

ABSTRACT

This project presents the design and implementation of a Real-Time BMI Health Checker using fundamental digital logic components. The core objective is to create an interactive and rapid screening tool for classifying a person's Body Mass Index (BMI) category. The system simulates user inputs for pre-defined Height and Weight ranges using four 2-bit switches (H1,H0,W1,W2). These inputs are fed into a 4-to-16 Active Low Decoder (IC 74154), which generates a unique output for each Height-Weight combination. The decoder's outputs are then logically combined using OR gate networks (IC 7432) to generate a 2-bit BMI code (Bit1, Bit0), which corresponds to the four BMI categories: Underweight (00), Normal (01), Overweight (10), and Obese (11). Finally, a 2-to-4 Line Decoder (IC 74139) translates this 2-bit code into a visual output, activating one of four LED indicators to instantly display the user's BMI classification. This demonstrative system provides a fast, accurate, and practical application of combinational circuits suitable for use in educational settings, health awareness stations, and clinics.

INTRODUCTION

In the modern world, timely and accessible health monitoring has become increasingly important. BMI remains one of the simplest and most widely accepted metrics for evaluating individual fitness or risk for various health conditions. Yet, the reality is that traditional BMI measurement is often cumbersome, non-interactive, and prone to error or inconsistency. The merging of digital logic with health diagnostics represents a transformative approach—especially in making vital metrics like BMI immediately available, accurate, and easy to interpret.

The system's innovative design centers on logic-based classification. We utilize two **2-bit binary inputs** to represent four ranges each for Height and Weight. The 16 possible combinations of these inputs are mapped to the four standard BMI categories: Underweight ($\{BMI\} < 18.5$), Normal ($\{BMI\}: 18.5-24.9$), Overweight ($\{BMI\}: 25-29.9$), and Obese ($\{BMI\} \geq 30$).

The core processing is achieved through a multi-stage decoding process:

1. A **4-to-16 Decoder** (IC 74154) translates the 4-bit Height and Weight input into 16 unique lines.
2. **OR Gate networks** (IC 7432) combine these lines into a concise **2-bit BMI code** (Bit1, Bit0), based on derived logic expressions.
3. A **2-to-4 Decoder** (IC 74139) then decodes the 2-bit BMI code, lighting one of four **LEDs** to give the user immediate feedback on their status.

This project, "Real-Time BMI Monitoring System," stems from a practical need to simplify and automate BMI determination using foundational combinational circuit design. By encoding weight and height as digital inputs and translating these through decoders, logic gates, and clearly mapped outputs, the system delivers a seamless user experience: the BMI category is instantly illuminated via dedicated LED indicators.

