Property Generator: Generation of Formal Assertion IP

Tobias Ludwig, CEO

Gen AI and Coding Tasks

Software domain

From just an auto-complete tool to bringing up a web app in minutes

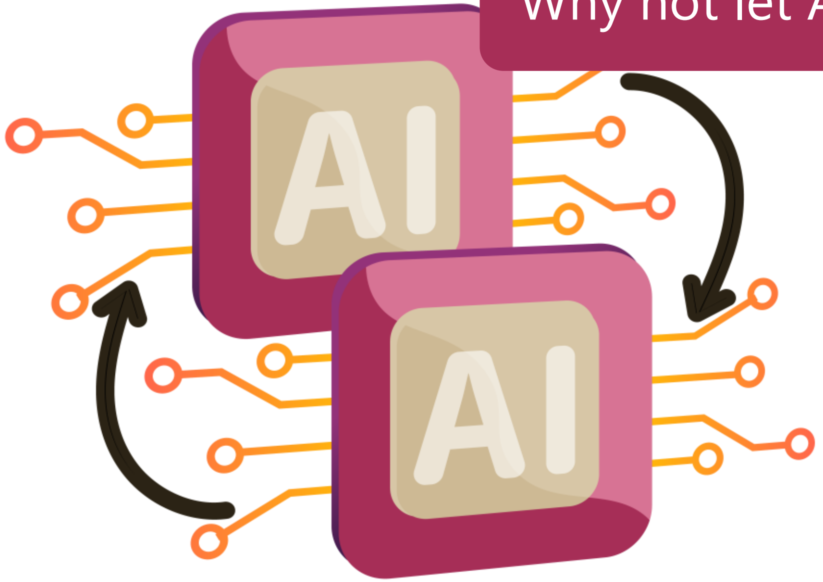
Not true for RTL code generation … yet

- Massive progress on VerilogEval benchmark, from 26% GPT3.5 to 63% GPT4o to 98% Agentic systems
- CVDP currently hoovers around 30%

Gen AI and RTL Design

Verification remains the bottleneck

Gen AI and RTL Design Verification



Why not let AI verify AI?

Will not scale for chip design … why?

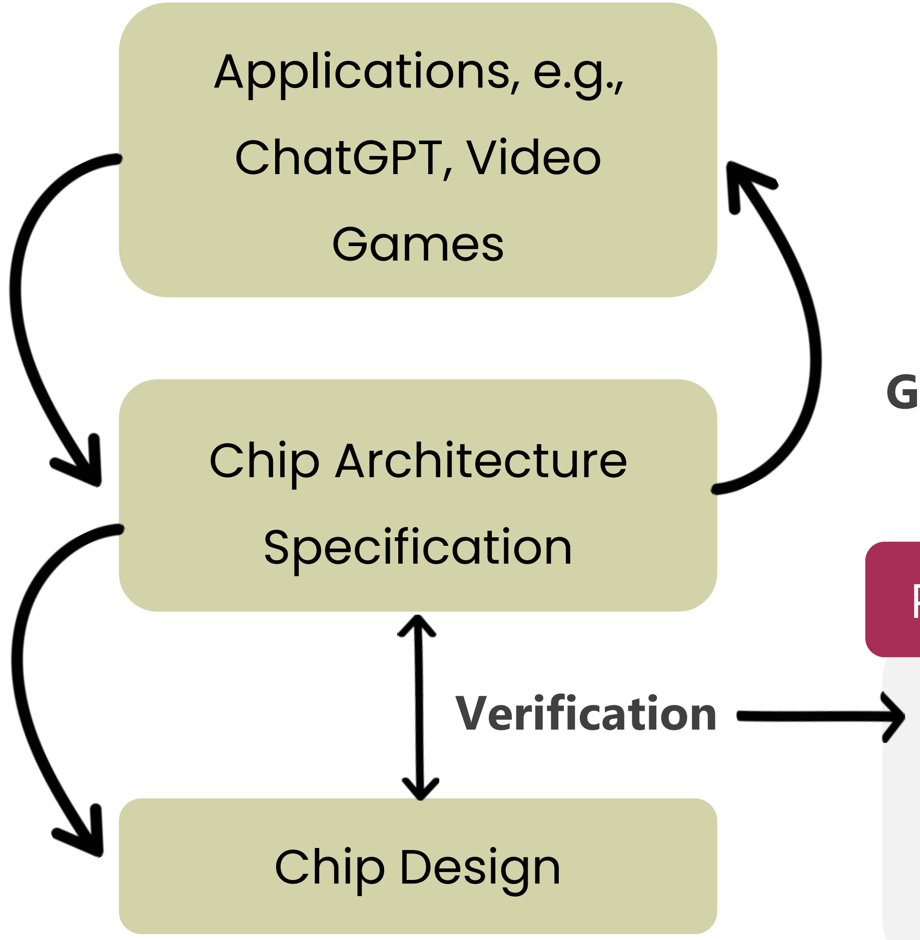
- Frontier LLMs are trained with similar data and reward functions
- They tend to make similar errors
- Amplifies the risk of masking bugs

