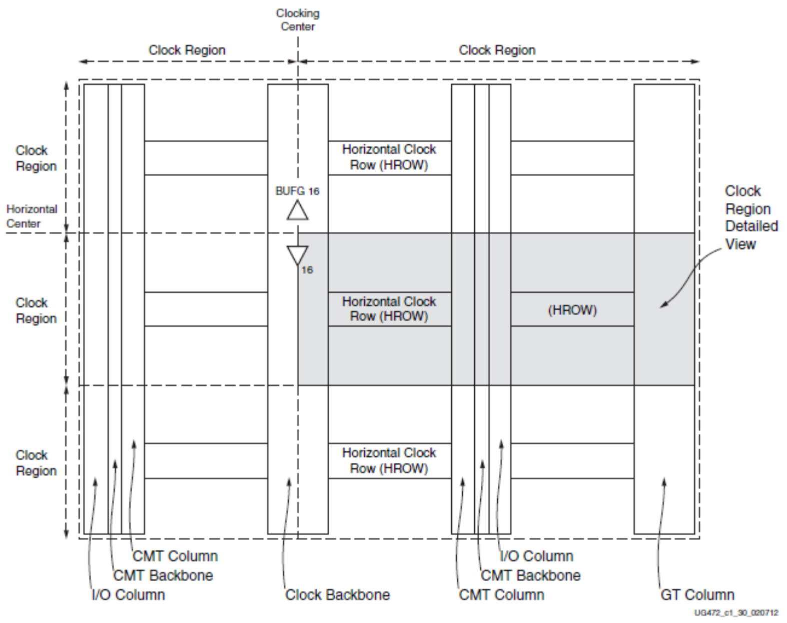
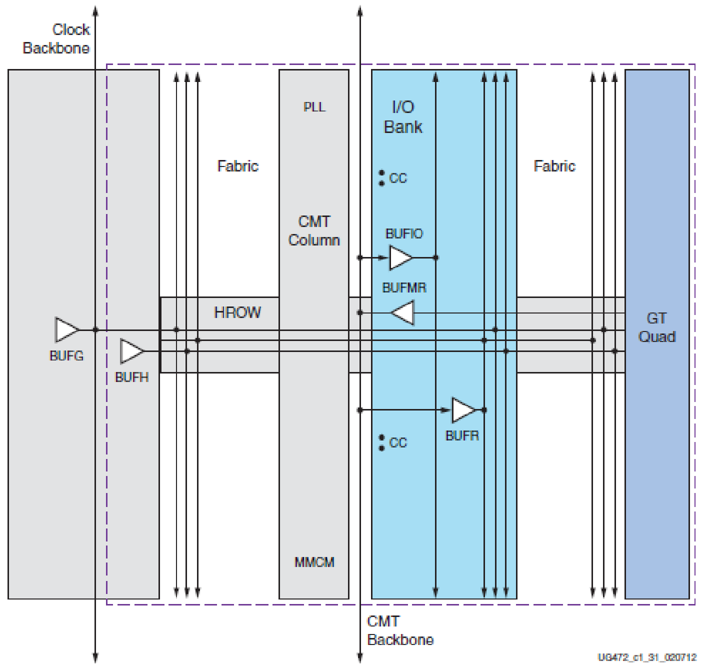
Series 7 Clocking

- Clocks need to be treated differently than other signals within the device.
  - Globally accessible
  - Minimal offset across the entire device
  - Handle high fan out
- Dedicated Resources to generating and routing clock signals.
- Xilinx UG 472

- Incorrect understanding and use of clocks is one of the main causes in loss of coherency/stability in digital systems.
  - SNR is significantly reduced when coherency is lost.

