Chat : https://chatgpt.com/share/cf37777d-9dff-42e8-8e2f-e10eb87933e7

I want to refactor this verilog module "`include "preemphasis_filter.v"

```
`include "framing_windowing.v"
`include "goertzel_dft.v"
`include "mel_filterbank.v"
`include "logarithm_comp.v"
`include "dct_comp.v"
`include "mfcc_accelerator.v"
`include "conv2d.v"
`include "maxpool2d.v"
`include "fully_connected.v"
`include "softmax.v"

module cnn_kws_accel #(
    parameter NUM_KEYWORDS = 10,
    parameter MFCC_FEATURES = 40,
    parameter ACTIV_BITS = 8,
    parameter FC1_INPUT_SIZE = (MFCC_FEATURES/2)*(CONV2_NUM_FILTERS),
    parameter FC1_OUTPUT_SIZE = 64,
    parameter FC2_INPUT_SIZE = 64,
    parameter FC2_OUTPUT_SIZE = NUM_KEYWORDS,
    parameter CONV1_KERNEL_SIZE = 3,
    parameter CONV1_NUM_FILTERS = 8,
    parameter CONV2_KERNEL_SIZE = 3,
    parameter CONV2_NUM_FILTERS = 8
)(
    input wire clk,
    input wire rst_n,
    input wire [15:0] audio_in,
    input wire audio_valid,
    output reg [NUM_KEYWORDS-1:0] kws_result,
    output reg kws_valid,
    input wire [7:0] frame_size,
    input wire [7:0] frame_overlap,
    input wire [7:0] num_mfcc_coeffs,
    input wire [7:0] goertzel_coefs,
    output wire goertzel_coefs_start,
    output wire goertzel_coefs_valid,
    output wire goertzel_coefs_done,
    input wire [CONV1_NUM_FILTERS*CONV1_KERNEL_SIZE*CONV1_KERNEL_SIZE*ACTIV_BITS-1:0] conv1_weights,
    input wire [CONV1_NUM_FILTERS*ACTIV_BITS-1:0] conv1_biases,
    input wire conv1_load_weights,
    input wire conv1_load_biases,
    input wire [CONV2_NUM_FILTERS*CONV1_NUM_FILTERS*CONV2_KERNEL_SIZE*CONV2_KERNEL_SIZE*ACTIV_BITS-1:0] conv2_weights,
    input wire [CONV2_NUM_FILTERS*ACTIV_BITS-1:0] conv2_biases,
    input wire conv2_load_weights,
    input wire conv2_load_biases,
    input wire [FC1_OUTPUT_SIZE*FC1_INPUT_SIZE*ACTIV_BITS-1:0] fc1_weights,
    input wire [FC1_OUTPUT_SIZE*ACTIV_BITS-1:0] fc1_biases,
    input wire fc1_load_weights,
    input wire fc1_load_biases,
    input wire [FC2_OUTPUT_SIZE*FC2_INPUT_SIZE*ACTIV_BITS-1:0] fc2_weights,
    input wire [FC2_OUTPUT_SIZE*ACTIV_BITS-1:0] fc2_biases,
    input wire fc2_load_weights,
    input wire fc2_load_biases
);


    // MFCC module signals
```

```
wire [MFCC_FEATURES*ACTIV_BITS-1:0] mfcc_out;
wire mfcc_valid;

// CNN-KWS layers
wire [MFCC_FEATURES*CONV1_NUM_FILTERS*ACTIV_BITS-1:0] conv1_out;
wire conv1_valid;
wire [MFCC_FEATURES*CONV2_NUM_FILTERS*ACTIV_BITS-1:0] conv2_out;
wire conv2_valid;
wire [(MFCC_FEATURES/2)*CONV2_NUM_FILTERS*ACTIV_BITS-1:0] maxpool_out;
wire maxpool_valid;
wire [FC1_OUTPUT_SIZE*ACTIV_BITS-1:0] fc1_out;
wire fc1_valid;
wire [FC2_OUTPUT_SIZE*ACTIV_BITS-1:0] fc2_out;
wire fc2_valid;
wire [FC2_OUTPUT_SIZE*ACTIV_BITS-1:0] softmax_out;
wire softmax_valid;

// MFCC module instantiation
mfcc_accelerator #(
  .MFCC_FEATURES(MFCC_FEATURES),
  .ACTIV_BITS(ACTIV_BITS)
) mfcc (
  .clk(clk),
  .rst_n(rst_n),
  .audio_in(audio_in),
  .audio_valid(audio_valid),
  .mfcc_out(mfcc_out),
  .mfcc_valid(mfcc_valid),
  .frame_size(frame_size),
  .frame_overlap(frame_overlap),
  .num_mfcc_coeffs(num_mfcc_coeffs),
        .goertzel_coefs(goertzel_coefs),
        .goertzel_coefs_start(goertzel_coefs_start),
