

Finite State Machine Implementation (2)

Schematic and HDL Methods

Notes from Reviewers

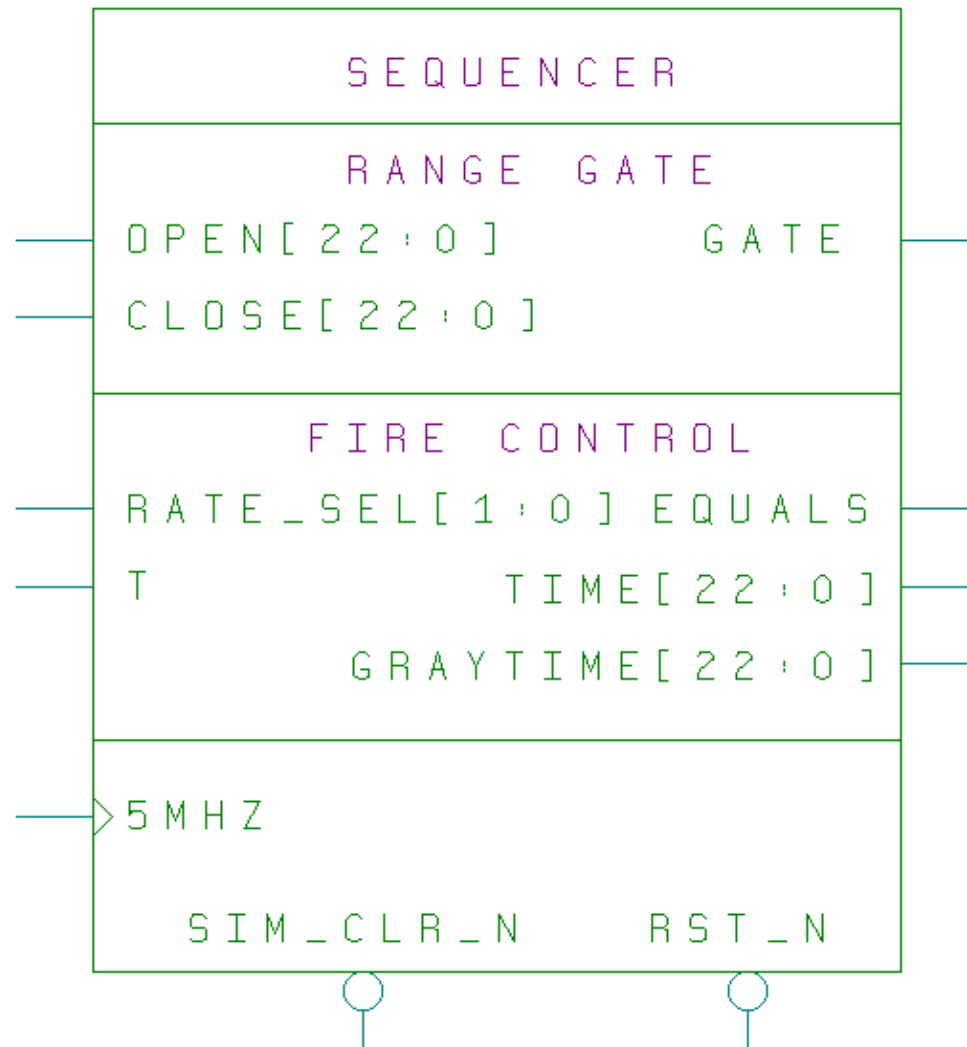
Examples from Two Flight Projects

I've been pondering VHDL coding styles lately, seeing as how I've been beating my head against the BOM FPGA for months now. The main part of the BOM is described in *33 pages of spaghetti VHDL* that is broken into 5 processes, one of which is 16 pages long. There's no block diagram and very little descriptive text to guide one through the design. In contrast, the 29KPL154 [microcontroller] was easy to read because it started with a block diagram and treated each block as a component; I don't think many of the components were longer than 2 pages. I can see how VHDL could be made easier to read if some rules were enforced.

