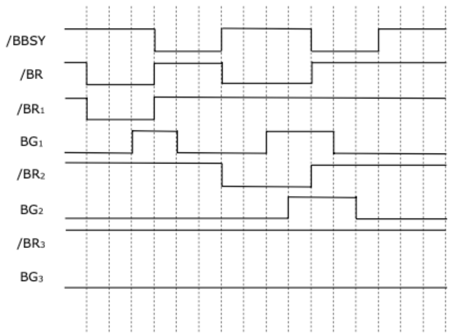
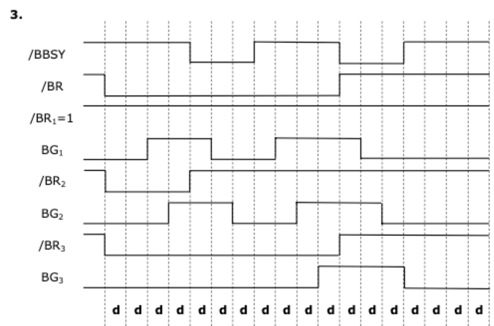
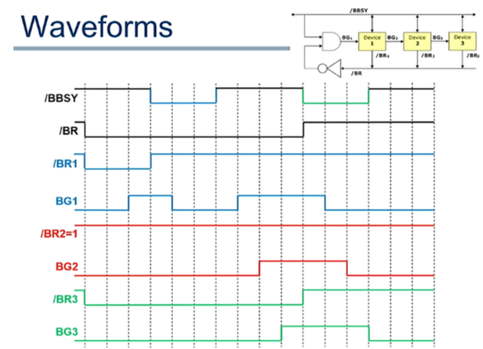
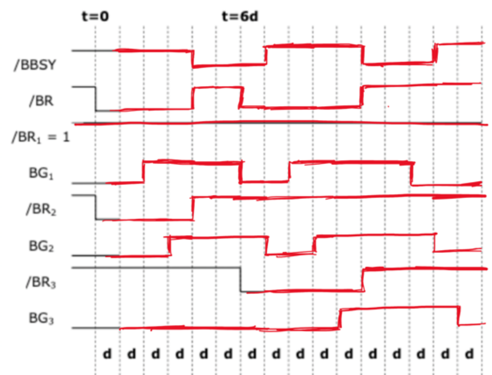
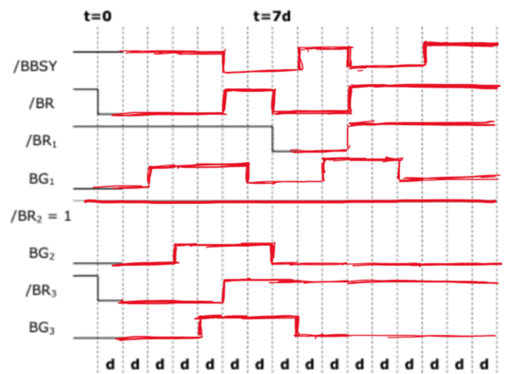
Daisy Waveforms