- Bugs are expensive, re-spins costs millions of dollars
- Human oversight will remain most for foreseeable future
- Auditing thousands of lines of code can be more effort than writing them

What it takes to have a successful chip?

Tens of thousands of engineer days

Genius Chip Architect



Software ecosystem

Compiler, etc.

Genius Software Developer

Genius Chip Designer

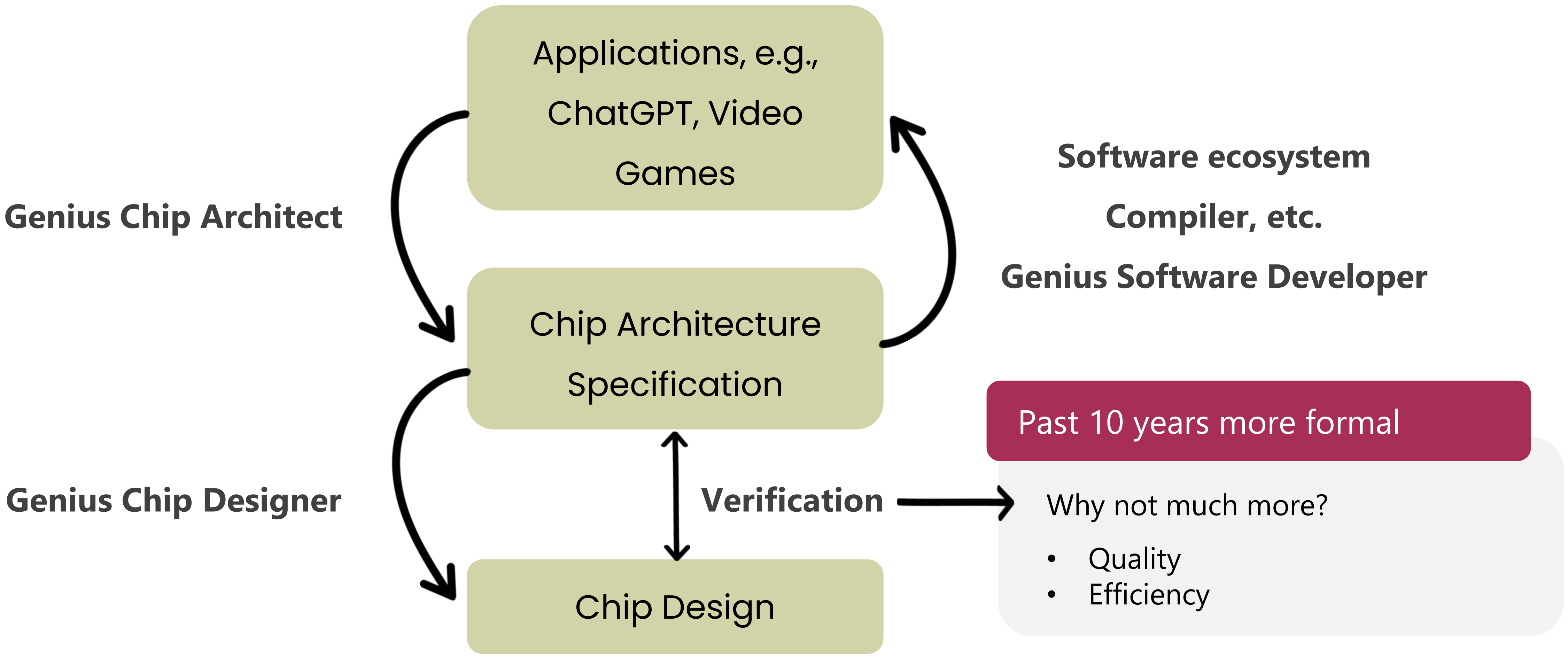
Past 10 years more formal

Why?