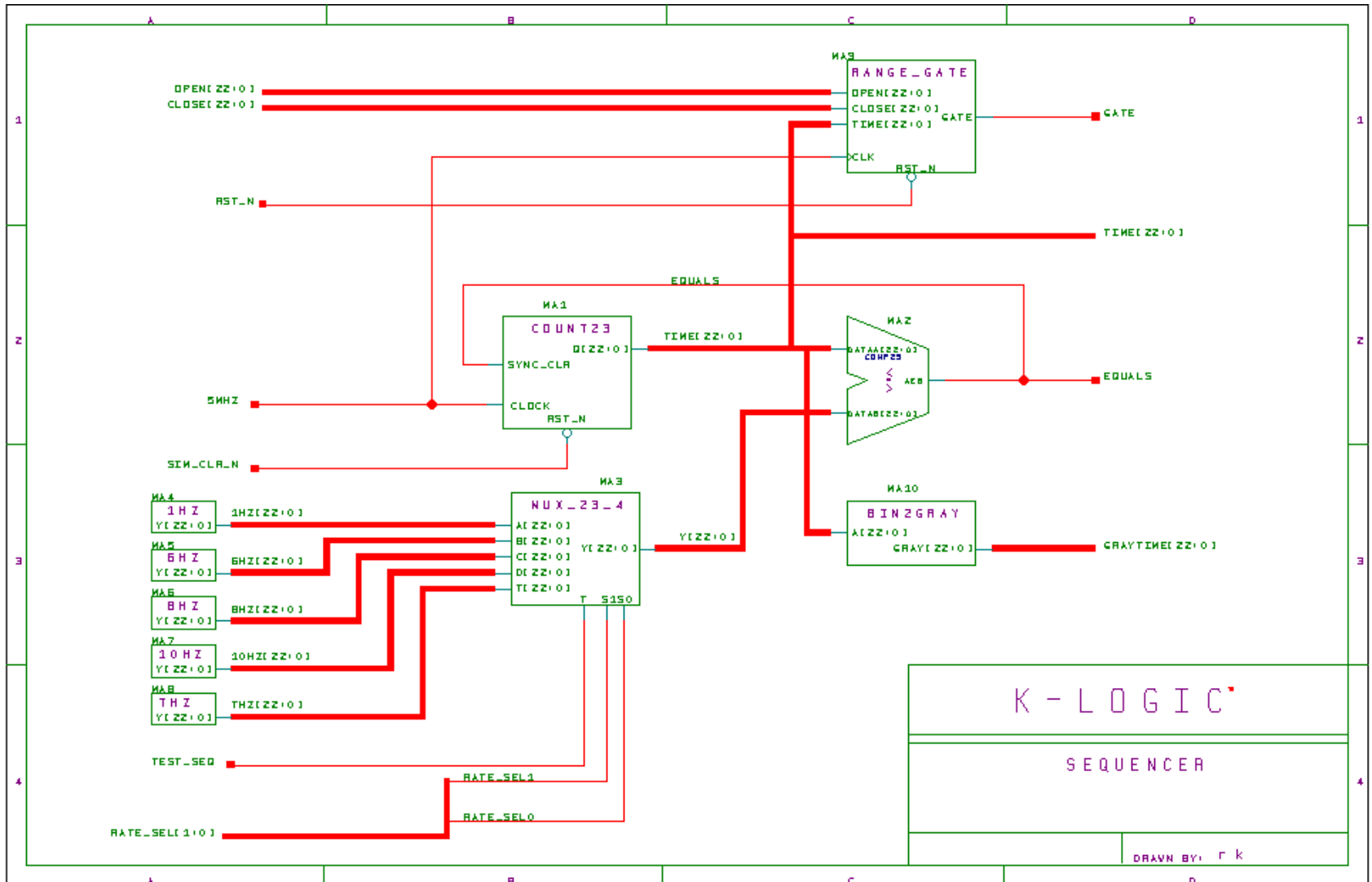
The VHDL description started with *16 pages of structural VHDL* with no comments. It was a mess and making a schematic from that, by hand, was painful. I never did finish the review as the design needed so many changes to make it flight worthy that it made more sense to simply trash it and start over. Modifying that mess would simply be too error prone.

Master Sequencer Example

Symbol, Schematic Representation



Master Sequencer Example Schematic



Master Sequencer Example

VHDL Implementation¹

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
--
--
entity SEQUENCER is
  port( RST_N, C5MHZ, SIM_CLR_N : in std_logic;
        THE : in std_logic_vector(22 downto 0);
        TEST_SEQ : in std_logic;
        RATE_SEL : in std_logic_vector(1 downto 0);
        OPEN_VALUE,CLOSE_VALUE : in std_logic_vector(22 downto 0);
        GATE : out std_logic;
        EQUALS : out std_logic;
        TIME_NOW, GRAYTIME : out std_logic_vector(22 downto 0)
  );
end SEQUENCER;

architecture RTL_ARCH of SEQUENCER is
  constant R1HZ : std_logic_vector(22 downto 0) := std_logic_vector(to_unsigned(500000,23));
  constant R6HZ : std_logic_vector(22 downto 0) := std_logic_vector(to_unsigned(83333,23));
  constant R8HZ : std_logic_vector(22 downto 0) := std_logic_vector(to_unsigned(625000,23));
  constant R10HZ : std_logic_vector(22 downto 0) := std_logic_vector(to_unsigned(500000,23));
  component COUNT23
    port( RST_N,SYNC_CLR, CLOCK : in std_logic;
          Q : out std_logic_vector(22 downto 0) );
  end component;
  component MUX_23_4
    port( A, B, C, D, T : in std_logic_vector(22 downto 0);
          T_SEL, S1, S0 : in std_logic;
          Y : out std_logic_vector(22 downto 0)
    );
  end component;
  component COMP23
    port( DATA, DATAB : in std_logic_vector(22 downto 0);
          AEB : out std_logic
    );
  end component;
  component GATE_RANGE
    port( RST_N, CLOCK : in std_logic;
          OPEN_VALUE,CLOSE_VALUE, TIME_NOW : in std_logic_vector(22 downto 0);
          GATE : out std_logic
    );
  end component;
  component BIN2GRAY23
    port( A : in std_logic_vector(22 downto 0);
          Y : out std_logic_vector(22 downto 0)
    );
  end component;
  signal Y : std_logic_vector(22 downto 0);
  signal EQUALS_internal : std_logic;
  signal TIME_NOW_internal : std_logic_vector(22 downto 0);