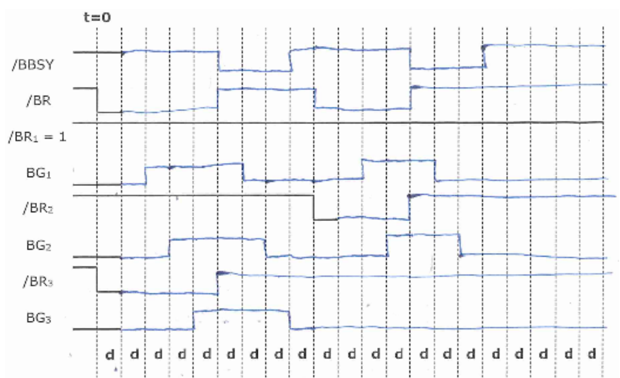


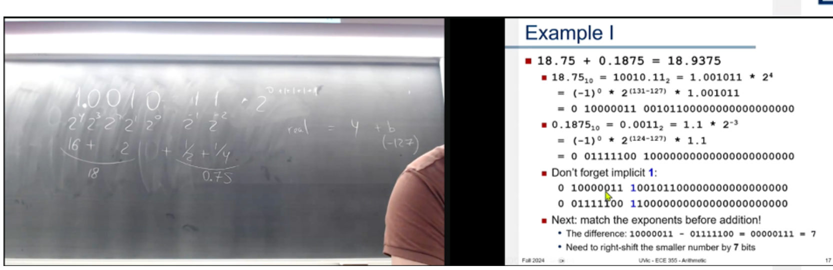


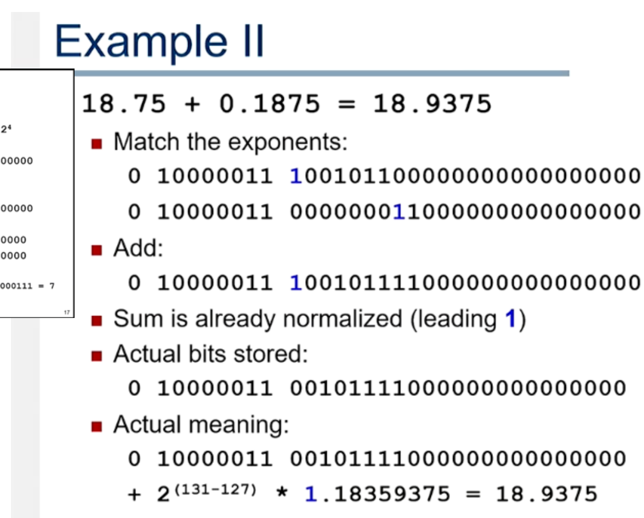


For br2=1:

1. AND gate sees /BR and /BBSY during first time interval and takes one time delay to assert BG1
2. /BR1 takes one time delay after BG1 to stop asserting
3. /BBSY also takes one time delay after BG1 to assert
4. /BR remains asserted because /BR3 is still active
5. BG1 goes low one time delay after /BBSY stops asserting (one delay through AND gate)
6. Since there's still a request, BG1 goes high one time delay after /BBSY has stopped asserting
7. Device 1 doesn't want the signal, so BG2 goes high after one delay
8. Device 2 doesn't want the signal, so BG3 goes high after one delay
9. /BR3 takes one time delay after BG3 to stop asserting. Since /BR3 is directly connected to /BR and no other devices are requesting, /BR stops being asserted immediately.
10. /BBSY also takes one time delay after BG3 to assert
11. /BBSY is active, so after one delay BG1 stops being asserted
12. BG1 isn't active, so after one delay BG2 stops being asserted
13. BG2 isn't active, so after one delay BG3 stops being asserted







Show 32-bit IEEE-754 number 1 11111111 000000000000
in the decimal format.

The binary representation `1 11111111 00000000000000000000000` in the IEEE-754 32-bit floating-point format can be broken down as follows:

Structure of IEEE-754 32-bit:

1. **Sign bit** (1 bit): The first bit indicates the sign.
   - `1` means negative, `0` means positive.
2. **Exponent** (8 bits): The next 8 bits represent the exponent, stored in a "bias-127" format.
   - Bias = 127, so the actual exponent is calculated as:

$$\text{Exponent}_{\text{actual}} = \text{Exponent}_{\text{binary}} - 127$$

3. **Mantissa (Fraction)** (23 bits): The final 23 bits represent the fractional part (normalized as $1.\text{Mantissa}$).

Breakdown of `1 11111111 00000000000000000000000`:

1. **Sign bit**: `1` → Negative number.
2. **Exponent**: `11111111` → 255 in decimal.
3. **Mantissa**: `00000000000000000000000`.