- Exhaustive Verification
- Fast Setup
- Fast debug

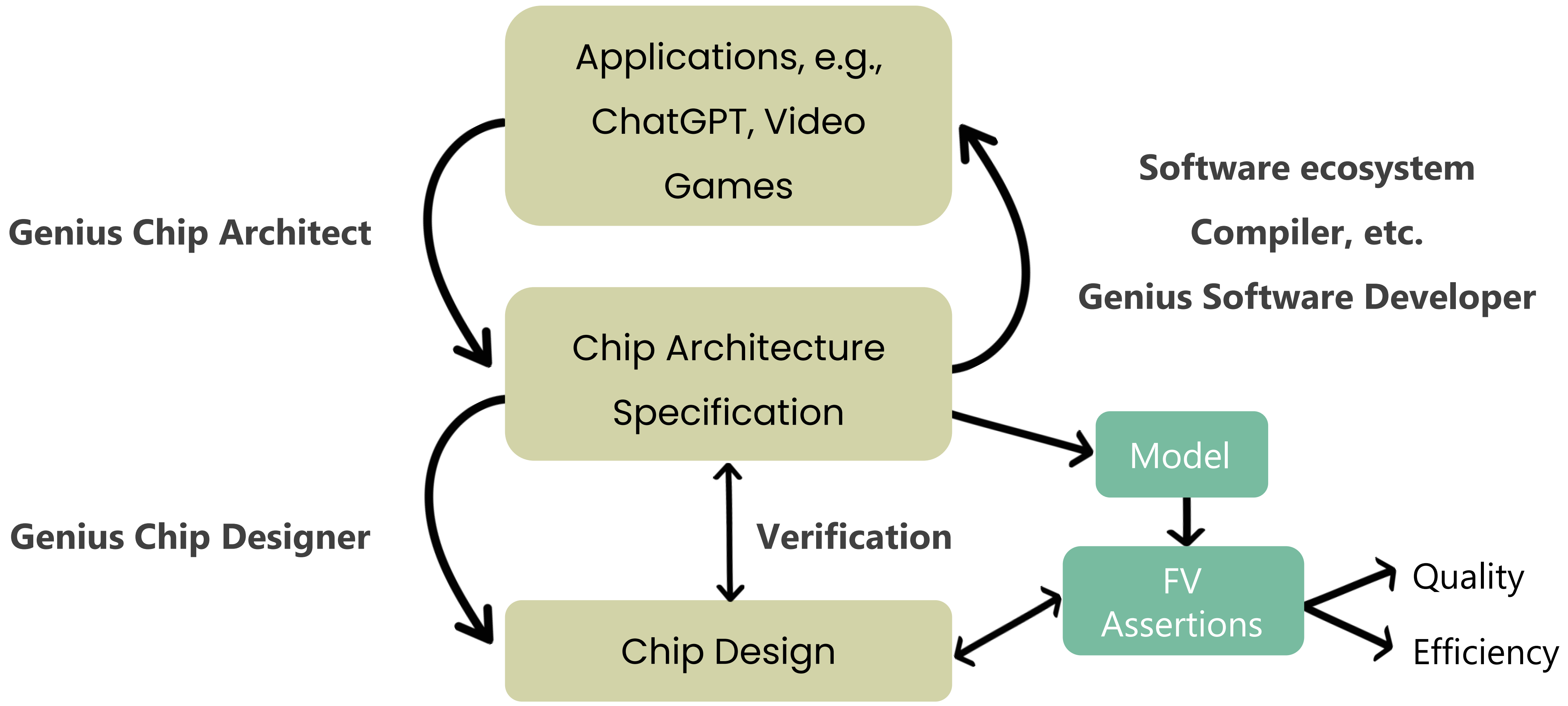
What it takes to have a successful chip?