begin
  MA1: COUNT23 port map
    (RST_N => SIM_CLR_N, SYNC_CLR => EQUALS_internal, CLOCK => C5MHZ,
     Q => TIME_NOW_internal
    );
  MA3: MUX_23_4 port map
    (A => R1HZ, B => R6HZ, C => R8HZ, D => R10HZ, T => THE,
     T_SEL => TEST_SEQ, S1 => RATE_SEL(1), S0 => RATE_SEL(0),
     Y => Y
    );
  MA2: COMP23 port map
    (DATA => TIME_NOW_internal, DATAB=> Y,
     AEB => EQUALS_internal
    );
  MA7: GATE_RANGE port map
    (RST_N => RST_N,
     CLOCK => C5MHZ,
     OPEN_VALUE => OPEN_VALUE,CLOSE_VALUE => CLOSE_VALUE, TIME_NOW => TIME_NOW_internal,
     GATE => GATE
    );
  MA8: BIN2GRAY23 port map
    (A => TIME_NOW_internal,
     Y => GRAYTIME
    );

  EQUALS <= EQUALS_internal;
  TIME_NOW <= TIME_NOW_internal;
end RTL_ARCH;
```

Broken down into sections and enlarged on the following pages.

¹ Design courtesy of an HDL design proponent.

Master Sequencer Example

Symbol, VHDL Representation

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
--
--
--
entity SEQUENCER is
    port( RST_N, C5MHZ, SIM_CLR_N : in std_logic;
          THZ                     : in std_logic_vector(22 downto 0);
          TEST_SEQ                 : in std_logic;
          RATE_SEL                 : in std_logic_vector(1 downto 0);
          OPEN_VALUE,CLOSE_VALUE  : in std_logic_vector(22 downto 0);
          GATE                     : out std_logic;
          EQUALS                   : out std_logic;
          TIME_NOW, GRAYTIME       : out std_logic_vector(22 downto 0)
    );
end SEQUENCER;
```

Master Sequencer Example

VHDL Implementation: Constants

```
architecture RTL_ARCH of SEQUENCER is
```

```
constant R1HZ   : std_logic_vector (22 downto 0) := std_logic_vector(to_unsigned(5000000,23));  
constant R6HZ   : std_logic_vector (22 downto 0) := std_logic_vector(to_unsigned( 833333,23));  
constant R8HZ   : std_logic_vector (22 downto 0) := std_logic_vector(to_unsigned( 625000,23));  
constant R10HZ  : std_logic_vector (22 downto 0) := std_logic_vector(to_unsigned( 500000,23));
```

Master Sequencer Example

VHDL Implementation: Components

```
component COUNT23
  port( RST_N, SYNC_CLR, CLOCK : in std_logic;
        Q : out std_logic_vector(22 downto 0)) ;
end component;
component MUX_23_4
  port( A, B, C, D, T : in std_logic_vector(22 downto 0);
        T_SEL, S1, S0 : in std_logic;
        Y : out std_logic_vector(22 downto 0)
  );
end component;
component COMP23
  port( DATAA, DATAB : in std_logic_vector(22 downto 0);
        AEB : out std_logic
  );
end component;
component GATE_RANGE
  port( RST_N, CLOCK : in std_logic;
        OPEN_VALUE, CLOSE_VALUE, TIME_NOW : in std_logic_vector(22 downto 0);
        GATE : out std_logic
  );
end component;
component BIN2GRAY23
  port( A : in std_logic_vector (22 downto 0);
        Y : out std_logic_vector (22 downto 0)
  );
end component;
```

Master Sequencer Example

VHDL Implementation: Net Definition

```
signal Y                : std_logic_vector (22 downto 0);  
signal EQUALS_internal : std_logic;  
signal TIME_NOW_internal : std_logic_vector(22 downto 0);  
  
begin
```

