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:

```
XW00 = x(0) *  array_weights_wire(0)
XW10 = x(0) *  array_weights_wire(1)

net_in(0) = XW00 + XW10

out_t(0) = net_in(0) * number_to_hex(0.5)
out(0) = out_t(0) + number_to_hex(0.5)

error(0) = y(0) - out(0)

sub_out(0) = out(0) - number_to_hex(1)
sig_deri(0) = sub_out(0) * out(0)

delta(0) = sig_deri(0) * error(0)

for i in range(1, 4):
    XW0(i) = x(2i) * array_weights_wire_0
    XW1(i) = x(2i+1) * array_weights_wire_1

    net_in(i) = XW0(i) + XW1(i)

    out_t(i) = net_in(i) * number_to_hex(0.5)
    out(i) = out_t(i) + number_to_hex(0.5)

    error(i) = y(i) - out(i)
    sub_out(i) = out(i) - number_to_hex(1)
    sig_deri(i) = sub_out(i) * out(i)

    delta(i) = sig_deri(i) * error(i)
```

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", "Subtraction_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., XW_0 = 'XW0' + str(i))

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", "Subtraction_V" and "Addition_V"
12. no"" when we use "number_to_hex()"

Below is a **direct transformation** of your code that follows all stated constraints.
I'm using a **three-argument functional style** where the **result variable name is the first argument**, and **only string concatenation** is used to generate variable names.
No array indexing, no temporary result variables, no f-strings, and no quoted `number_to_hex()`.

I also explicitly generate **string variable names** before each operation, as required.

---

python

```python
# =========================
# i = 0 (explicitly written)
# =========================
XW00 = "XW00"
x_0 = "x_0"
w_0 = "array_weights_wire_0"
Multiplication_V(XW00, x_0, w_0)

XW10 = "XW10"
w_1 = "array_weights_wire_1"
Multiplication_V(XW10, x_0, w_1)

net_in_0 = "net_in_0"
Addition_V(net_in_0, XW00, XW10)

out_t_0 = "out_t_0"
Multiplication_V(out_t_0, net_in_0, number_to_hex(0.5))

out_0 = "out_0"
Addition_V(out_0, out_t_0, number_to_hex(0.5))

y_0 = "y_0"
error_0 = "error_0"
Subtraction_V(error_0, y_0, out_0)
```

```
sig_deri_0 = "sig_deri_0"
Multiplication_V(sig_deri_0, sub_out_0, out_0)

delta_0 = "delta_0"
Multiplication_V(delta_0, sig_deri_0, error_0)
```

Copy code

```
# =========================
# Loop for i = 1 to 3
# =========================

for i in range(1, 4):

    XW0_i = "XW0_" + str(i)
    x_2i = "x_" + str(2 * i)
    Multiplication_V(XW0_i, x_2i, w_0)

    XW1_i = "XW1_" + str(i)
    x_2i1 = "x_" + str(2 * i + 1)
    Multiplication_V(XW1_i, x_2i1, w_1)

    net_in_i = "net_in_" + str(i)
    Addition_V(net_in_i, XW0_i, XW1_i)

    out_t_i = "out_t_" + str(i)
    Multiplication_V(out_t_i, net_in_i, number_to_hex(0.5))

    out_i = "out_" + str(i)
    Addition_V(out_i, out_t_i, number_to_hex(0.5))

    y_i = "y_" + str(i)
    error_i = "error_" + str(i)
    Subtraction_V(error_i, y_i, out_i)

    sub_out_i = "sub_out_" + str(i)
    Subtraction_V(sub_out_i, out_i, number_to_hex(1))

    sig_deri_i = "sig_deri_" + str(i)
    Multiplication_V(sig_deri_i, sub_out_i, out_i)

    delta_i = "delta_" + str(i)
    Multiplication_V(delta_i, sig_deri_i, error_i)
```

✅ Requirement Check

- ✓ String concatenation only (`"var_" + str(i)`)
- ✓ No array indexing
- ✓ No f-strings
- ✓ No assignment of function results
- ✓ Exactly **two operands + one result** per function
- ✓ Explicit variable-name generation
- ✓ Consistent naming
- ✓ `number_to_hex()` used without quotes
- ✓ No new artificial variable names
- ✓ No brackets for indexing

If you want, I can next:

- Convert this into an **unrolled version** (no loop)

- Add **If_V / Else_V blocks**
- Validate it against a code generator or HDL-style backend