Tens of thousands of engineer days



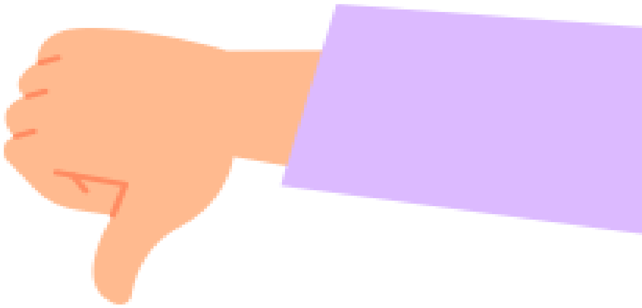
What it takes to have a successful chip?

Tens of thousands of engineer days



Mind the gap

```
// FIFO watermark comparator context
// fill_level: current FIFO occupancy (0..DEPTH)
// watermark: programmable threshold
// above_wm: 1 when fill_level > watermark
// below_wm: 1 when fill_level < watermark

above_a: assert property (disable iff(!rst_n)
                         (fill_level > watermark) |-> (above_wm && !below_wm)
);

below_a: assert property (disable iff(!rst_n)
                         (fill_level < watermark |-> (!above_wm && below_wm)
);
```



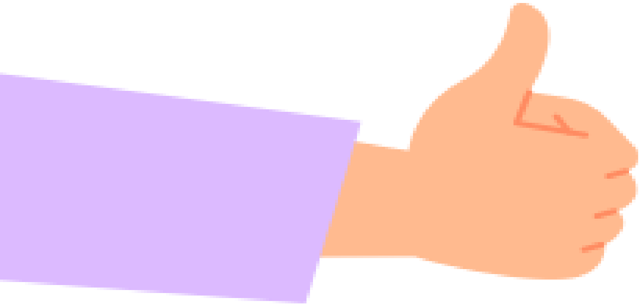
Complexity sneaks in fast

```
// NoC ingress flow-control with four throttle modes
// q_depth: 0..MAX
// t0 < t1 < t2: programmable thresholds
// mode: 2’b00=PASS, 2’b01=SLOW1, 2’b10=SLOW2, 2’b11=DROP

pass_a: assert property (disable iff(!rst_n)
        q_depth < t0 |-> (mode == 2’b00)
);

slow1_a: assert property (disable iff(!rst_n)
        ((q_depth > t0) && (q_depth < t1)) |-> (mode == 2’b01)
);

slow2_a: assert property (disable iff(!rst_n)
        ((q_depth > t1) && (q_depth < t2)) |-> (mode == 2’b10)
);

drop_a: assert property (disable iff(!rst_n)
        (q_depth > t2) |-> (mode == 2’b11)
);
```

Turning intend into properties

```
int fill_level, watermark;
bool above_wm, below_wm;
if (fill_level > watermark) {
        above_wm = true;
        below_wm = false;
} else {
        above_wm = false;
        below_wm = true;
}
```

