**3. Design entities and configurations**

3.2 ) Entity Declerations

An **entity** in VHDL defines the **interface** of a hardware design unit with the external world. It specifies the input and output signals and provides a framework for how the design interacts with its environment. In short, an **entity** describes **what** the design does, while the **architecture** defines **how** it does it.

3.3) Architecture bodies

Entities and architectures serve different roles. Multiple architectures can be associated with a single entity, and each represents a different design approach for that entity. However, architectures defined for different entities can have the same name.

3.4) Configuration declarations

NOTE4)!!A block configuration is allowed to appear immediately within a configuration declaration only if the entity declaration denoted by the entity name of the enclosing configuration declaration has associated architectures.

\*\* Block Configuration Example   
configuration Config1 of TopLevel is

for Structural

for **AdderBlock**(Block named it defines the configuration) : Adder(0 to 3)

use entity work.AdderEntity(AdderArch);

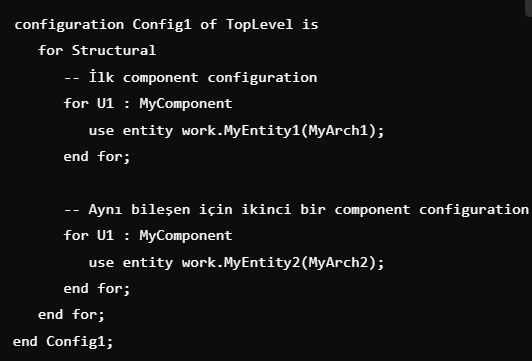
end for;

end for;

end Config1;

3.4.3)Component Configuration

!! It is an error if two component configurations apply to the same component instance.

 metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Code 1 Code 2

**4. Subprograms**

4.1) General

Subprograms define algorithms for computing values or exhibiting behavior, while packages provide a way for different design units or parts of a unit to share these and other resources through common declarations.

There are two forms of subprograms

1. Procedure (it can be used to clk)
2. Function (it can’t be used to clk )

Function: The return type of the function must match the type of the declaration it is assigned to.

\* Subprograms are classified based on their headers as simple, uninstantiated, and generic-mapped subprograms.

\* Uninstantiated subprograms have limited usage; they cannot be called directly from outside and are restricted to specific purposes.

\* Generic lists and mapping aspects determine how a subprogram is defined and how it can be used.

4.2.2.2) Constant and variable parameters

Variable parameters can be changed by the subprograms

\* \* **Copying a parameter** refers to the process where the values of parameters passed to a subprogram are copied.

\* \* Changes made within the subprogram occur on the copy, and if necessary, these changes are copied back to the original parameter (in the case of inout or out mode).

\*\* This approach helps control the impact of subprograms on external data and enhances the reliability of the design.

4.2.2.3) Signal parameters

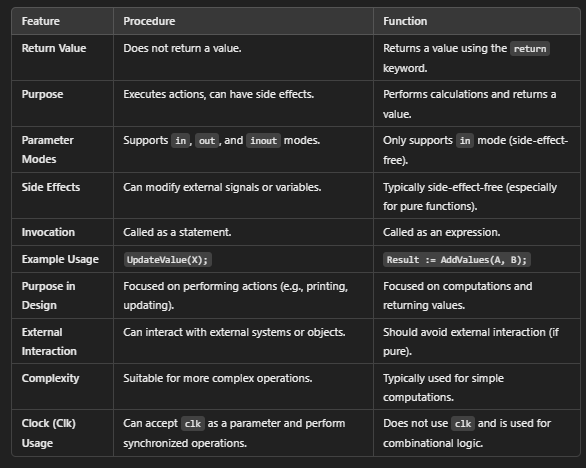
**Formal signal parameters**: Signals passed into a subprogram as parameters.

**Signal-valued attributes**:

* 'STABLE: Checks how long a signal has maintained its value.
* 'QUIET: Checks how long a signal has not had an event.
* 'TRANSACTION: Tracks changes to a signal.
* 'DELAYED: Represents a delayed version of the signal.

4.5.3) Signatures

\*\* If the reserved word return is present, the subprogram is a function and the base type of the type mark following the reserved word in the signature is the same as the base type of the return type of the function, or the reserved word return is absent and the subprogram is a procedure.



Diffrent Table 1

4.6) Resolution functions

A resolution function determines how the values from multiple sources driving a signal are combined into a single resolved value.