        .goertzel_coefs_valid(goertzel_coefs_valid),
        .goertzel_coefs_done(goertzel_coefs_done)
    );


// Convolutional layer 1
conv2d #(
    .INPUT_WIDTH(MFCC_FEATURES),
    .INPUT_HEIGHT(1),
    .INPUT_CHANNELS(1),
    .KERNEL_SIZE(CONV1_KERNEL_SIZE),
    .NUM_FILTERS(CONV1_NUM_FILTERS),
    .PADDING(1),
    .ACTIV_BITS(ACTIV_BITS)
) conv1 (
    .clk(clk),
    .rst_n(rst_n),
    .data_in(mfcc_out),
    .data_valid(mfcc_valid),
    .data_out(conv1_out),
    .data_out_valid(conv1_valid),
    .weights_in(conv1_weights),
    .biases_in(conv1_biases),
    .load_weights(conv1_load_weights),
    .load_biases(conv1_load_biases)
);

// Convolutional layer 2
conv2d #(
    .INPUT_WIDTH(MFCC_FEATURES),
    .INPUT_HEIGHT(1),
    .INPUT_CHANNELS(CONV1_NUM_FILTERS),
    .KERNEL_SIZE(CONV2_KERNEL_SIZE),
    .NUM_FILTERS(CONV2_NUM_FILTERS),
    .PADDING(1),
    .ACTIV_BITS(ACTIV_BITS)
) conv2 (
    .clk(clk),
    .rst_n(rst_n),
    .data_in(conv1_out),
    .data_valid(conv1_valid),
    .data_out(conv2_out),
    .data_out_valid(conv2_valid),
    .weights_in(conv2_weights),
    .biases_in(conv2_biases),
    .load_weights(conv2_load_weights),
    .load_biases(conv2_load_biases)
);

            maxpool2d #(
                .INPUT_WIDTH(MFCC_FEATURES),
                .INPUT_HEIGHT(1),
                .INPUT_CHANNELS(CONV2_NUM_FILTERS),
                .KERNEL_SIZE(2),
                .STRIDE(2),
                .ACTIV_BITS(ACTIV_BITS)
            ) maxpool (
                .clk(clk),
                .rst_n(rst_n),
                .data_in(conv2_out),
    .data_valid(conv2_valid),
    .data_out(maxpool_out),
    .data_out_valid(maxpool_valid)
);

fully_connected #(
    .INPUT_SIZE(FC1_INPUT_SIZE),
    .OUTPUT_SIZE(FC1_OUTPUT_SIZE),
    .ACTIV_BITS(ACTIV_BITS)
) fc1 (
    .clk(clk),
    .rst_n(rst_n),
    .data_in(maxpool_out),
    .data_valid(maxpool_valid),
    .data_out(fc1_out),
    .data_out_valid(fc1_valid),
    .weights_in(fc1_weights[FC1_OUTPUT_SIZE*FC1_INPUT_SIZE*ACTIV_BITS-1:0]),
    .biases_in(fc1_biases),
    .load_weights(fc1_load_weights),
    .load_biases(fc1_load_biases)
);

// Fully connected layer 2 (output layer)
fully_connected #(
    .INPUT_SIZE(FC2_INPUT_SIZE),
    .OUTPUT_SIZE(FC2_OUTPUT_SIZE),
    .ACTIV_BITS(ACTIV_BITS)
) fc2 (
    .clk(clk),
    .rst_n(rst_n),
    .data_in(fc1_out),
            .data_valid(fc1_valid),
            .data_out(fc2_out),
            .data_out_valid(fc2_valid),
            .weights_in(fc2_weights),
            .biases_in(fc2_biases),
            .load_weights(fc2_load_weights),
            .load_biases(fc2_load_biases)
        );

        // Softmax layer
        softmax #(
            .INPUT_SIZE(NUM_KEYWORDS),
            .ACTIV_BITS(ACTIV_BITS)
        ) softmax (
            .clk(clk),
            .rst_n(rst_n),
            .data_in(fc2_out),
            .data_valid(fc2_valid),
            .data_out(softmax_out),
            .data_out_valid(softmax_valid)
        );

// Output assignment
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        kws_result <= 'b0;
        kws_valid <= 1'b0;
    end else begin
        kws_result <= softmax_out[NUM_KEYWORDS-1:0];
        kws_valid <= softmax_valid;
    end
    end
```