High Level Clock Architecture

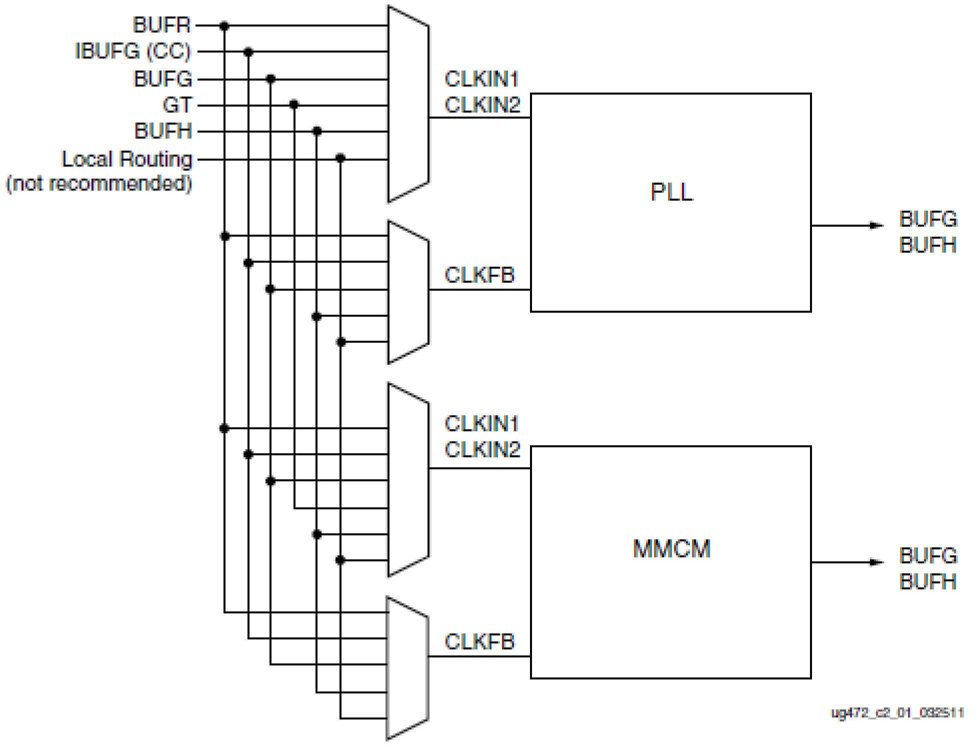


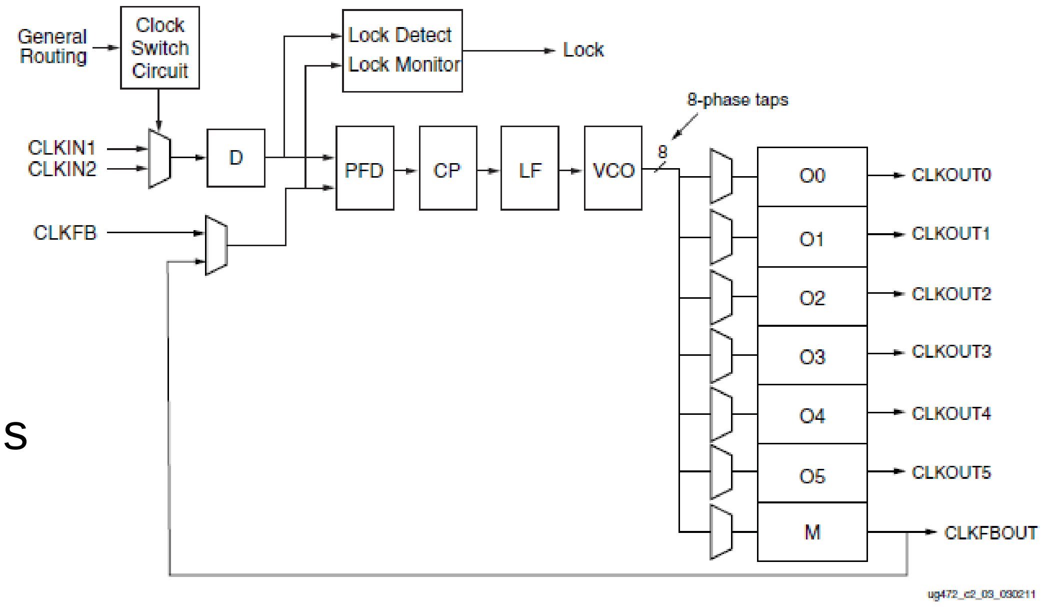


- 1 to 24 Clock Regions (Horizontal)
- A Clock Region Contains
  - 50x CLBs per column
  - 10x 36k Block RAMs
  - 20x DSP Slices
  - 12x BUFHs to Route clocks across the horizontal clock region
  - MMCM and PLL
    - Mixed-Mode Clock Manager
    - Phase Locked Loop
- BUFG are used to route clocks throughout the entire device.

MMCM and PLL

- Very Similar
  - Capability to generate various frequencies and phase offsets between multiple clocks.
  - Why do we need different clocks.
    - Decimation
    - Different timing requirements
    - DDR, ...
    - Gigabit Transcievers.
    - IO serializers and deserializers





Load RAM Contents from a file

- Easy in Verilog
- Use the following statement to init.
  `$readmemh(`*`filename`*`);`

- Need to define a function in VHDL.

rom.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use std.textio.all;

entity rom is
  port (clock  : in std_logic;
        addr   : in std_logic_vector(4 downto 0);
        data   : out std_logic_vector(15 downto 0));
end rom;

architecture behavior of rom is

  type mem_type is array (29 downto 0) of std_logic_vector (15 downto 0);

  impure function load_mem(file_name : in string) return mem_type is

    FILE ram_file : text is in file_name;
    variable ram_line : line;
    variable temp_value : bit_vector(15 downto 0);
    variable ram : mem_type;

    begin

      for i in mem_type'range loop

        readline(ram_file, ram_line);
        read(ram_line, temp_value);
        ram(i) := to_stdlogicvector(temp_value);

      end loop;

    return ram;

  end function;

  signal mem: mem_type := load_mem("mem.txt");

begin

  process (clock)
  begin
    if (clock'event and clock = '1') then
      data <= mem(conv_integer(addr)) ;
    end if;
  end process;

end behavior;
```

rom.vhd

```
use std.textio.all;

impure function load_mem(file_name : in string) return mem_type is

  FILE ram_file : text is in file_name;
  variable ram_line : line;
  variable temp_value : bit_vector(15 downto 0);
  variable ram : mem_type;

  begin

    for i in mem_type'range loop

      readline(ram_file, ram_line);
      read(ram_line, temp_value);
      ram(i) := to_stdlogicvector(temp_value);

    end loop;

  return ram;