Master Sequencer Example

VHDL Implementation: Draw Nets

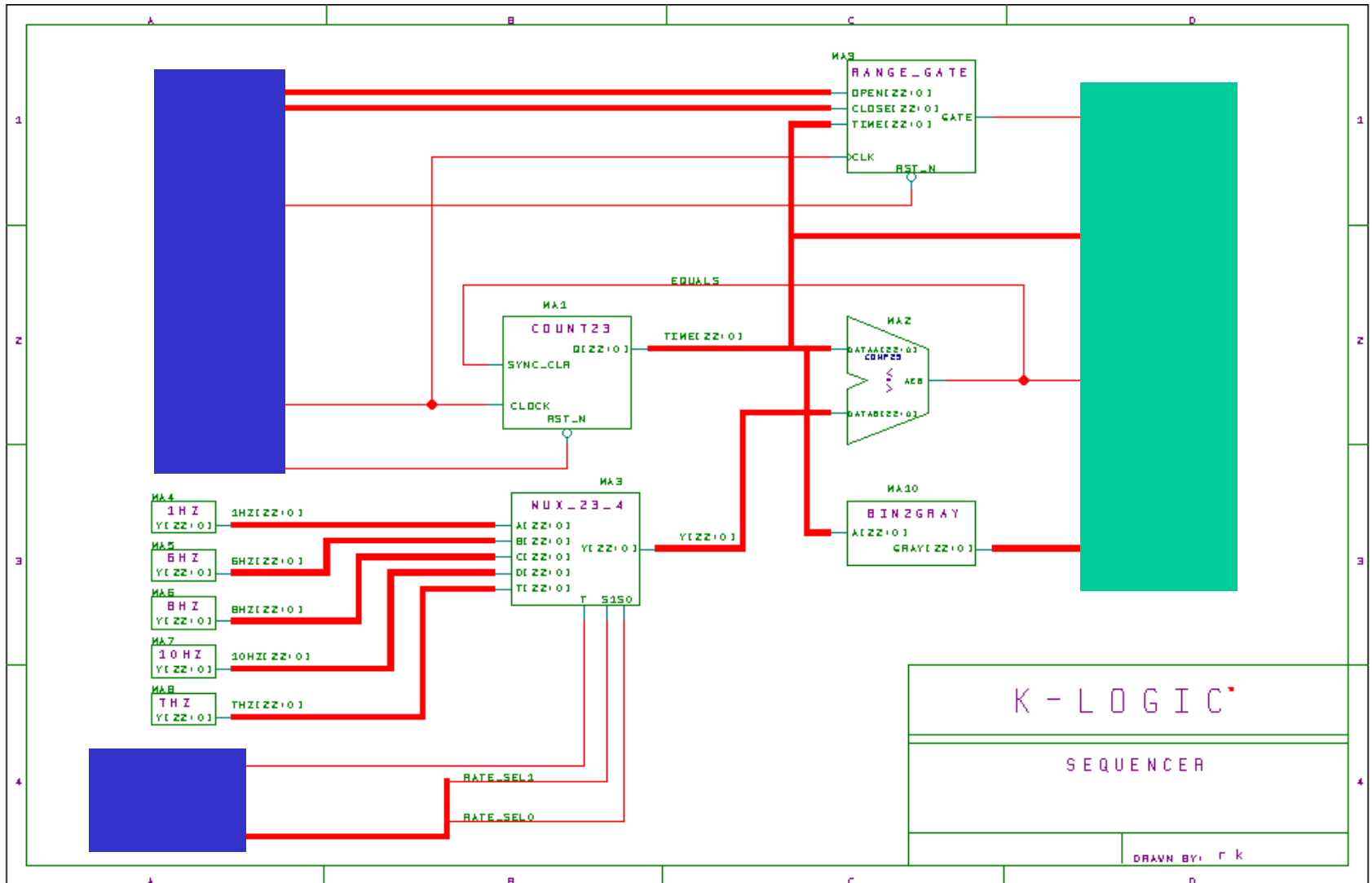
```
MA1: COUNT23    port map
                (RST_N => SIM_CLR_N, SYNC_CLR => EQUALS_internal, CLOCK => C5MHZ,
                 Q => TIME_NOW_internal
                );
MA3: MUX_23_4   port map
                (A => R1HZ, B => R6HZ, C => R8HZ, D => R10HZ, T => THZ,
                 T_SEL => TEST_SEQ, S1 => RATE_SEL(1), S0 => RATE_SEL(0),
                 Y => Y
                );
MA2: COMP23    port map
                (DATAA => TIME_NOW_internal, DATAB=> Y,
                 AEB   => EQUALS_internal
                );
MA7: GATE_RANGE port map
                (RST_N => RST_N,
                 CLOCK => C5MHZ,
                 OPEN_VALUE => OPEN_VALUE, CLOSE_VALUE => CLOSE_VALUE,
                 TIME_NOW   => TIME_NOW_internal,
                 GATE => GATE
                );
MA8: BIN2GRAY23 port map
                (A => TIME_NOW_internal,
                 Y => GRAYTIME
                );

EQUALS    <= EQUALS_internal;
TIME_NOW  <= TIME_NOW_internal;
end RTL_ARCH;
```