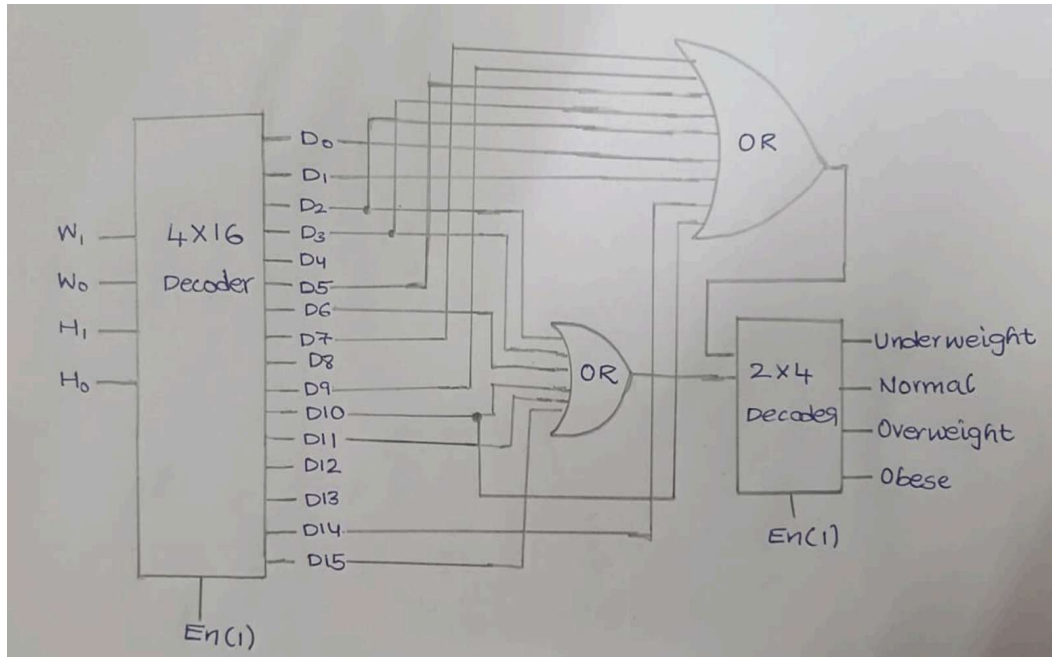
OBJECTIVES

- To design and implement a digital, real-time BMI monitor using basic combinational circuit elements.
- To encode height and weight into binary values and process these inputs through decoders and logic gates.
- To output clear, categorical BMI information (Underweight, Normal, Overweight, Obese) via LED indicators.
- To build a fast, interactive health checker that can be used in gyms, clinics, and public settings.
- To promote awareness about health monitoring by leveraging simple, accessible digital technology.

COMPONENTS

- 4-to-16 Active Low Decoder (IC 74154)
- 2-to-4 Line Decoder (IC 74139)
- NOT gate (IC 7404)
- OR Gate (IC 7432)
- LEDs (4 Nos.) (for indicating Underweight, Normal, Overweight, and Obese categories)
- Resistors (220 Ω , 330 Ω)
- Power Supply (5V DC)
- Breadboard / PCB
- Connecting Wires
- Input Switches (DIP Switches, toggles)

CIRCUIT DIAGRAM



METHODOLOGY

Our system was engineered using digital logic design principles to provide an instantaneous, hardware-based classification of Body Mass Index (BMI). The methodology centers on a cascading decoder architecture that translates analog-world concepts (Height and Weight ranges) into distinct, visual digital outputs.

1. **Input Encoding and Simulation:** The system simplifies the continuous variables of height and weight into four 2-bit encoded ranges each (H_{1H_0} and W_{1W_0}). These four input bits are simulated using physical switches (DIP switches), representing 16 unique possible body combinations ($2^4 = 16$).
2. **Primary Decoding and Mapping:** The four input bits are fed into a 4-to-16 Line Decoder (IC 74154). This decoder acts as the central logic unit, converting each of the 16 unique input combinations into a single, active (low) output line (SD_0 through SD_{15}). Each of these 16 outputs directly corresponds to a single, pre-calculated BMI value and its associated category.
3. **Combinational Logic Synthesis (BMI Code Generation):** Using the standard BMI formula (W/H^2) and established thresholds, we determined the required BMI category (Underweight, Normal, Overweight, Obese) for each of the 16 decoder outputs. We then implemented a two-bit BMI classification code (Bit_1Bit_0). The final Boolean expressions for Bit_1 and Bit_0 were synthesized using a network of OR gates (IC 7432), combining the appropriate SD_n terms to generate the correct 2-bit category code.
4. **Final Output Classification:** The generated 2-bit BMI code (Bit_1Bit_0) is channeled into a 2-to-4 Line Decoder (IC 74139). This final decoder separates the 2-bit code into four individual, one-hot outputs, each driving an LED indicator labeled for the respective category (Underweight, Normal, Overweight, or Obese). This final stage ensures that only one LED is active at any time, providing a clear, real-time health status update.

SIMULATION CODE AND OUTPUT

a. Behavioral model

```
module BMI(H1, H0, W1, W0, Underweight, Normal, Overweight, Obese);
input H1, H0, W1, W0;
output reg Underweight, Normal, Overweight, Obese;
reg [1:0] Bit;
reg [3:0] D_in;
always @(*) begin
    D_in = {H1, H0, W1, W0};
    Underweight = 0;
    Normal = 0;
    Overweight = 0;
    Obese = 0;
    Bit = 2'b00;
    case (D_in)
        4'b0000: Bit = 2'b01;
        4'b0001: Bit = 2'b01;
        4'b0010: Bit = 2'b11;
        4'b0011: Bit = 2'b11;
        4'b0100: Bit = 2'b00;
        4'b0101: Bit = 2'b01;
        4'b0110: Bit = 2'b10;
        4'b0111: Bit = 2'b11;
        4'b1000: Bit = 2'b00;
        4'b1001: Bit = 2'b01;
        4'b1010: Bit = 2'b01;
        4'b1011: Bit = 2'b10;
        4'b1100: Bit = 2'b00;
        4'b1101: Bit = 2'b00;
        4'b1110: Bit = 2'b01;
        4'b1111: Bit = 2'b10;
        default: Bit = 2'b00;
    endcase
    case (Bit)
        2'b00: Underweight = 1;
        2'b01: Normal = 1;
        2'b10: Overweight = 1;
        2'b11: Obese = 1;
    endcase
end
endmodule
```