end function;
```

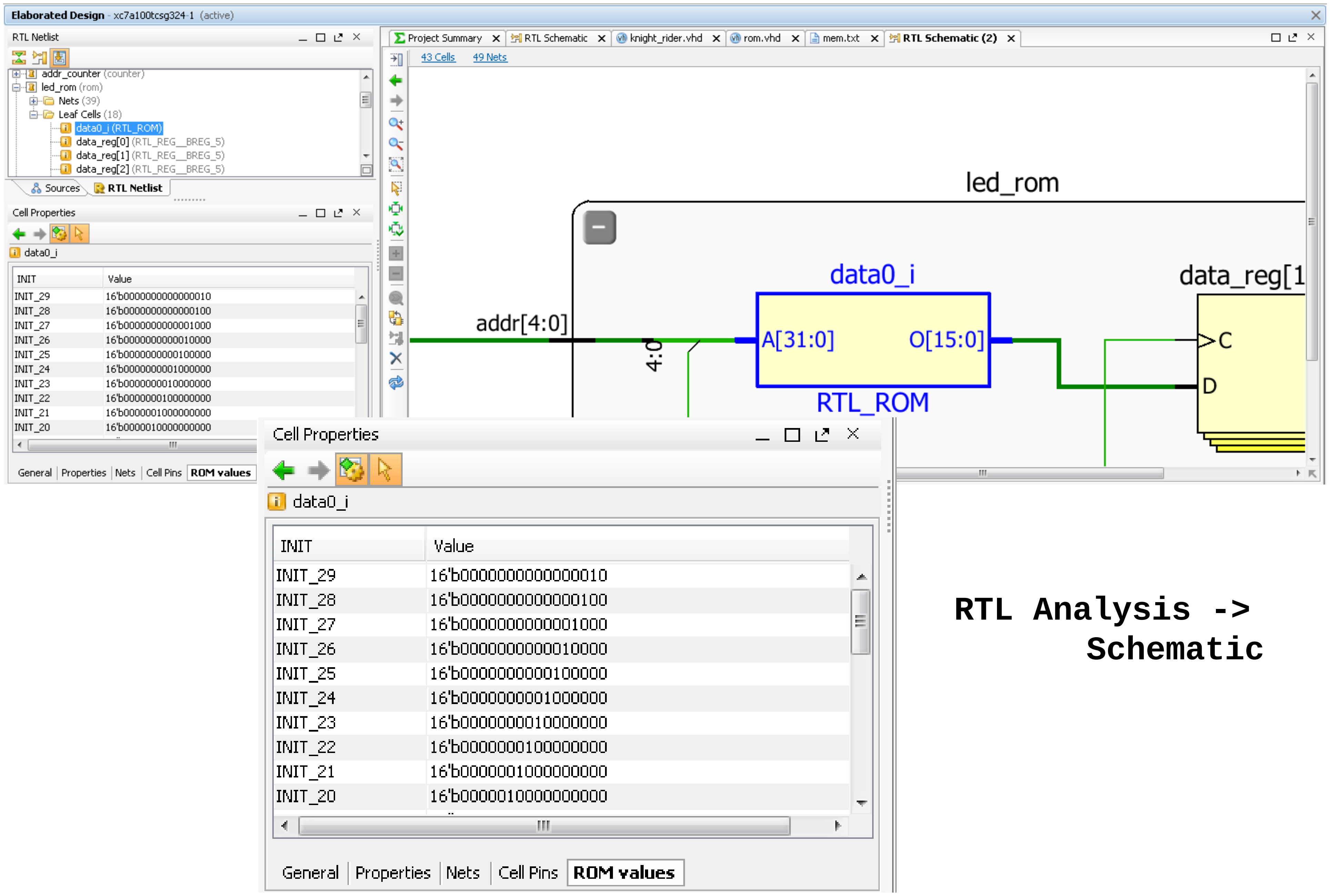
Initializatins

```
signal mem: mem_type := load_mem("mem.txt");
```

mem.txt contents:

```
0000000000000010
0000000000000100
0000000000001000
0000000000010000
0000000000100000
0000000001000000
0000000010000000
0000000100000000
0000001000000000
0000010000000000
0000100000000000
0001000000000000
0010000000000000
0100000000000000
1000000000000000
0100000000000000
0010000000000000
0001000000000000
0000100000000000
0000010000000000
0000001000000000
0000000100000000
0000000010000000
0000000001000000
0000000000100000
0000000000010000
0000000000001000
0000000000000100
0000000000000010
0000000000000001
```

Verifying ROM Contents



INIT	Value
INIT_29	16'b0000000000000010
INIT_28	16'b0000000000000100
INIT_27	16'b0000000000001000
INIT_26	16'b0000000000010000
INIT_25	16'b0000000000100000
INIT_24	16'b0000000001000000
INIT_23	16'b0000000010000000
INIT_22	16'b0000000100000000
INIT_21	16'b0000001000000000
INIT_20	16'b0000010000000000

RTL Analysis -> Schematic

What about larger files?

- Use another program to generate the contents.
- Matlab for example.

```
fid = fopen('mem.txt','w');
for i = 1:15,
    fprintf(fid,'%s\r\n',dec2bin(2^i,16));
end;
for i = 14:-1:0,
    fprintf(fid,'%s\r\n',dec2bin(2^i,16));
end;
fclose(fid);
```