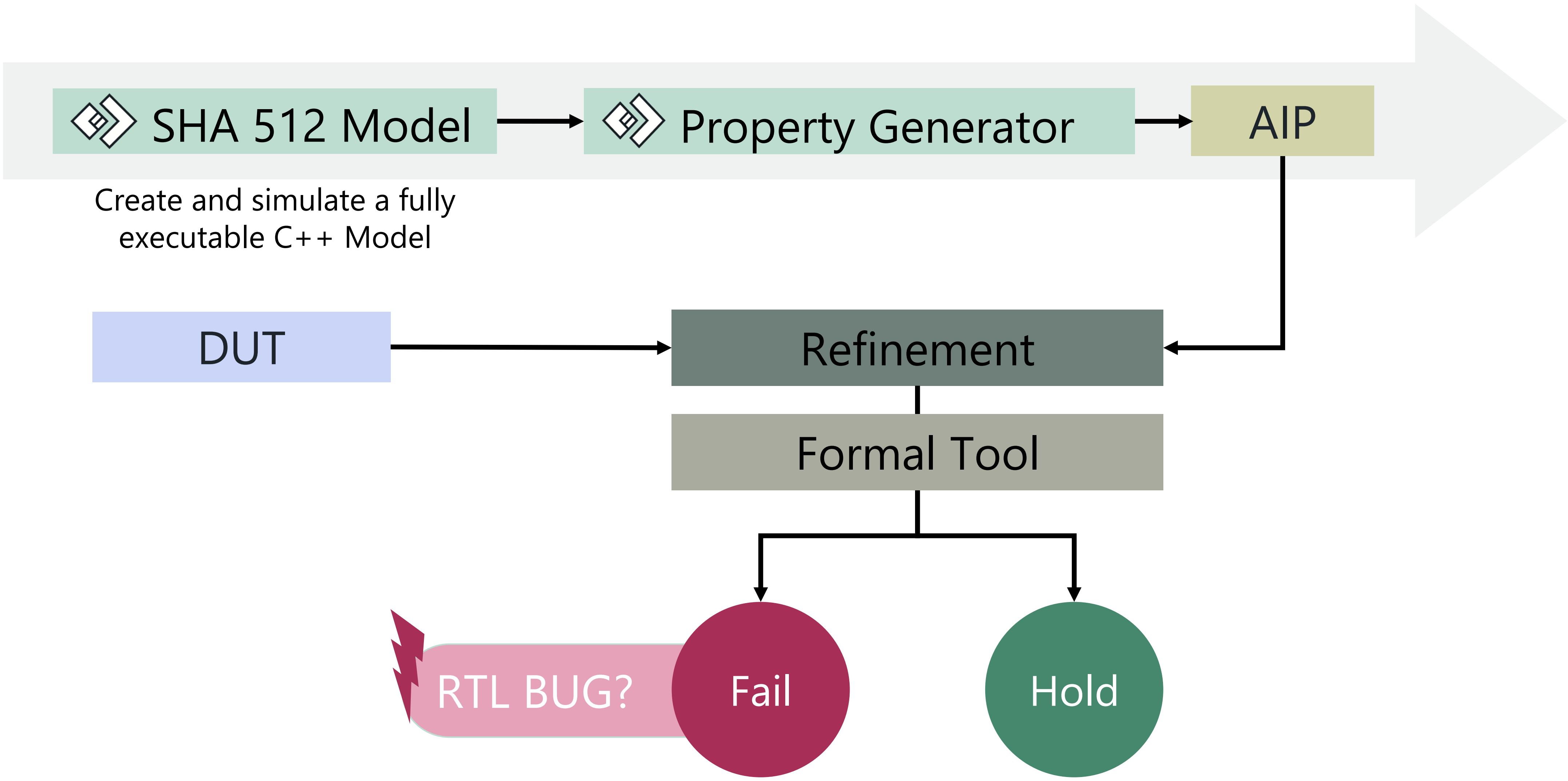
```
// FIFO watermark comparator context
// fill_level: current FIFO occupancy (0..DEPTH)
// watermark: programmable threshold
// above_wm: 1 when fill_level > watermark
// below_wm: 1 when fill_level <= watermark

above_a: assert property (disable iff(!rst_n)
        (fill_level > watermark) |-> (above_wm && !below_wm)
);

below_a: assert property (disable iff(!rst_n)
        (fill_level <= watermark) |-> (!above_wm && below_wm)
);
```



SHA512 Algorithm

```
for (i=0; i<NUM_ROUNDS; ++i) {
        k = K[i];
        if (i < 16){
                w = W[i];
        } else {
                tmp_w = compute_w(W[14],W[9],W[1],W[0]);
                for (j=0; j<15; ++j) {
                        W[j] = W[(j+1)];
                }
                W[15] = tmp_w;
                w = tmp_w;
        }

        t1 = T1(e, f, g, h, K[i],w);
        t2 = T2(a, b, c);
        h = g; g = f;
        f = e; e = (d + t1);
        d = c; c = b;
        b = a; a = (t1 + t2);
}
```