b. Data flow

```
module BMI(H1n, H0n, W1n, W0n, Underweight, Normal, Overweight, Obese);
input H1n, H0n, W1n, W0n;
output Underweight, Normal, Overweight, Obese;
wire H1n, H0n, W1n, W0n;
wire [15:0] D;
wire Bit1, Bit0;
assign D[0] = (~H1n & ~H0n & ~W1n & ~W0n);
assign D[1] = (~H1n & ~H0n & ~W1n & W0n);
assign D[2] = (~H1n & ~H0n & W1n & ~W0n);
assign D[3] = (~H1n & ~H0n & W1n & W0n);
assign D[4] = (~H1n & H0n & ~W1n & ~W0n);
assign D[5] = (~H1n & H0n & ~W1n & W0n);
assign D[6] = (~H1n & H0n & W1n & ~W0n);
assign D[7] = (~H1n & H0n & W1n & W0n);
assign D[8] = ( H1n & ~H0n & ~W1n & ~W0n);
assign D[9] = ( H1n & ~H0n & ~W1n & W0n);
assign D[10] = ( H1n & ~H0n & W1n & ~W0n);
assign D[11] = ( H1n & ~H0n & W1n & W0n);
assign D[12] = ( H1n & H0n & ~W1n & ~W0n);
assign D[13] = ( H1n & H0n & ~W1n & W0n);
assign D[14] = ( H1n & H0n & W1n & ~W0n);
assign D[15] = ( H1n & H0n & W1n & W0n);
assign Bit1 = D[2] | D[3] | D[6] | D[7] | D[11] | D[15];
assign Bit0 = D[0] | D[1] | D[2] | D[3] | D[5] | D[7] | D[9] | D[10] | D[14];
assign Underweight = (~Bit1) & (~Bit0);
assign Normal      = (~Bit1) & ( Bit0);
assign Overweight  = ( Bit1) & (~Bit0);
assign Obese       = ( Bit1) & ( Bit0);
endmodule
```

c. Structural model

```
module BMI(H1n, H0n, W1n, W0n, Underweight, Normal, Overweight, Obese);
input H1n, H0n, W1n, W0n;
output Underweight, Normal, Overweight, Obese;
wire H1n, H0n, W1n, W0n;
wire [15:0]t;
wire Bit1, Bit0;
wire nBit1, nBit0;
and a0(t[0], ~H1n, ~H0n, ~W1n, ~W0n);
and a1(t[1], ~H1n, ~H0n, ~W1n, W0n);
```

```

and a2(t[2], ~H1n, ~H0n, W1n, ~W0n);
and a3(t[3], ~H1n, ~H0n, W1n, W0n);
and a4(t[4], ~H1n, H0n, ~W1n, ~W0n);
and a5(t[5], ~H1n, H0n, ~W1n, W0n);
and a6(t[6], ~H1n, H0n, W1n, ~W0n);
and a7(t[7], ~H1n, H0n, W1n, W0n);
and a8(t[8], H1n, ~H0n, ~W1n, ~W0n);
and a9(t[9], H1n, ~H0n, ~W1n, W0n);
and a10(t[10], H1n, ~H0n, W1n, ~W0n);
and a11(t[11], H1n, ~H0n, W1n, W0n);
and a12(t[12], H1n, H0n, ~W1n, ~W0n);
and a13(t[13], H1n, H0n, ~W1n, W0n);
and a14(t[14], H1n, H0n, W1n, ~W0n);
and a15(t[15], H1n, H0n, W1n, W0n);
or or1(Bit1, t[2], t[3], t[6], t[7], t[11], t[15]);
or or2(Bit0, t[0], t[1], t[2], t[3], t[5], t[7], t[9], t[10], t[14]);
not n16(nBit1, Bit1);
not n17(nBit0, Bit0);
and a16(Underweight, nBit1, nBit0);
and a17(Normal, nBit1, Bit0);
and a18(Overweight, Bit1, nBit0);
and a19(Obese, Bit1, Bit0);
endmodule