endmodule" so as not to instantiate the mfcc_accelerator module but instead the mfcc_accel module that has this code "module mfcc_accel (

```
    input wire clk,
    input wire rst,
    input wire [15:0] audio_sample,
    input wire sample_valid,
    output wire [15:0] mfcc_feature,
    output wire mfcc_valid
);

// Instantiate the Hanning window real module
wire [15:0] hanning_real_out;
wire hanning_real_valid;

hanning_window_real hanning_real (
    .clk(clk),
    .rst(rst),
    .sample_in(audio_sample),
    .sample_valid(sample_valid),
    .sample_out(hanning_real_out),
    .sample_out_valid(hanning_real_valid)
);

// Instantiate the Hanning window imaginary module
wire [15:0] hanning_imag_out;
wire hanning_imag_valid;

hanning_window_imag hanning_imag (
    .clk(clk),
    .rst(rst),
    .sample_in(audio_sample),
    .sample_valid(sample_valid),
    .sample_out(hanning_imag_out),
    .sample_out_valid(hanning_imag_valid)
);

// Instantiate the periodogram module
wire [31:0] periodogram_out;
wire periodogram_valid;

periodogram_squared periodogram (
    .clk(clk),
    .rst(rst),
    .sample_in_real(hanning_real_out),
    .sample_in_imag(hanning_imag_out),
    .sample_valid(hanning_real_valid), // Use the valid signal from the real Hanning window
    .periodogram_out(periodogram_out),
    .periodogram_valid(periodogram_valid)
);

// Instantiate the pow_module
wire [31:0] pow_out;
wire pow_valid;

pow_module pow (
    .clk(clk),
    .rst(rst),
    .data_in(periodogram_out),
    .data_valid(periodogram_valid),
    .data_out(pow_out),
    .data_out_valid(pow_valid)
);

// Instantiate the mel_filterbank module
wire [31:0] mel_out;
wire mel_valid;

mel_filterbank mel (
    .clk(clk),
    .rst(rst),
    .data_in(pow_out),
    .data_valid(pow_valid),
    .mel_out(mel_out),
    .mel_valid(mel_valid)
);

// Instantiate the log_module
wire [15:0] log_out;
wire log_valid;

log_module log (
    .clk(clk),
    .rst(rst),
    .data_in(mel_out),
    .data_valid(mel_valid),
    .log_out(log_out),
    .log_valid(log_valid)
);

// Instantiate the dct_module
dct_module dct (
    .clk(clk),
    .rst(rst),
    .data_in(log_out),
    .data_valid(log_valid),
    .dct_out(mfcc_feature),
    .dct_valid(mfcc_valid)
);

endmodule"
```

ChatGPT

To refactor the cnn_kws_accel Verilog module to use the mfcc_accel module instead of mfcc_accelerator, you need to update the instantiation of the MFCC module in cnn_kws_accel accordingly. Here's how the refactored cnn_kws_accel module would look:

verilog

Copy code

```
`include "preemphasis_filter.v"
`include "framing_windowing.v"
`include "goertzel_dft.v"
`include "mel_filterbank.v"
`include "logarithm_comp.v"
`include "dct_comp.v"
`include "conv2d.v"
`include "maxpool2d.v"
`include "fully_connected.v"
`include "softmax.v"

module cnn_kws_accel #(
    parameter NUM_KEYWORDS = 10,
    parameter MFCC_FEATURES = 40,
    parameter ACTIV_BITS = 8,
    parameter FC1_INPUT_SIZE = (MFCC_FEATURES/2)*(CONV2_NUM_FILTERS),
    parameter FC1_OUTPUT_SIZE = 64,
    parameter FC2_INPUT_SIZE = 64,
    parameter FC2_OUTPUT_SIZE = NUM_KEYWORDS,
    parameter CONV1_KERNEL_SIZE = 3,
    parameter CONV1_NUM_FILTERS = 8,
    parameter CONV2_KERNEL_SIZE = 3,
    parameter CONV2_NUM_FILTERS = 8
)(
    input wire clk,
    input wire rst_n,
    input wire [15:0] audio_in,
    input wire audio_valid,
    output reg [NUM_KEYWORDS-1:0] kws_result,
    output reg kws_valid,
    input wire [7:0] frame_size,
    input wire [7:0] frame_overlap,
    input wire [7:0] num_mfcc_coeffs,
    input wire [7:0] goertzel_coefs,
    output wire goertzel_coefs_start,
    output wire goertzel_coefs_valid,
    output wire goertzel_coefs_done,
    input wire [CONV1_NUM_FILTERS*CONV1_KERNEL_SIZE*CONV1_KERNEL_SIZE*ACTIV_BITS-1:0] conv1_weights,
    input wire [CONV1_NUM_FILTERS*ACTIV_BITS-1:0] conv1_biases,
    input wire conv1_load_weights,
    input wire conv1_load_biases,
    input wire [CONV2_NUM_FILTERS*CONV1_NUM_FILTERS*CONV2_KERNEL_SIZE*CONV2_KERNEL_SIZE*ACTIV_BITS-1:0] conv2_weights,
    input wire [CONV2_NUM_FILTERS*ACTIV_BITS-1:0] conv2_biases,
    input wire conv2_load_weights,
    input wire conv2_load_biases,
    input wire [FC1_OUTPUT_SIZE*FC1_INPUT_SIZE*ACTIV_BITS-1:0] fc1_weights,
    input wire [FC1_OUTPUT_SIZE*ACTIV_BITS-1:0] fc1_biases,
    input wire fc1_load_weights,
    input wire fc1_load_biases,
    input wire [FC2_OUTPUT_SIZE*FC2_INPUT_SIZE*ACTIV_BITS-1:0] fc2_weights,
    input wire [FC2_OUTPUT_SIZE*ACTIV_BITS-1:0] fc2_biases,
    input wire fc2_load_weights,
    input wire fc2_load_biases
);

    // MFCC module signals
    wire [15:0] mfcc_out;
    wire mfcc_valid;

    // CNN-KWS layers
    wire [MFCC_FEATURES*CONV1_NUM_FILTERS*ACTIV_BITS-1:0] conv1_out;
    wire conv1_valid;
    wire [MFCC_FEATURES*CONV2_NUM_FILTERS*ACTIV_BITS-1:0] conv2_out;
    wire conv2_valid;
    wire [(MFCC_FEATURES/2)*CONV2_NUM_FILTERS*ACTIV_BITS-1:0] maxpool_out;
    wire maxpool_valid;
    wire [FC1_OUTPUT_SIZE*ACTIV_BITS-1:0] fc1_out;
    wire fc1_valid;
    wire [FC2_OUTPUT_SIZE*ACTIV_BITS-1:0] fc2_out;
    wire fc2_valid;
    wire [FC2_OUTPUT_SIZE*ACTIV_BITS-1:0] softmax_out;
    wire softmax_valid;

    // MFCC module instantiation
    mfcc_accel mfcc (
        .clk(clk),
        .rst(rst_n), // Adjusted to rst_n
        .audio_sample(audio_in), // Adjusted to audio_sample
        .sample_valid(audio_valid), // Adjusted to sample_valid
        .mfcc_feature(mfcc_out), // Adjusted to mfcc_feature
        .mfcc_valid(mfcc_valid) // Adjusted to mfcc_valid
    );

// Convolutional layer 1
conv2d #(
    .INPUT_WIDTH(MFCC_FEATURES),
    .INPUT_HEIGHT(1),
    .INPUT_CHANNELS(1),
    .KERNEL_SIZE(CONV1_KERNEL_SIZE),
    .NUM_FILTERS(CONV1_NUM_FILTERS),