Analysis: Master Sequencer Example

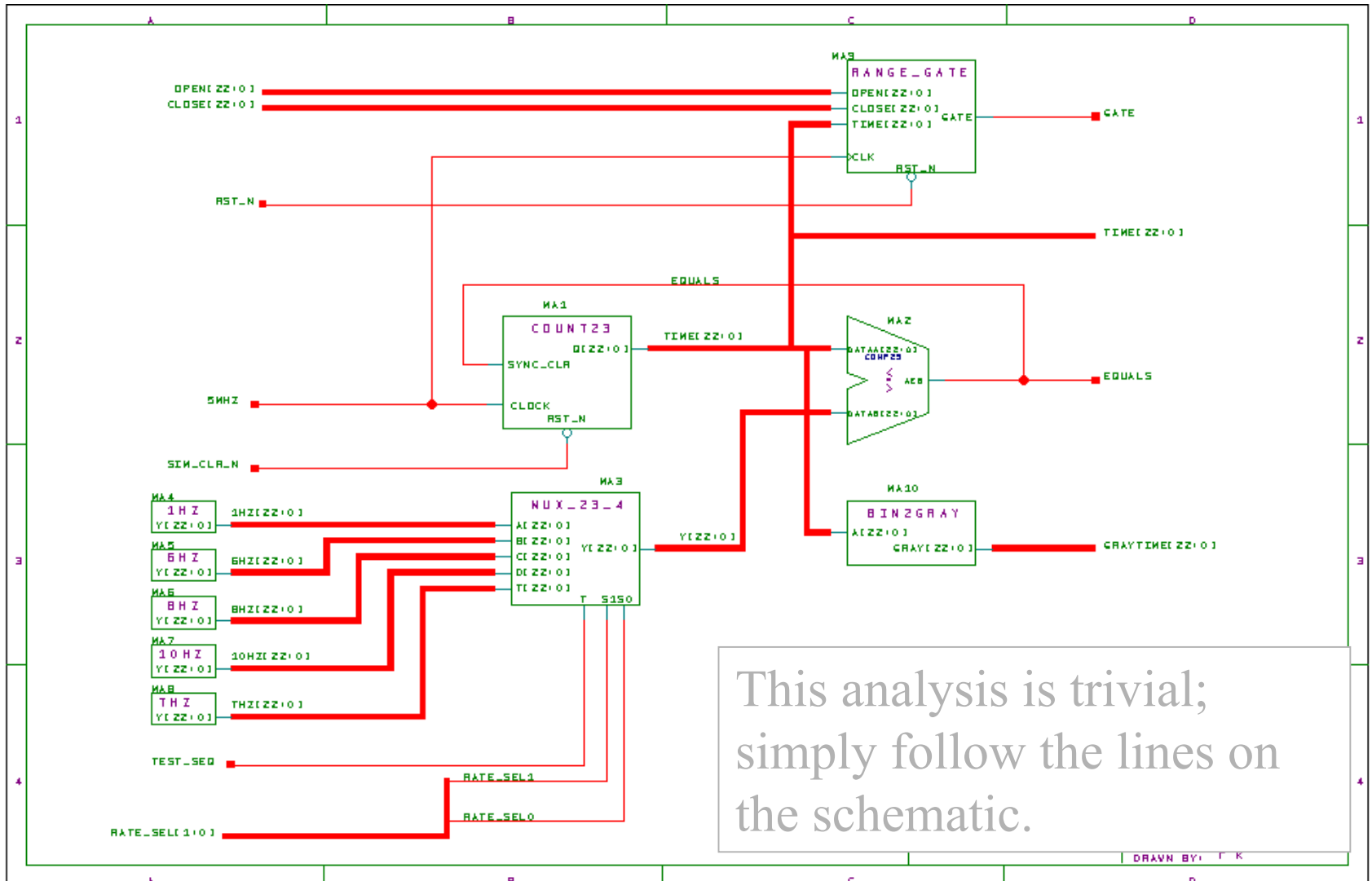
Master Sequencer Example

Schematic Analysis: Inputs and Outputs



Master Sequencer Example

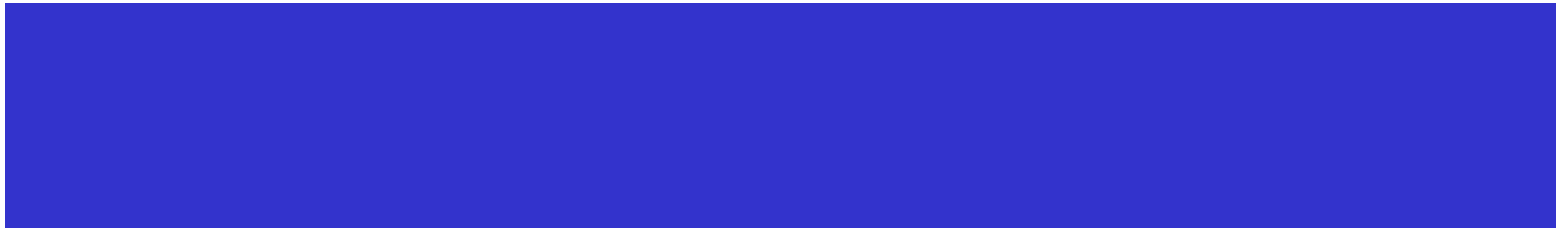
Schematic Analysis: Connectivity



Master Sequencer Example

Symbol Analysis: Inputs and Outputs

```
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
--  
--  
--  
entity SEQUENCER is
```



```
);  
end SEQUENCER;
```

In designing this slide I put the colored boxes in the wrong spot. An error-prone methodology!