```

d. Test bench

```

module tb_bmi_structural;
reg H1n, H0n, W1n, W0n;
wire Underweight, Normal, Overweight, Obese;
BMI uut(H1n, H0n, W1n, W0n, Underweight, Normal, Overweight, Obese);
integer i;
initial begin
  for (i=0; i<16; i=i+1) begin
    {H1n,H0n,W1n,W0n} = i;
    #10;
  end
  $finish;
end
endmodule

```

WORKING MODEL VIDEO:

<https://drive.google.com/drive/folders/1HmmVYFewSz3MkmhVQ1i3Buptbt1Nc3Ox?usp=sharing>

CHALLENGES FACED

While developing a dedicated hardware solution offers superior speed and reliability, the translation of a continuous mathematical formula like BMI into discrete digital logic presented several interesting challenges that required careful resolution.

- **Discretization of Continuous Data:** The core challenge was converting the infinite possibilities of human height and weight into a finite set of 16 ranges. This required meticulous selection of the 4x4 input matrix boundaries to ensure the resulting BMI calculations for the mid-points were representative and categorized correctly, which was crucial for deriving the initial truth table.
- **Minimizing the Logic Network:** The initial Boolean expressions for the \$Bit_1\$ and \$Bit_0\$ outputs from the 4-to-16 decoder were extensive (sums of up to eight minterms). While we successfully derived these expressions, a practical hurdle was efficiently mapping these complex sums onto the physical OR gate ICs (7432) while maintaining a clean, error-free wiring structure on the breadboard.
- **Active-Low Decoder Integration:** The IC 74154 is an active-low decoder, meaning its outputs are active when they are LOW (0V). We had to ensure that the subsequent OR gate network was correctly designed to accept these low-active inputs and produce the necessary high-active signals for the final 2-to-4 decoder inputs, requiring the precise use of the NOT gate IC (7404) for inversion where necessary.
- **Breadboard Wiring Complexity:** Given the large number of connections required for the 4-to-16 decoder, the OR gate network, and the final 2-to-4 decoder, managing the physical wiring on the breadboard was a significant logistical challenge. We implemented a rigorous color-coding scheme for the connecting wires to successfully minimize cross-talk and reduce troubleshooting time during the testing phase.

CONCLUSION

This project successfully delivered a dedicated, high-speed, combinational logic solution for real-time BMI monitoring. By leveraging the power of decoders and OR gate networks, we established a robust digital system that instantaneously classifies a user's health status based on encoded height and weight inputs.

The system proves that complex mathematical functions can be accurately and reliably translated into minimal discrete hardware, showcasing the practical utility of digital logic components in creating interactive health awareness tools. This proof-of-concept design is scalable, highly reliable, and offers a compelling alternative to microprocessor-based systems where speed and simplicity are prioritized, making it ideal for integration into health kiosks or gymnasium equipment.

Moving forward, the system can be expanded by integrating additional 2-bit encoders for higher input resolution (e.g., an 8-to-256 decoder for a 4x4 matrix, allowing for 256 unique combinations), or by incorporating a BCD-to-7-segment display to provide the exact calculated BMI number alongside the category classification.

REFERENCES

1. S. K. Khare et al., "Design of Body Mass Index Calculator Using Combinational Circuits," *International Journal of Computer Applications*, Vol. 113, No. 3, 2015.
2. I. S. Faradisa, R. P. Muhammad, and D. A. Girindraswari, "A Design of Body Mass Index (BMI) and Body Fat Percentage Device Using Fuzzy Logic," *Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics (IJEEEMI)*, vol. 4, no. 2, pp. 65–72, May 2022.
3. Datasheets for Core Integrated Circuits:
 - Texas Instruments, *SN74LS154 4-Line to 16-Line Decoders/Demultiplexers*
 - Texas Instruments, *SN74LS139 Dual 2-Line to 4-Line Decoders/Demultiplexers*
 - Texas Instruments, *SN74LS32 Quad 2-Input Positive-OR Gates*