    .PADDING(1),
    .ACTIV_BITS(ACTIV_BITS)
) conv1 (
    .clk(clk),
    .rst_n(rst_n),
    .data_in(mfcc_out),
    .data_valid(mfcc_valid),
    .data_out(conv1_out),
    .data_out_valid(conv1_valid),
    .weights_in(conv1_weights),
    .biases_in(conv1_biases),
    .load_weights(conv1_load_weights),
    .load_biases(conv1_load_biases)
);

// Convolutional layer 2
conv2d #(
    .INPUT_WIDTH(MFCC_FEATURES),
    .INPUT_HEIGHT(1),
    .INPUT_CHANNELS(CONV1_NUM_FILTERS),
    .KERNEL_SIZE(CONV2_KERNEL_SIZE),
    .NUM_FILTERS(CONV2_NUM_FILTERS),
    .PADDING(1),
    .ACTIV_BITS(ACTIV_BITS)
) conv2 (
    .clk(clk),
    .rst_n(rst_n),
    .data_in(conv1_out),
    .data_valid(conv1_valid),
    .data_out(conv2_out),
    .data_out_valid(conv2_valid),
    .weights_in(conv2_weights),
    .biases_in(conv2_biases),
    .load_weights(conv2_load_weights),
    .load_biases(conv2_load_biases)
);

            maxpool2d #(
                .INPUT_WIDTH(MFCC_FEATURES),
                .INPUT_HEIGHT(1),
                .INPUT_CHANNELS(CONV2_NUM_FILTERS),
                .KERNEL_SIZE(2),
                .STRIDE(2),
                .ACTIV_BITS(ACTIV_BITS)
            ) maxpool (
                .clk(clk),
                .rst_n(rst_n),
                .data_in(conv2_out),
    .data_valid(conv2_valid),
    .data_out(maxpool_out),
    .data_out_valid(maxpool_valid)
);

fully_connected #(
    .INPUT_SIZE(FC1_INPUT_SIZE),
    .OUTPUT_SIZE(FC1_OUTPUT_SIZE),
    .ACTIV_BITS(ACTIV_BITS)
) fc1 (
    .clk(clk),
    .rst_n(rst_n),
    .data_in(maxpool_out),
    .data_valid(maxpool_valid),
    .data_out(fc1_out),
    .data_out_valid(fc1_valid),
    .weights_in(fc1_weights[FC1_OUTPUT_SIZE*FC1_INPUT_SIZE*ACTIV_BITS-1:0]),
    .biases_in(fc1_biases),
    .load_weights(fc1_load_weights),
    .load_biases(fc1_load_biases)
);

// Fully connected layer 2 (output layer)
fully_connected #(
    .INPUT_SIZE(FC2_INPUT_SIZE),
    .OUTPUT_SIZE(FC2_OUTPUT_SIZE),
    .ACTIV_BITS(ACTIV_BITS)
) fc2 (
    .clk(clk),
    .rst_n(rst_n),
    .data_in(fc1_out),
            .data_valid(fc1_valid),
            .data_out(fc2_out),
            .data_out_valid(fc2_valid),
            .weights_in(fc2_weights),
            .biases_in(fc2_biases),
            .load_weights(fc2_load_weights),
            .load_biases(fc2_load_biases)
        );

        // Softmax layer
        softmax #(
            .INPUT_SIZE(NUM_KEYWORDS),
            .ACTIV_BITS(ACTIV_BITS)
        ) softmax (
            .clk(clk),
            .rst_n(rst_n),
            .data_in(fc2_out),
            .data_valid(fc2_valid),
            .data_out(softmax_out),
            .data_out_valid(softmax_valid)
        );

// Output assignment
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        kws_result <= 'b0;
        kws_valid <= 1'b0;
    end else begin
        kws_result <= softmax_out[NUM_KEYWORDS-1:0];
        kws_valid <= softmax_valid;
    end
    end


endmodule
```

Changes made:

Removed the parameters specific to mfcc_accelerator.

Adjusted the instantiation of the MFCC module to use mfcc_accel with the corresponding input and output signals.

Adjusted signal names and connections to match those used in mfcc_accel.