Master Sequencer Example

VHDL Analysis: Inputs and Outputs

```
MA1: COUNT23 port map
  (RST_N => [REDACTED], SYNC_CLR => EQUALS_internal, CLOCK => [REDACTED]
  Q => TIME_NOW_internal
  );
MA3: MUX_23_4 port map
  (A => R1HZ, B => R6HZ, C => R8HZ, D => R10HZ, T => [REDACTED]
  T_SEL => [REDACTED], S1 => [REDACTED], S0 => [REDACTED],
  Y => Y
  );
MA2: COMP23 port map
  (DATAA => TIME_NOW_internal, DATAB=> Y,
  AEB => EQUALS_internal
  );
MA7: GATE_RANGE port map
  (RST_N => [REDACTED]
  CLOCK => [REDACTED]
  OPEN_VALUE => [REDACTED] CLOSE_VALUE => [REDACTED],
  TIME_NOW => TIME_NOW_internal,
  GATE => [REDACTED]
  );
MA8: BIN2GRAY23 port map
  (A => TIME_NOW_internal,
  Y => [REDACTED]
  );
```

Note: Had to print out the entity just to make this slide.

```
[REDACTED] <= EQUALS_internal;
[REDACTED] <= TIME_NOW_internal;
end RTL_ARCH;
```

 Inputs
 Outputs

Master Sequencer Example

VHDL Analysis: Connectivity

```
MA1: COUNT23 port map
  (RST_N => SIM_CLR_N, SYNC_CLR => EQUALS_internal, CLOCK => C5MHZ,
   Q => TIME_NOW_internal
  );
MA3: MUX_23_4 port map
  (A => R1HZ, B => R6HZ, C => R8HZ, D => R10HZ, T => THZ,
   T_SEL => TEST_SEQ, S1 => RATE_SEL(1), S0 => RATE_SEL(0),
   Y => Y
  );
MA2: COMP23 port map
  (DATAA => TIME_NOW_internal, DATAB => Y,
   AEB => EQUALS_internal
  );
MA7: GATE_RANGE port map
  (RST_N => RST_N,
   CLOCK => C5MHZ,
   OPEN_VALUE => OPEN_VALUE, CLOSE_VALUE => CLOSE_VALUE,
   TIME_NOW => TIME_NOW_internal,
   GATE => GATE
  );
MA8: BIN2GRAY23 port map
  (A => TIME_NOW_internal,
   Y => GRAYTIME
  );

EQUALS <= EQUALS_internal;
TIME_NOW <= TIME_NOW_internal;
end RTL_ARCH;
```

Making this chart was a lot of work and was error prone.

Master Sequencer Analysis

Some Observations (1)

- Block Level Inputs vs. Outputs
 - Schematic: Inputs on left of page; outputs on right of page.
 - VHDL: Listed in entity; all over the code.
- Internal Inputs vs. Outputs
 - Schematic: Inputs left, bottom, or top of symbol; outputs on right of symbol.
 - VHDL RTL: Left of assignment operator
 - VHDL Structural: All over the code

Master Sequencer Analysis

Some Observations (2)

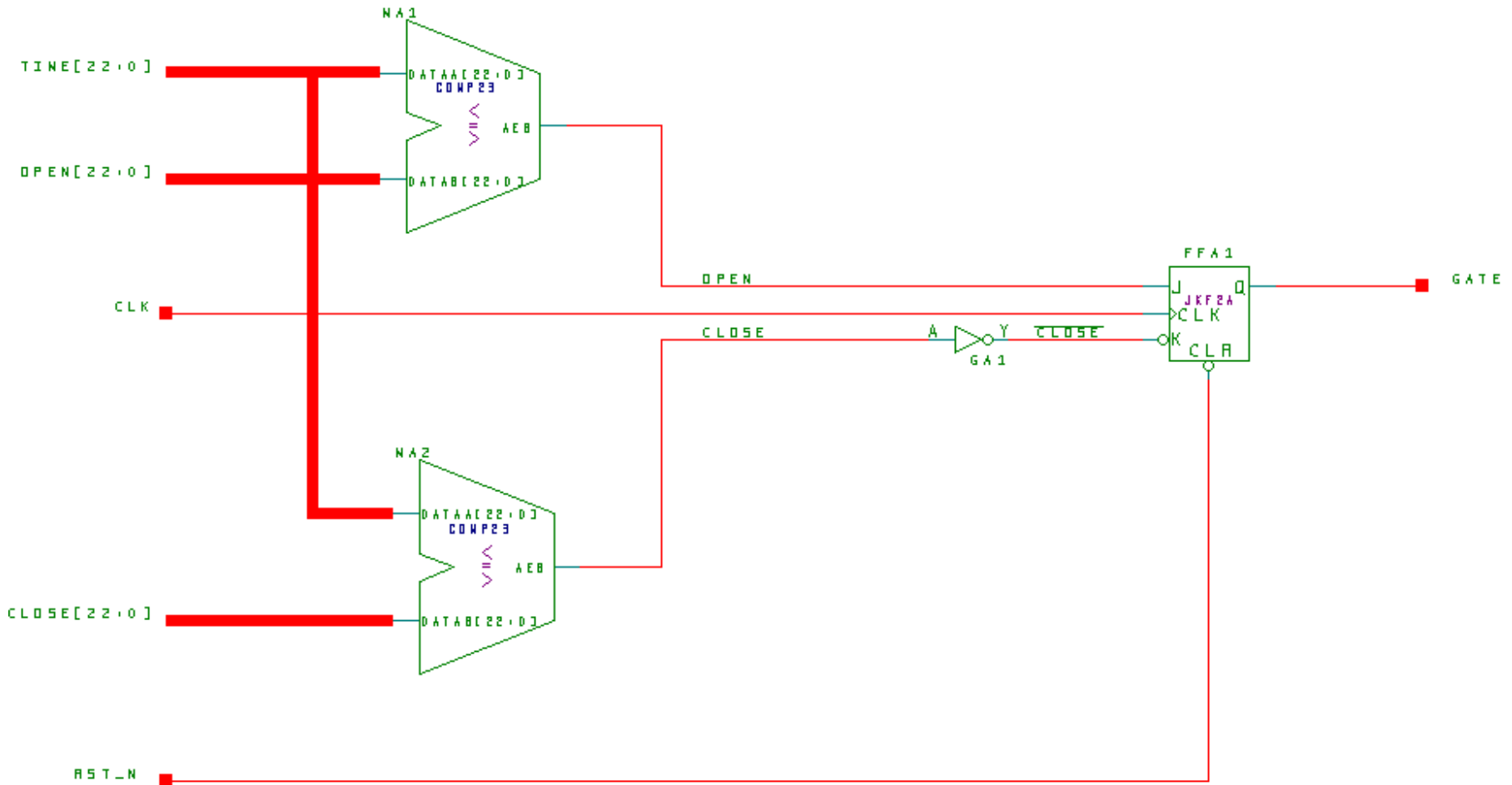
- Signal Connectivity
 - Schematic: Follow the nets and busses
 - VHDL: Draw “nets” with a highlighter and search with a text editor.

