

BlockGateXNOR.java

```
package net.minecraft.src;

// Created by MoareAI
// XNOR Gate [F=n(A⊗B⊗C)]

import java.util.Random;

public class BlockGateXNOR extends Block
{
    //Class parameters
    protected BlockGateXNOR(int i, boolean OutputF)
    {
        super(i, 255, Material.circuits);
        field_OutputF = OutputF;
        setBlockBounds(0.0F, 0.0F, 0.0F, 1.0F, 0.125F, 1.0F);
    }

    //Use render types
    public boolean renderAsNormalBlock()
    {
        return false;
    }

    //Block type (preset)
    public int getRenderType()
    {
        return 14;
    }

    public boolean canPlaceBlockAt(World world, int i, int j, int k)
    {
        if(!world.isBlockOpaqueCube(i, j - 1, k))
        {
            return false;
        } else
        {
            return super.canPlaceBlockAt(world, i, j, k);
        }
    }

    public boolean canBlockStay(World world, int i, int j, int k)
    {
        if(!world.isBlockOpaqueCube(i, j - 1, k))
        {
            return false;
        } else
        {
            return super.canBlockStay(world, i, j, k);
        }
    }

    //Updates on/off
    public void updateTick(World world, int i, int j, int k, Random random)
    {
        int l = world.getBlockMetadata(i, j, k);
        boolean InputA = func_InputA(world, i, j, k, l);
        boolean InputB = func_InputB(world, i, j, k, l);
        boolean InputC = func_InputC(world, i, j, k, l);
        if(InputA ^ InputB ^ InputC)
        {
            world.setBlockAndMetadataWithNotify(i, j, k,
mod_LogicalGates.GateXNOROff.blockID, l);
        } else
        {

```

BlockGateXNOR.java

```

        world.setBlockAndMetadataWithNotify(i, j, k,
mod_LogicalGates.GateXNOROn.blockID, 1);
    }
}

//Update block
public void onNeighborBlockChange(World world, int i, int j, int k, int l)
{
    if(!canBlockStay(world, i, j, k))
    {
        dropBlockAsItem(world, i, j, k, world.getBlockMetadata(i, j, k));
        world.setBlockWithNotify(i, j, k, 0);
        return;
    }
    int i1 = world.getBlockMetadata(i, j, k);
    boolean InputA = func_InputA(world, i, j, k, i1);
    boolean InputB = func_InputB(world, i, j, k, i1);
    boolean InputC = func_InputC(world, i, j, k, i1);
    if(!InputA || !InputB || !InputC || (InputA && InputB && InputC))
    {
        world.scheduleBlockUpdate(i, j, k, blockID, 0);
    }
}

//Signal out
public boolean isPoweringTo(IBlockAccess iblockaccess, int i, int j, int k, int l)
{
    if(!field_OutputF)
    {
        return false;
    }
    int i1 = iblockaccess.getBlockMetadata(i, j, k) & 3;
    if((i1 == 0) && l == 3)
    {
        return true;
    }
    if(i1 == 1 && l == 4)
    {
        return true;
    }
    if(i1 == 2 && l == 2)
    {
        return true;
    }
    return i1 == 3 && l == 5;
}

//Signal in
//Left
private boolean func_InputA(World world, int i, int j, int k, int l)
{
    int i1 = l & 3;
    switch(i1)
    {
        case 0: // '\0'
            return world.isBlockIndirectlyProvidingPowerTo(i - 1, j, k, 4);

        case 2: // '\002'
            return world.isBlockIndirectlyProvidingPowerTo(i + 1, j, k, 5);

        case 3: // '\003'
            return world.isBlockIndirectlyProvidingPowerTo(i, j, k + 1, 3);
    }
}