Here is what we are going to do



Model to Property

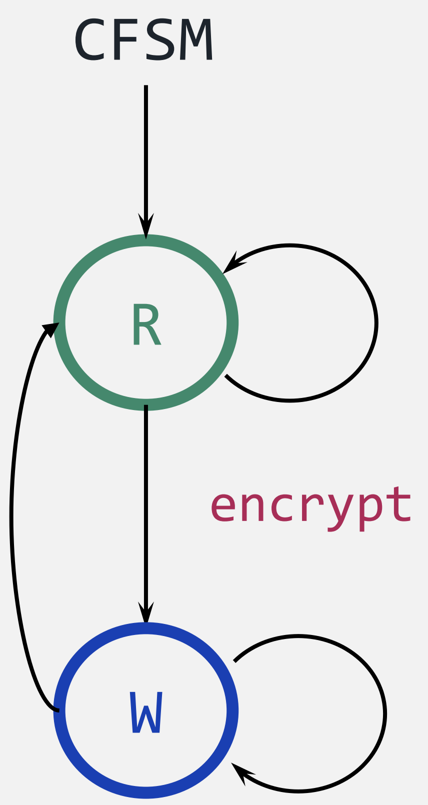
```
SHA_Input->read(SHA_in, "IDLE");

if(init){
        // Some Computation
}
// Some Computations
for(i = 0; i < NUM_ROUNDS; ++i){
  insert_state("SHA_Rounds");
  if(i < 16){
    // Some Computation
  } else {
        // Some Computation
  }
  // Some Computation
}
insert_state("DONE");
```

```
property fsm_SHA_Rounds_to_SHA_Rounds_1_p;
        SHA_Rounds &&
        i >= 16 &&
        ((i + 1) < 80)
|->
        ##1 ((SHA_Input_rdy == 0) &&
            (out_vld == 0)) and
        ##1
        SHA_Rounds &&
        $stable(H) &&
        W == $past(W_2_i, 1) &&
        a == ($past(T1_1_i, 1) + $past(T2_0_i, 1)) &&
        b == $past(a, 1) &&
        c == $past(b, 1) &&
        d == $past(c, 1) &&
        e == ($past(d, 1) + $past(T1_1_i, 1)) &&
        f == $past(e, 1) &&
        g == $past(f, 1) &&
        h == $past(g, 1) &&
        i == (7'd1 + $past(i, 1));
endproperty
```

Modelling

Conceptualize the abstract behavior of the design as a model

```
SC_MODULE(encrypt) {
    // constructor and declarations
    void fsm() {
       while(true){
        input->read(data);
        output->write(encrypt(data));
        // some code
     } } };
```

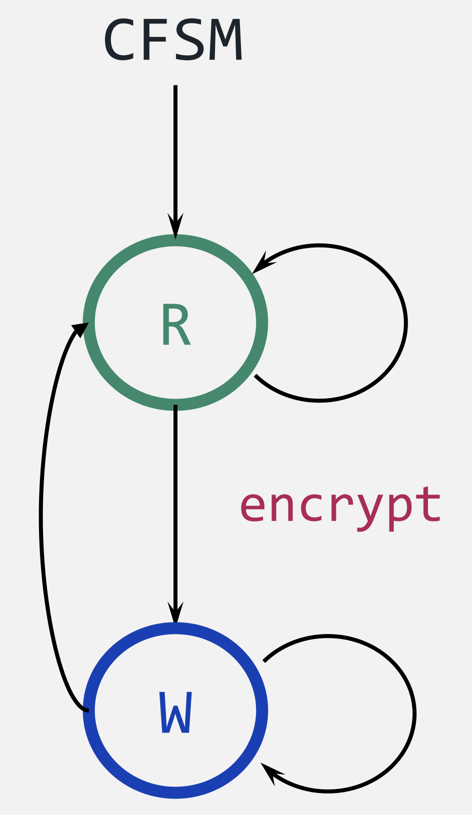


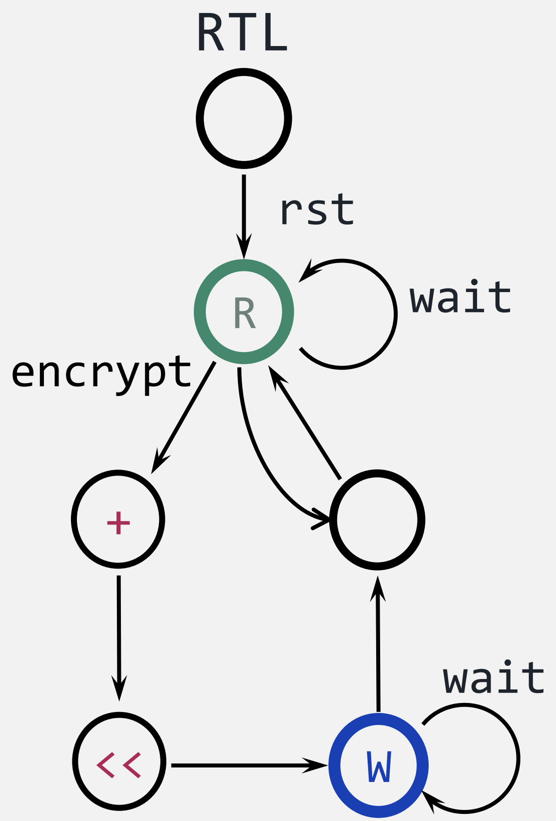

```
Property encrypt;
assume:
   at t:   state == R;
   at t:   data_valid == 1;
prove:
   at t+n: out(enc(data@t));
   at t+n: state == W;
endproperty;
```

