nxt;

endmodule
```



Junior Digital Designer: WTF. This is not what I wrote in RTL !

Senior Digital Designer: You forgot the slide at the beginning: synthesis allowed to restructure expression, keeping final expression intact

Low-Power Design - Clock Gating

Flop clock input: 0.4 uW/clk_input (100MHz)

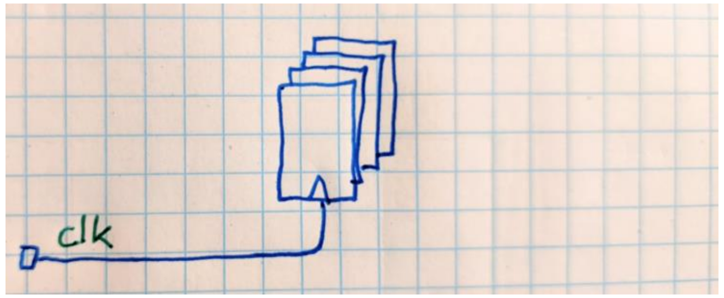
Clock-gater input: 0.1 uW (100MHz)

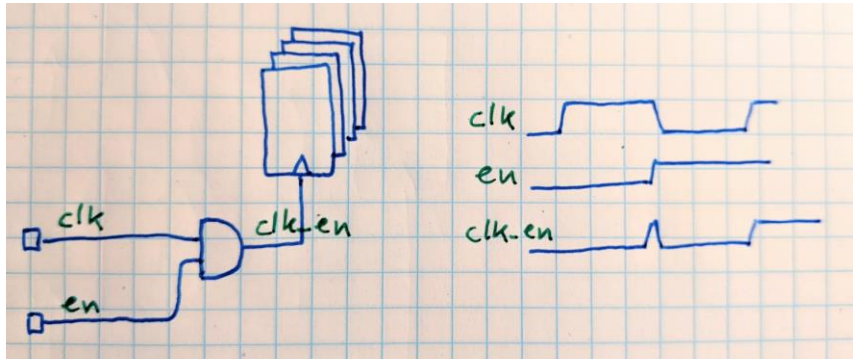
Q: how much power did the clock-gater save (assume mostly stopping the clock) ?

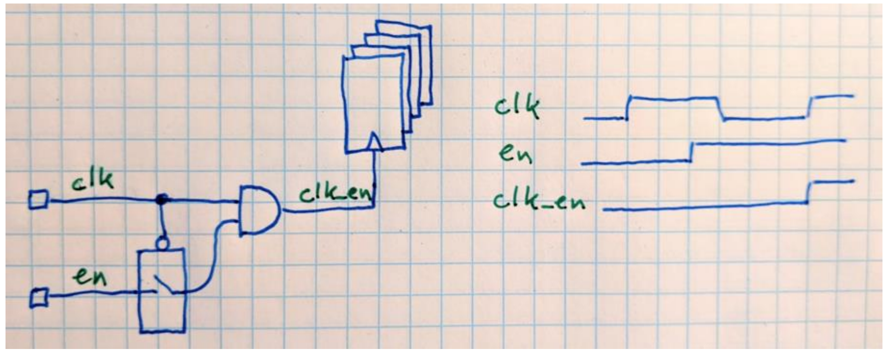
A: before 4*0.4uW, after 0.1uW, saved 1.5uW

```
module conquer_world (clk, inp1, inp2, sum);
  input clk;
  input [3:0] inp1, inp2;
  output logic [3:0] sum;

  logic [3:0] sum_nxt;
  always_comb
    sum_nxt = inp1 + inp2;

  always_ff @(posedge clk)
    sum <= sum_nxt;

endmodule
```

```
module conquer_world (clk, en1, inp1, inp2, sum);
  input clk, en1;
  input [3:0] inp1, inp2;
  output logic [3:0] sum;

  logic [3:0] sum_nxt;
  always_comb
    sum_nxt = inp1 + inp2;

  always_ff @(posedge clk)
   if (en1)
     sum <= sum_nxt;

endmodule
```

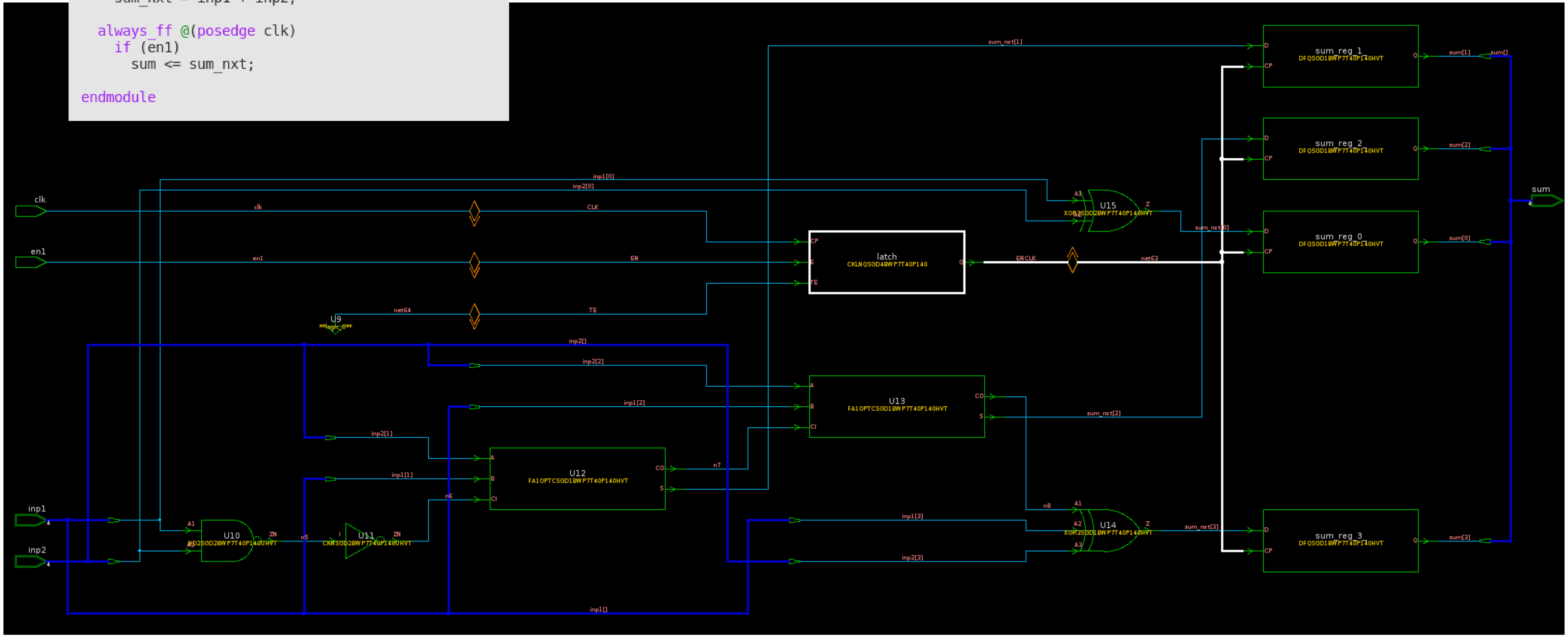






Low-Power Design - Clock Gating

```
module conquer_world (clk, en1, inp1, inp2, sum);
  input clk, en1;
  input [3:0] inp1, inp2;
  output logic [3:0] sum;

  logic [3:0] sum_nxt;
  always_comb
    sum_nxt = inp1 + inp2;

  always_ff @(posedge clk)
    if (en1)
      sum <= sum_nxt;

endmodule
```



Ready to take over the world ?