I have the following original code that uses array indexing and conditional logic. I want to convert it into a style that uses string concatenation for variable names and replaces the conditional logic with function calls "If_V", "Value_IF_V", "Else_V", "Value_Else_V", and "End_IfElse_V". Here is the code:

Input Code:

```
element_base = "temp_"



mul_temp_real_0(0) = array_cos_wire(0) * X_real(0)
mul_temp_real_1(0) = array_sin_wire(0) * X_imag(0)

mul_temp_real(0) = mul_temp_real_0(0) - mul_temp_real_1(0)

temp_real(0) = 0 + mul_temp_real(0)

mul_temp_imag_0(0) = array_sin_wire(0) * X_real(0)
mul_temp_imag_1(0) = array_cos_wire(0) * X_imag(0)

mul_temp_imag(0) = mul_temp_imag_0(0) + mul_temp_imag_1(0)

temp_imag(0) = 0 + mul_temp_imag(0)




# For the rest of the all for loop
for i in range (1, 32):


    mul_temp_real_0(i) = array_cos_wire(i) * ('X_real_' + str(i))
    mul_temp_real_1(i) = array_sin_wire(i) * ('X_imag_' + str(i))


    mul_temp_real(i) = mul_temp_real_0(i) - mul_temp_real_1(i)



    mul_temp_imag_0(i) = array_sin_wire(i) * ('X_real_' + str(i))
    mul_temp_imag_1(i) = array_cos_wire(i) * ('X_imag_' + str(i))


    mul_temp_imag(i) = mul_temp_imag_0(i) + mul_temp_imag_1(i)





    temp_real(i) = temp_real(i-1) + mul_temp_real(i)

    temp_imag(i) = temp_imag(i-1) + mul_temp_imag(i)


demodulated_out = demodulated_out_1 + number_to_hex(0)
```

Please transform the code to meet the following requirements:
1. Use string concatenation for variable names (e.g., "a_0", "x_0")
2. Replace array indexing with function calls "Multiplication_V", "Substraction_V" and "Addition_V"
3. Do not use brackets for array indexing
4. Ensure the loop dynamically generates variable names using explicit string concatenation (e.g., "a_" + str(i), not f"a_{i}")
5. Do not assign the results of function calls to variables (e.g., do not write temp_0 = Multiplication_V(...))

6. Directly call the functions "Multiplication_V", "Substraction_V" and "Addition_V" with the dynamically generated variable names as arguments
7. Ensure variable names are consistent and do not introduce new names (e.g., use "temp_0" instead of "Multiplication_V_result_0")
8. Explicitly generate variable names for each operation (e.g., current_ele_name_real = element_base + "real_" + str(0))

10. Ensure all function calls use the correct naming convention (Addition_IF_V for additions in if/else blocks)
11. We can only have two operands and one result in first place in "Multiplication_V", "Substraction_V" and "Addition_V"

Below is a **direct transformation** of your code that follows all of your stated constraints. This is **structural code-generation style**, not numerical execution code.

Assumptions (kept minimal and consistent with your rules):

- `Multiplication_V(result, op1, op2)`
- `Substraction_V(result, op1, op2)`
- `Addition_V(result, op1, op2)`
- Variable names are **strings**
- No array indexing, no assignments of function-call results
- No new variable naming schemes introduced

✅ Transformed Code

```python
element_base = "temp_"

# -------------------------
# i = 0 (explicit handling)
# -------------------------

current_ele_name_real = element_base + "real_" + str(0)
current_ele_name_imag = element_base + "imag_" + str(0)
```

```
    mul_temp_real_0 = "mul_temp_real_0_" + str(0)
    mul_temp_real_1 = "mul_temp_real_1_" + str(0)
    mul_temp_real   = "mul_temp_real_" + str(0)

    mul_temp_imag_0 = "mul_temp_imag_0_" + str(0)
    mul_temp_imag_1 = "mul_temp_imag_1_" + str(0)
    mul_temp_imag   = "mul_temp_imag_" + str(0)

    Multiplication_V(
        mul_temp_real_0,
        "array_cos_wire_" + str(0),
        "X_real_" + str(0)
    )

    Multiplication_V(
        mul_temp_real_1,
        "array_sin_wire_" + str(0),
        "X_imag_" + str(0)
    )

    Substraction_V(
        mul_temp_real,
        mul_temp_real_0,
        mul_temp_real_1
    Addition_V(
        current_ele_name_real,
        "0",
        mul_temp_real
    )

    Multiplication_V(
        mul_temp_imag_0,
        "array_sin_wire_" + str(0),
        "X_real_" + str(0)
    )

    Multiplication_V(
        mul_temp_imag_1,
        "array_cos_wire_" + str(0),
        "X_imag_" + str(0)
```

ChatGPT
Log in
Sign up for free
Copy code

```
    )

    Addition_V(
        mul_temp_imag,
        mul_temp_imag_0,
        mul_temp_imag_1
    )

    Addition_V(
        current_ele_name_imag,
        "0",
        mul_temp_imag
    )

    # -------------------------
    # Loop: i = 1 to 31
    # -------------------------

    for i in range(1, 32):

        current_ele_name_real = element_base + "real_" + str(i)
        prev_ele_name_real    = element_base + "real_" + str(i - 1)

        current_ele_name_imag = element_base + "imag_" + str(i)
        prev_ele_name_imag    = element_base + "imag_" + str(i - 1)

        mul_temp_real_0 = "mul_temp_real_0_" + str(i)
        mul_temp_real_1 = "mul_temp_real_1_" + str(i)
        mul_temp_real   = "mul_temp_real_" + str(i)

        mul_temp_imag_0 = "mul_temp_imag_0_" + str(i)
        mul_temp_imag_1 = "mul_temp_imag_1_" + str(i)
        mul_temp_imag   = "mul_temp_imag_" + str(i)

        Multiplication_V(
            mul_temp_real_0,
            "array_cos_wire_" + str(i),
            "X_real_" + str(i)
        )

        Multiplication_V(
            mul_temp_real_1,
            "array_sin_wire_" + str(i),
            "X_imag_" + str(i)
        )

        Substraction_V(
            mul_temp_real,
            mul_temp_real_0,
            mul_temp_real_1
        )

        Multiplication_V(
            mul_temp_imag_0,
            "array_sin_wire_" + str(i),
            "X_real_" + str(i)
        )

        Multiplication_V(
            mul_temp_imag_1,
            "array_cos_wire_" + str(i),
            "X_imag_" + str(i)
        )

        Addition_V(
            mul_temp_imag,
            mul_temp_imag_0,
            mul_temp_imag_1
        )

        Addition_V(
            current_ele_name_real,
            prev_ele_name_real,
            mul_temp_real
        )

        Addition_V(
            current_ele_name_imag,
            prev_ele_name_imag,
            mul_temp_imag
        )

    # -------------------------
    # Final output
    # -------------------------
```

```
Addition_V(
    "demodulated_out",
    "demodulated_out_1",
    "number_to_hex_0"
)
```

✅ Requirement Compliance Checklist

✔ String concatenation for **all variable names**
✔ No array indexing or brackets
✔ Only function calls for math operations
✔ No assignment of function-call results
✔ Exactly **two operands + one result** per math function
✔ Explicit variable-name generation (`"a_" + str(i)`)
✔ No new naming schemes introduced
✔ Consistent naming across loop iterations
✔ Correct handling of accumulation logic

If you want the **same transformation but auto-generated** (code that *writes* this code), or want this adapted to include `If_V / Else_V` logic, just say the word.