Range Gate Example

Open and close a “Gate” based on start and stop times programmed by an on-board computer.

Range Gate Example

Schematic Implementation



Range Gate Example

VHDL Implementation - Entity

```
library ieee;
use ieee.std_logic_1164.all;

--
-- This entity checks the current time against the OPEN value and CLOSE
-- value and sets the GATE output when the current time is in the range
-- between the OPEN value and the CLOSE value
--

entity GATE_RANGE is
    port( RST_N, CLOCK                : in  std_logic;
          OPEN_VALUE, CLOSE_VALUE, TIME_NOW : in  std_logic_vector(22 downto 0);
          GATE                            : out std_logic
        );

end GATE_RANGE;
```

Range Gate Example

VHDL Implementation - Architecture

```
architecture RTL_ARCH of GATE_RANGE is
signal GATE_REG, OPEN_NOW, CLOSE_NOW : std_logic;
begin

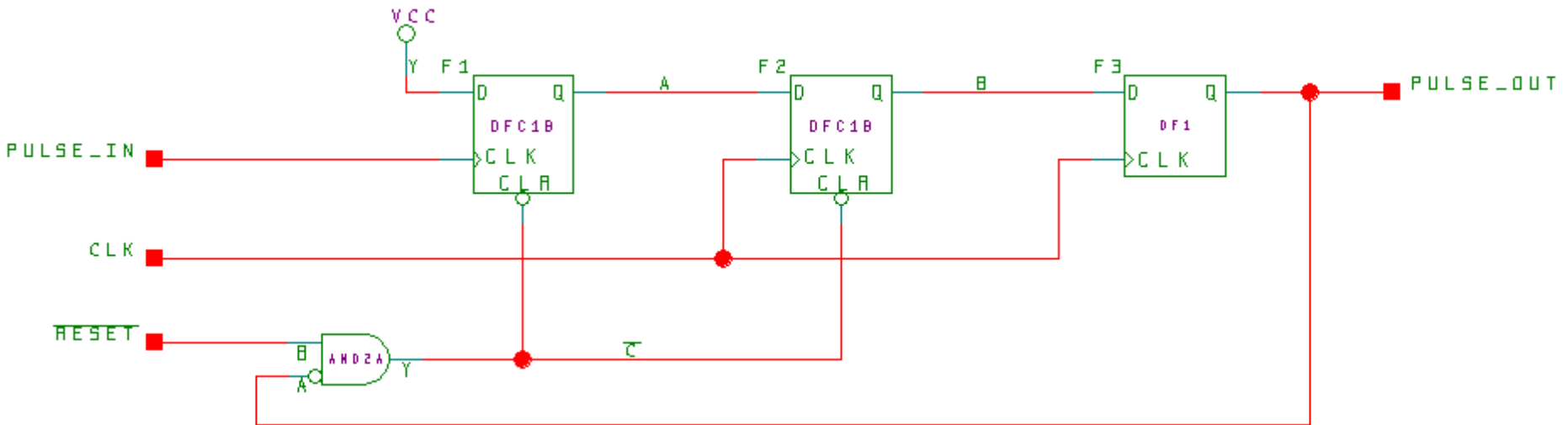
OPEN_NOW  <= '1' when (OPEN_VALUE=TIME_NOW)  else '0';
CLOSE_NOW <= '1' when (CLOSE_VALUE=TIME_NOW) else '0';

process (CLOCK, RST_N)
variable GATE_UPDATE : std_logic_vector(1 downto 0);
begin
if (RST_N='0') then
GATE_REG <= '0';
elsif (CLOCK'event and CLOCK='1') then
GATE_UPDATE := OPEN_NOW & CLOSE_NOW;
case (GATE_UPDATE) is
when "00" => GATE_REG <= GATE_REG;
when "01" => GATE_REG <= '0';
when "10" => GATE_REG <= '1';
when "11" => GATE_REG <= NOT GATE_REG;
when others => GATE_REG <= 'X';
end case;
end if;
end process;

GATE <= GATE_REG;

end RTL_ARCH;
```