Special Case: Exponent = 255

When the exponent is `255` in IEEE-754:

- If the mantissa is all zeros, the value is either **positive infinity** ($+\infty$) or **negative infinity** ($-\infty$), depending on the sign bit.
- If the mantissa is non-zero, the value is **NaN** (Not a Number).

Result:

- Since the sign bit is `1` and the mantissa is zero, the value is **negative infinity** ($-\infty$).

Show 32-bit IEEE-754 number 0 00000000
11000000000000000000000
in the decimal format.

The binary representation `0 00000000 11000000000000000000000` in the IEEE-754 32-bit floating-point format can be broken down as follows:

Structure of IEEE-754 32-bit:

1. **Sign bit** (1 bit): The first bit indicates the sign.
   - `0` means positive, `1` means negative.
2. **Exponent** (8 bits): The next 8 bits represent the exponent, stored in a "bias-127" format.
   - Bias = 127, so the actual exponent is calculated as:

$$\text{Exponent}_{\text{actual}} = \text{Exponent}_{\text{binary}} - 127$$

3. **Mantissa (Fraction)** (23 bits): The final 23 bits represent the fractional part (normalized or denormalized based on the exponent).

Breakdown of `0 00000000 11000000000000000000000`:

1. **Sign bit**: `0` → Positive number.
2. **Exponent**: `00000000` → Special case (denormalized number or zero).
3. **Mantissa**: `11000000000000000000000`.

Special Case: Exponent = 0

- When the exponent is `0` (all bits are zero), the number is **denormalized**.
- For denormalized numbers:
  - The exponent is treated as $1 - \text{Bias}$, i.e., $1 - 127 = -126$.
  - The number is represented as:
    $$\text{Value} = (-1)^{\text{Sign}} \times 2^{\text{Exponent}} \times 0.\text{Mantissa}$$
    The leading `1` in the mantissa is not assumed for denormalized numbers.

Calculate the Decimal Value:

1. **Sign**: $(-1)^0 = 1$ (positive).
2. **Exponent**: -126.
3. **Mantissa**:
   - Convert the mantissa `11000000000000000000000` to decimal:
     $$1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + \ldots = 0.75$$
   - Since this is a denormalized number, the full value is:
     $$\text{Value} = 1 \times 2^{-126} \times 0.75 = 0.75 \times 2^{-126}.$$

Final Result:

The value is approximately:

$$0.75 \times 2^{-126} \approx 8.410 \times 10^{-39}.$$

(c) Show decimal number −0.625 in the 32-bit IEEE-754 representation.

To represent -0.625 in the 32-bit IEEE-754 floating-point format, we break the process into the following steps:

Step 1: Analyze the Decimal Number

The given number is -0.625.

- **Sign bit (S):** Since the number is negative, the sign bit is 1.
- **Magnitude:** The absolute value is 0.625.

Step 2: Convert to Binary Representation

1. **Whole part:** The whole part of 0.625 is 0. Its binary representation is 0.
2. **Fractional part:** Convert 0.625 to binary by repeatedly multiplying the fractional part by 2 and taking the integer part of the result:
   - $0.625 \times 2 = 1.25$ → Take 1, fractional part 0.25.
   - $0.25 \times 2 = 0.5$ → Take 0, fractional part 0.5.
   - $0.5 \times 2 = 1.0$ → Take 1, fractional part 0.0.

   Binary representation of 0.625 is 0.101.

Thus, -0.625 in binary (ignoring the sign) is 0.101.

Step 3: Normalize the Binary Representation

IEEE-754 format uses normalized numbers, where the binary representation is in the form:

$$1.\text{Mantissa} \times 2^{\text{Exponent}}.$$

For 0.101, normalize it to 1.01×2^{-1}:

- **Mantissa:** 01 (drop the leading $1.$, which is implicit).
- **Exponent:** -1.