```

```

        case 1: // '\001'
            return world.isBlockIndirectlyProvidingPowerTo(i, j, k - 1, 2);
        }
        return false;
    }

    //Back
    private boolean func_InputB(World world, int i, int j, int k, int l)
    {
        int i1 = l & 3;
        switch(i1)
        {
            case 0: // '\0'
                return world.isBlockIndirectlyProvidingPowerTo(i, j, k + 1, 3);

            case 2: // '\002'
                return world.isBlockIndirectlyProvidingPowerTo(i, j, k - 1, 2);

            case 3: // '\003'
                return world.isBlockIndirectlyProvidingPowerTo(i + 1, j, k, 5);

            case 1: // '\001'
                return world.isBlockIndirectlyProvidingPowerTo(i - 1, j, k, 4);
        }
        return false;
    }

    //Right
    private boolean func_InputC(World world, int i, int j, int k, int l)
    {
        int i1 = l & 3;
        switch(i1)
        {
            case 0: // '\0'
                return world.isBlockIndirectlyProvidingPowerTo(i + 1, j, k, 5);

            case 2: // '\002'
                return world.isBlockIndirectlyProvidingPowerTo(i - 1, j, k, 4);

            case 3: // '\003'
                return world.isBlockIndirectlyProvidingPowerTo(i, j, k - 1, 2);

            case 1: // '\001'
                return world.isBlockIndirectlyProvidingPowerTo(i, j, k + 1, 3);
        }
        return false;
    }

    public void onBlockPlacedBy(World world, int i, int j, int k, EntityLiving
entityliving)
    {
        int l = ((MathHelper.floor_double(((double)((entityliving.rotationYaw * 4F) /
360F) + 0.5D) & 3) + 2) % 4;
        world.setBlockMetadataWithNotify(i, j, k, l);
        boolean InputB = func_InputB(world, i, j, k, l);
        if(InputB)
        {
            world.scheduleBlockUpdate(i, j, k, blockID, 1);
        }
    }

    public void onBlockAdded(World world, int i, int j, int k)
    {
        world.notifyBlocksOfNeighborChange(i + 1, j, k, blockID);
    }

```

BlockGateXNOR.java

```

world.notifyBlocksOfNeighborChange(i - 1, j, k, blockID);
world.notifyBlocksOfNeighborChange(i, j, k + 1, blockID);
world.notifyBlocksOfNeighborChange(i, j, k - 1, blockID);
world.notifyBlocksOfNeighborChange(i, j - 1, k, blockID);
world.notifyBlocksOfNeighborChange(i, j + 1, k, blockID);
}

public boolean isOpaqueCube()
{
    return false;
}

//Dropped
public int idDropped(int i, Random random)
{
    return mod_LogicalGates.GateXNOR.shiftedIndex;
}

//Torch particles
public void randomDisplayTick(World world, int i, int j, int k, Random random)
{
    if(!field_OutputF)
    {
        return;
    }
    int l = world.getBlockMetadata(i, j, k);
    double d = (double) 0.0D;
    double d1 = (double)((float)j + 0.2F) + (double)(random.nextFloat() - 0.5F) *
0.200000000000000001D;
    double d2 = (double) 0.0D;
    double d3 = 0.0D;
    double d4 = 0.0D;
    if(random.nextInt(2) == 0)
    {
        int i1 = l;
        switch(i1)
        {
            case 0: // '\0'
                d3 = ((float)k + 0.0F) + (double)(random.nextFloat() - 0.5F) *
0.200000000000000001D;
                d4 = ((float)k + 0.8F) + (double)(random.nextFloat() - 0.5F) *
0.200000000000000001D;
                break;

            case 1: // '\001'
                d = ((float)i + 0.8F) + (double)(random.nextFloat() - 0.5F) *
0.200000000000000001D;
                d2 = ((float)k + 0.5F) + (double)(random.nextFloat() - 0.5F) *
0.200000000000000001D;
                break;

            case 2: // '\002'
                d3 = ((float)k + 0.5F) + (double)(random.nextFloat() - 0.5F) *
0.200000000000000001D;
                d4 = (double)((float)k + 0.5F) + (double)(random.nextFloat() - 0.5F) *
0.200000000000000001D;
                break;

            case 3: // '\003'
                d = ((float)i + 0.2F) + (double)(random.nextFloat() - 0.5F) *
0.200000000000000001D;
                d2 = ((float)k + 0.5F) + (double)(random.nextFloat() - 0.5F) *
0.200000000000000001D;
                break;
        }
    }
}

```

BlockGateXNOR.java

```
    }  
    }  
    world.spawnParticle("reddust", d + d3, d1, d2 + d4, 0.0D, 0.0D, 0.0D);  
}  
  
private final boolean field_OutputF;  
}
```