Simpler Example: Synchronizer Schematic Version



Simpler Example: Synchronizer HDL Version

Class Exercise: Design the circuit on the preceding page in the HDL of your choice and we shall analyze it for clarity, correctness, and reviewability.

Conclusions (1a)

1. Top Level of the Hierarchy

The top level of the design should be a schematic giving the connectivity of the design. It is easy to see the the flow of signals in a schematic, graphically. This is particularly true for parallel and/or pipelined designs.¹

It is difficult to see the flow of signals in text, with the eye having to perform a pattern matching operation on signal names. Often, following the flow of signals in a structural HDL implementation results in the analyst drawing a schematic or block diagram by hand.

¹ See Ben Cohen's poster paper for coding recommendations.
Paper P30, "Coding HDL for Reviewability"

Conclusions (1b)

- Structural VHDL is similar (but worse!) than the GOTO in programming languages.
- In a programming language, targets of GOTOs are often easy to see graphically on the page, as they are in the left hand side, often separated by white space and delimited by a “:”.
- In VHDL, names are embedded inside parentheses and are tough to dig out and do pattern matching on, without making errors.

Case study: In a real design, there were two different signals with the following names:

CSEN

CS_EN

Conclusions (2)

2. HDL for Components

Components should normally be of the types that people can intuitively understand, such as counters, multiplexors, decoders, etc. VHDL code works well for these types of blocks, particularly considering design reuse.

VHDL also works well for expressing some complex state machines, with the CASE statement easily encoding old-fashioned flow charts. Ironically, for some simple state machines, VHDL is ill-suited for the task. (Examples to follow.)

3. HDL Component Complexity

HDL component descriptions should be no longer than one or two pages.

Conclusions (3)

4. HDL Component Data Sheets

Accompanying the design should be a "data book" giving a description of each component of the type one would find in a normal data book. HDL data sheets should include:

- Textual description
- Schematic or block diagram
- Timing diagram
- Description of algorithm

This data sheet for HDL components will aid in understanding the components' code by engineers writing test benches, reviewers, and analysts.

Conclusions (4)

5. Managing Complexity and Design Details

HDL allows designers to leave the confines of rigidly defined parts.

However, when constructing new components one shouldn't stray, too far from the components that we all are familiar with from our board design days. There's a reason why those parts are the ones the manufacturers provided.

In the same way that ANDs, ORs, and NOTs constitute a complete set for creating logic, higher level logic is made of things like muxes, counters, and decoders. While we may no longer be limited to choosing between 4-1, 8-1, and 16-1 muxes, we're still using muxes, everyone understands how they work, and there's probably some cosmic reason why the common MSI structures are the ones that have endured.

The Conclusion

There may be some who don't want to be confined to this or another similar type of coding style. There will always be the class of designers, such as C programmers, who take pride in writing the most obtuse code as possible and view this as a point of pride and manliness.

These designers should not be designing safety critical systems, weapons systems, or machines that our nations build as a matter of pride.

“At first sight, the idea of any rules or principles being superimposed on the creative mind seems more likely to hinder than to help, but this is quite untrue in practice. Disciplined thinking focuses inspiration rather than blinkers it.”

This quote starts Chapter 2, “Design Considerations,” in The Well-Tempered Digital Design, Robert B. Seidensticker, 1986.

Designing for Reviews

In high reliability design, the review and analysis of the design is of equal importance to the design itself.

All designers will make errors. A design that is not reviewed is at risk.

Designs that are too difficult to be reviewed will either not be reviewed well or at all. Under these circumstances failure then is not an option; it is a matter of when, not if.

From this it follows that the design description should be as easy to review as possible. This would eliminate FPGAs described in 33 pages of “VHDL spaghetti.” It will also eliminate FPGAs that start out with 16 pages of structural VHDL with no comments. These are not contrived examples; these are real flight designs that we have reviewed, as we showed in the beginning of this seminar.

Should I Use a Schematic Or an HDL for an FSM Design?

- Depends on the structure of the circuit you are designing.
- What form would you use if you had to explain the circuit to someone else?
 - One engineer had a generic VHDL flip-flop for his design. In the peer-review, he explained it using a schematic.
- What for would you use if someone had to pick up the circuit without you there?