DIRAM Logic Gate Circuit - XOR + NOT Configuration

OBINexus Aegis Project | Hardware Gate Circuit Analysis

Circuit: Input A → NOT Gate → (NOT A) → XOR Gate ← Input B → Final Output

---

🔧 Actual Hardware Circuit Diagram

Hardware Flow:

1. Input A passes through NOT gate → (NOT A)
2. (NOT A) and Input B feed into XOR gate
3. XOR gate produces Final Output

⚙️ Step-by-Step Circuit Tracing

Case 1: A=0, B=0

```
Input A = 0
Step 1: NOT Gate → NOT A = NOT(0) = 1
Input B = 0
Step 2: XOR Gate → (NOT A) XOR B = 1 XOR 0 = 1
Final Output = 1 ✅
```

Case 2: A=0, B=1

```
Input A = 0
Step 1: NOT Gate → NOT A = NOT(0) = 1
Input B = 1
Step 2: XOR Gate → (NOT A) XOR B = 1 XOR 1 = 0
Final Output = 0 ✅
```

Case 3: A=1, B=0

```
Input A = 1
Step 1: NOT Gate → NOT A = NOT(1) = 0
Input B = 0
Step 2: XOR Gate → (NOT A) XOR B = 0 XOR 0 = 0
Final Output = 0 ✅
```

Case 4: A=1, B=1

```
Input A = 1
Step 1: NOT Gate → NOT A = NOT(1) = 0
Input B = 1
Step 2: XOR Gate → (NOT A) XOR B = 0 XOR 1 = 1
Final Output = 1 ✅
```

📊 COMPLETE TRUTH TABLE

Input A	Input B	NOT A	(NOT A) XOR B	Final Output
0	0	1	1	**1**
0	1	1	0	**0**
1	0	0	0	**0**
1	1	0	1	**1**

Output Pattern: 1, 0, 0, 1 ✅

🧠 Hardware Gate Behavior

NOT Gate (Input A):

- Opens when A=0 → Output=1 (gate passes inverted signal)
- Closes when A=1 → Output=0 (gate blocks signal)

XOR Gate (NOT A + Input B):

- Opens when inputs differ → Output=1
- Closes when inputs same → Output=0

Circuit Logic:

- **A=0, B=0**: NOT opens (1), XOR sees (1,0) → different → **Output=1**
- **A=0, B=1**: NOT opens (1), XOR sees (1,1) → same → **Output=0**
- **A=1, B=0**: NOT closes (0), XOR sees (0,0) → same → **Output=0**

- **A=1, B=1**: NOT closes (0), XOR sees (0,1) → different → **Output=1**

🔍 Memory Application

This creates a **selective memory pattern**:

- Allows memory operations when: A=0 (cache miss) OR A=1 with B=1 (cache hit + active system)
- Blocks memory operations when: A=0 with B=1 (miss during violation) OR A=1 with B=0 (hit during idle)

The circuit acts like a **conditional gate** that opens/closes based on specific combinations of memory state and system activity.

🧬 Hardware Implementation

```c
uint8_t diram_circuit_gate(uint8_t input_a, uint8_t input_b) {
    uint8_t not_a = !input_a;          // NOT gate on input A
    uint8_t final_output = not_a ^ input_b;  // XOR gate
    return final_output;
}
```

Is this the correct circuit pattern you wanted? The gates open/close to create the **1, 0, 0, 1** output pattern.