```
-- Memory_MUX
-- MUX THAT SELECTS BETWEEN THE CCM or I2C Controller or AD_Controller or Error reporter
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
library synplify;
use synplify.attributes.all;

entity Memory_MUX is port

(
-- MUX controls
AD_RUN                  : in std_logic;
OPERATE                 : in std_logic;
--REGISTER_ERROR          : in std_logic;


-- I2C_Controls
I2C_RAM_ADDRESS         : in std_logic_vector(4 downto 0);
I2C_RAM_DATA_D          : in std_logic_vector(7 downto 0);
I2C_RAM_WRITE_ENABLE    : in std_logic;
I2C_RAM_READ_ENABLE     : in std_logic;
I2C_RAM_DATA_Q          : out std_logic_vector(7 downto 0);

-- CCM_Controls
CCM_RAM_ADDRESS         : in std_logic_vector(4 downto 0);
CCM_RAM_DATA_D          : in std_logic_vector(7 downto 0);
CCM_RAM_WRITE_ENABLE    : in std_logic;
CCM_RAM_READ_ENABLE     : in std_logic;
CCM_RAM_DATA_Q          : out std_logic_vector(7 downto 0);

-- error_Controls
--ERROR_RAM_ADDRESS         : in std_logic_vector(5 downto 0);
--ERROR_RAM_DATA_D          : in std_logic_vector(7 downto 0);
--ERROR_RAM_WRITE_ENABLE    : in std_logic;
--ERROR_RAM_READ_ENABLE     : in std_logic;
--ERROR_RAM_DATA_Q          : out std_logic_vector(7 downto 0);

-- AD_Controls
AD_RAM_ADDRESS          : in std_logic_vector(4 downto 0);
AD_RAM_DATA_D           : in std_logic_vector(7 downto 0);
AD_RAM_WRITE_ENABLE     : in std_logic;

-- mux outputs
RAM_ADDRESS             : out std_logic_vector(4 downto 0);
RAM_DATA_D              : out std_logic_vector(7 downto 0);
RAM_WRITE_ENABLE        : out std_logic;
RAM_READ_ENABLE         : out std_logic;
RAM_DATA_Q              : in std_logic_vector(7 downto 0));
-----------------------------------------------------------------

end Memory_MUX;


architecture rtl of Memory_MUX is
attribute syn_radhardlevel of rtl : architecture is "tmr";

begin

    I2C_RAM_DATA_Q      <=   RAM_DATA_Q;
    CCM_RAM_DATA_Q      <=   RAM_DATA_Q;
    --ERROR_RAM_DATA_Q    <=   RAM_DATA_Q;

      --RAM_ADDRESS <=    --ERROR_RAM_ADDRESS when (REGISTER_ERROR = '1') else
      RAM_ADDRESS <=    I2C_RAM_ADDRESS when (AD_RUN = '0' and  OPERATE = '1') else
                        AD_RAM_ADDRESS  when (AD_RUN = '1' and  OPERATE = '0') else
                        CCM_RAM_ADDRESS when (AD_RUN = '0' and  OPERATE = '0') else
```

```
                        CCM_RAM_ADDRESS when (AD_RUN = '1' and  OPERATE = '1');

      --RAM_DATA_D  <=    --ERROR_RAM_DATA_D  when (REGISTER_ERROR = '1') else
      RAM_DATA_D  <=    I2C_RAM_DATA_D  when (AD_RUN = '0' and  OPERATE = '1') else
                        AD_RAM_DATA_D   when (AD_RUN = '1' and  OPERATE = '0') else
                        CCM_RAM_DATA_D  when (AD_RUN = '0' and  OPERATE = '0') else
                        CCM_RAM_DATA_D  when (AD_RUN = '1' and  OPERATE = '1');


      --RAM_WRITE_ENABLE <=   --ERROR_RAM_WRITE_ENABLE when (REGISTER_ERROR = '1') else
      RAM_WRITE_ENABLE <=   I2C_RAM_WRITE_ENABLE when (AD_RUN = '0' and  OPERATE = '1') el
se
                            AD_RAM_WRITE_ENABLE  when (AD_RUN = '1' and  OPERATE = '0') el
se
                            CCM_RAM_WRITE_ENABLE when (AD_RUN = '0' and  OPERATE = '0') el
se
                            CCM_RAM_WRITE_ENABLE when (AD_RUN = '1' and  OPERATE = '1');

      --RAM_READ_ENABLE  <=   ERROR_RAM_READ_ENABLE  when (REGISTER_ERROR = '1') else
      RAM_READ_ENABLE  <=   I2C_RAM_READ_ENABLE  when (AD_RUN = '0' and  OPERATE = '1') el
se
                            '0'                  when (AD_RUN = '1' and  OPERATE = '0') el
se
                            CCM_RAM_READ_ENABLE  when (AD_RUN = '0' and  OPERATE = '0') el
se
                            CCM_RAM_READ_ENABLE  when (AD_RUN = '1' and  OPERATE = '1');

 end rtl;
```