Abstraction to Implementation

Match the model with the concrete implementation

```
SC_MODULE(encrypt) {
     // constructor and declarations
     void fsm() {
        while(true){
         input→read(data);
         output→write(encrypt(data));
         // some code
      } } };
```





Refining the model

Provide the details missing in the abstract model, like timing and bit-accuracy

```
property encrypt(length);
    //local variables
    logic[1023:0] data_0;
    // store values @t
    ##0 (valid_data, data_0 = bus_in_data()) and
    // triggers
    ##0 state() == R
|->
    ##length state() == W and
    ##length bus_out()==enc(data_0)
endproperty;
```

```
function logic[1023:0] bus_in_data()
        return data_in;
endfunction;
```

```
function logic[1023:0] bus_in_data()
  return {
        $past(data_in,3),
        $past(data_in,2),
        $past(data_in,1),
        data_in};
endfunction;
```

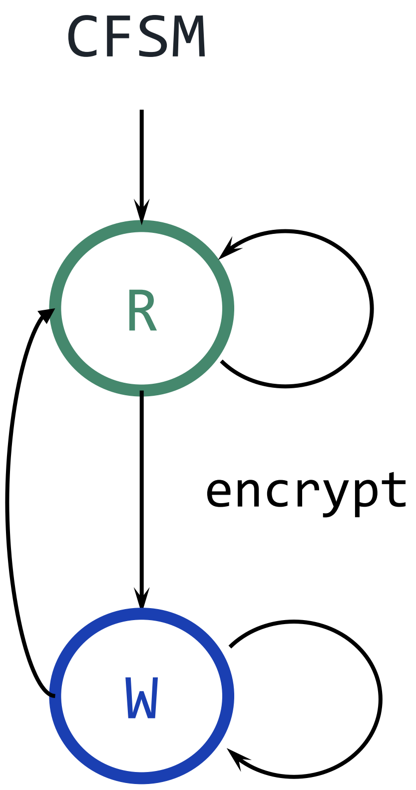
Refining the model

Provide the details missing in the abstract model, like timing and bit-accuracy

```
property encrypt(length);
    //local variables
    logic[1023:0] data_0;
    // store values @t
    ##0 (valid_data, data_0 = bus_in_data()) and
    // triggers
    ##0 state() == R
|->
    ##length state() == W and
    ##length bus_out()==enc(data_0)
endproperty;
```

```
function logic[1:0] state()
     if(state_reg == 2‘b00)
         return R
     else if(state_reg == 2‘b01)
         return W
endfunction;
```

Generating properties without gaps



FSM Path	Property in Assertion IP
Reset -> Read	reset_p
Read -> Read	read_to_read_p
Read -> Write	read_to_write_p
Write -> Write	write_to_write_p
Read -> Read	write_to_read_p

We help chip teams reach formal sign-off

1 **Transforming** FV from an art into a system

2 **Services & Training** to achieve Formal Sign-Off

3 **Technology R&D** to enhance productivity in Formal

Team 35+

Kaiserslautern, Germany

We love formal

since 2020

What we’ve delivered

250+

Projects completed

Comprehensive coverage across CPU, cache, and interconnect projects

6

Fortune 100 Customers

Trusted by the world's leading semiconductor companies

750+

Design Bugs Uncovered

Critical issues identified and reported through formal verification

Want to learn more?

Follow us to get notified on news about: Formal services, training and software

Tobias Ludwig

CEO

Email: Tobias.ludwig@lubis-eda.com

Web: www.lubis-eda.com

LinkedIn