Locality of Reference:

Locality of reference is a key concept in the design and operation of cache memory which describes the pattern of memory access in programs, where certain types of memory locations are more likely to be accessed than others, either in the near future or frequently.
The two main types of locality of reference are **Temporal Locality** and **Spatial Locality**.

Temporal Locality:

- If a piece of data is accessed once, it is likely to be accessed again in the near future.
- Cache memory exploits this by storing recently accessed data, making it available for faster retrieval if needed again soon.

Spatial Locality:

- If a particular memory location is accessed, nearby memory locations are likely to be accessed as well.
- Cache memory takes advantage of this by storing not only the requested data but also the surrounding data.
- This is based on the observation that programs often access data stored in contiguous memory locations.

Cache memory is designed to leverage both temporal and spatial locality to increase the efficiency of data retrieval.

VHDL Limitation

- **Complexity:** VHDL is a complex language that requires a strong understanding of digital design concepts in order to use it effectively. This can make it difficult for designers who are new to digital design to learn and use VHDL.
- **Verbosity:** VHDL is a verbose language which requires a lot of code to describe even simple designs. This can make it time-consuming to write and debug VHDL code, particularly for large and complex designs.
- **Execution speed:** VHDL simulations can be slow, particularly for large and complex designs. This can make it difficult to use VHDL for real-time simulation or rapid prototyping.

Symbols for AND, OR and NOT functions:

If inputs are A and B, output is Y, Then

AND Functions: Y = A && B;

OR Functions: Y = A || B;

NOT Functions: Y = !A;

VHDL code for a 2-input AND gate:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Entity declaration
entity AND_Gate is
   Port (
      A : in  STD_LOGIC;
      B : in  STD_LOGIC;
      Y : out STD_LOGIC
   );
end AND_Gate;

-- Architecture body
architecture Behavioral of AND_Gate is
begin
   -- Process block for the AND gate logic
   Y <= A and B;
end Behavioral;
```

VHDL code for a 2-input OR gate:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Entity declaration
entity OR_Gate is
   Port (
      A : in  STD_LOGIC;
      B : in  STD_LOGIC;
      Y : out STD_LOGIC
   );
end OR_Gate;

-- Architecture body
architecture Behavioral of OR_Gate is
begin
   -- Process block for the OR gate logic
   Y <= A or B;
end Behavioral;
```

VHDL code for a NOT gate:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Entity declaration
entity NOT_Gate is
   Port (
      A : in  STD_LOGIC;
      Y : out STD_LOGIC
   );
end NOT_Gate;

-- Architecture body
architecture Behavioral of NOT_Gate is
begin
   -- Process block for the NOT gate logic
   Y <= not A;
end Behavioral;
```

VHDL code for a 2x1 multiplexer:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Entity declaration
entity Mux2x1 is
   Port (
      A : in  STD_LOGIC;  -- Input 1
      B : in  STD_LOGIC;  -- Input 2
      Sel : in  STD_LOGIC;  -- Select line
      Y : out STD_LOGIC  -- Output
   );
end Mux2x1;

-- Architecture body
architecture Behavioral of Mux2x1 is
begin
   -- Process block for the MUX logic
   process(A, B, Sel)
   begin
      if Sel = '0' then
         Y <= A;
      else
         Y <= B;
      end if;
   end process;
end Behavioral;
```

VHDL code for a 1x2 demultiplexer:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Entity declaration
entity Demux1x2 is
   Port (
      D : in  STD_LOGIC;   -- Input data
      Sel : in  STD_LOGIC; -- Select line
      Y0 : out STD_LOGIC;  -- Output 0
      Y1 : out STD_LOGIC   -- Output 1
   );
end Demux1x2;

-- Architecture body
architecture Behavioral of Demux1x2 is
begin
   -- Process block for the Demux logic
   process(D, Sel)
   begin
      if Sel = '0' then
         Y0 <= D;
         Y1 <= '0';
      else
         Y0 <= '0';
         Y1 <= D;
      end if;
   end process;
end Behavioral;
```

VHDL code for a 4x2 binary encoder:

This encoder converts 4 input signals into a 2-bit binary output.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Entity declaration
entity Encoder4to2 is
   Port (
      D0 : in  STD_LOGIC;  -- Input 0
      D1 : in  STD_LOGIC;  -- Input 1
      D2 : in  STD_LOGIC;  -- Input 2
      D3 : in  STD_LOGIC;  -- Input 3
      Y0 : out STD_LOGIC;  -- Output bit 0
      Y1 : out STD_LOGIC   -- Output bit 1
   );
end Encoder4to2;

-- Architecture body
architecture Behavioral of Encoder4to2 is
begin
   -- Process block for the Encoder logic
   process(D0, D1, D2, D3)
   begin
      -- Default outputs
      Y0 <= '0';
      Y1 <= '0';

      if D0 = '1' then
         Y0 <= '0';
         Y1 <= '0';
      elsif D1 = '1' then
         Y0 <= '1';
         Y1 <= '0';
      elsif D2 = '1' then
         Y0 <= '0';
         Y1 <= '1';
      elsif D3 = '1' then
         Y0 <= '1';
         Y1 <= '1';
      end if;
   end process;
end Behavioral;
```

VHDL code for a 2x4 binary decoder:

This decoder converts a 2-bit binary input into one of four outputs.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Entity declaration
entity Decoder2to4 is
  Port (
    A0 : in  STD_LOGIC;  -- Input bit 0
    A1 : in  STD_LOGIC;  -- Input bit 1
    Y0 : out STD_LOGIC;  -- Output 0
    Y1 : out STD_LOGIC;  -- Output 1
    Y2 : out STD_LOGIC;  -- Output 2
    Y3 : out STD_LOGIC   -- Output 3
  );
end Decoder2to4;

-- Architecture body
architecture Behavioral of Decoder2to4 is
begin
  -- Process block for the Decoder logic
  process(A0, A1)
  begin
    -- Default outputs to '0'
    Y0 <= '0';
    Y1 <= '0';
    Y2 <= '0';
    Y3 <= '0';

    -- Decode logic
    if (A0 = '0' and A1 = '0') then
      Y0 <= '1';  -- If input is 00, Y0 is activated
    elsif (A0 = '1' and A1 = '0') then
      Y1 <= '1';  -- If input is 01, Y1 is activated
    elsif (A0 = '0' and A1 = '1') then
      Y2 <= '1';  -- If input is 10, Y2 is activated
    elsif (A0 = '1' and A1 = '1') then
      Y3 <= '1';  -- If input is 11, Y3 is activated
    end if;
  end process;
end Behavioral;
```