Step 4: Encode in IEEE-754 Format

1. **Sign bit (S):** 1 (negative number).
2. **Exponent (E):** Add the bias (127) to the actual exponent (-1):
   $$E = -1 + 127 = 126.$$
   The 8-bit binary representation of 126 is 01111110.
3. **Mantissa (M):** The fractional part 01 is padded to 23 bits:
   01000000000000000000000.

Final Representation

Combine the sign bit, exponent, and mantissa:

$$\text{IEEE-754 (32-bit): } 1\ 01111110\ 01000000000000000000000.$$

Verification

The decoded value from $1\ 01111110\ 01000000000000000000000$ is:

$$(-1)^1 \times 2^{-1} \times 1.01 = -0.625.$$

The representation is correct.

(d) Given two 32-bit IEEE-754 numbers X and Y below, calculate (in
the binary
format) Z = X−Y. Then, convert your IEEE-754 result Z to the decimal
format:
X = 0 01111011 10000000000000000000000,
Y = 1 01111110 11010000000000000000000.

Solution (d):
X = 0 01111011 11000000000000000000000
= 0 01111110 00011000000000000000000
−Y = 0 01111110 11101000000000000000000
X +(−Y) = 0 01111110 100000000000000000000000 (overflow)
= 0 01111111 100000000000000000000000 (normalized)
= (−1)0*1.0*2(127-127) -> 1.0

< 2/2 >

Let's break the problem down step by step to understand how $Z = X - Y$ is calculated and verify the process.

Given Numbers:

1. $X = 0\ 01111011\ 10000000000000000000000$:
   - Sign bit (S): 0 → Positive number.
   - Exponent (E): 01111011 → Decimal 123.
     $$\text{Actual Exponent} = 123 - 127 = -4.$$
   - Mantissa (M): 1.1 (implicit leading 1).
     $$X = (+1) \times 2^{-4} \times 1.1 = 1.5 \times 2^{-4} = 0.09375.$$
2. $Y = 1\ 01111110\ 11010000000000000000000$:
   - Sign bit (S): 1 → Negative number.
   - Exponent (E): 01111110 → Decimal 126.
     $$\text{Actual Exponent} = 126 - 127 = -1.$$
   - Mantissa (M): 1.1101 (implicit leading 1).
     $$Y = (-1) \times 2^{-1} \times 1.1101 = -1.8125.$$
3. **Operation:**
   $$Z = X - Y = 0.09375 - (-1.8125) = 0.09375 + 1.8125 = 1.90625.$$

Verify Intermediate Steps in Binary

1. **Convert X and $-Y$ to Binary for Alignment:**
   - **Normalize X:** Adjust X to the same exponent as Y (i.e., -1):
     - Shift mantissa of X left by 3 bits ($E = -4$ → $E = -1$):
       $$X = 0\ 01111110\ 00011000000000000000000.$$
   - **Invert Sign of Y:** $-Y = 0\ 01111110\ 11101000000000000000000.$
2. **Add X and $-Y$:**
   - Exponents are aligned, so directly add mantissas:
     $\text{Mantissa: } 00011000000000000000000 + 11101000000000000000000 = 100000000000000000000000 \text{ (overflow occurs)}.$
3. **Handle Overflow:**
   - Overflow shifts the result one place to the right, incrementing the exponent by 1:
     $$\text{Result: } 0\ 01111111\ 10000000000000000000000.$$

Convert Final Result Z to Decimal

1. **Representation of Z:**
   $$Z = 0\ 01111111\ 10000000000000000000000.$$
   - Sign bit (S): 0 → Positive number.
   - Exponent (E): 01111111 → Decimal 127.
     $$\text{Actual Exponent} = 127 - 127 = 0.$$
   - Mantissa (M): 1.1 (implicit leading 1).
     $$Z = (+1) \times 2^0 \times 1.1 = 1.5.$$

Final Answer:

$$Z = 1.5 \text{ in decimal format.}$$