ECE 464 / ECE 564
Project
Revision: 1.2

On-line turn-in. Individual assignment. Check out the submission instructions below. We will be checking your submission using Code Comparison tools for plagiarism. If your code is substantially similar to someone else's you will both receive an academic violation for cheating.

Transformers:

Transformers models represent a breakthrough in processing sequential data for large language models and generative artificial intelligence (AI), natural language processing (NLP), machine translation, and sentiment analysis. A. Vaswani et.al propose "Transformer" in their paper "Attention is all you need". The Transformer overcomes drawbacks of Long Short Term Memory (LSTM) based Recurrent Neural Networks (RNNs) and Convolution Neural Networks (CNNs) as they process in parallel rather than processing sequentially to attend to different parts of the input sequence.

The transformers employ unique mechanisms such as positional encoding, embedding, and self-attention to enable the creation of sequence relationships. The building blocks of a Transformer are:

1. *Positional encoding and embedding:*
   - The primary target for the Transformer is NLP, which has unique word embedding maps for each word in the sequence to a word vector of d_{model} size. Word vectors reduce the dimension and improves contextual similarity.
   - To retain the order of the sequence, "positional encoding" the input embeddings is performed.
2. *Self-attention:*
   - Sometimes called intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation if the sequence. Let's look at an example: "The animal didn't cross the street because it was too tired". The word "it" may refer to animal or to the street. Self-attention mechanisms allow it to associate "it" with "animal".
   - An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, and values are trainable vectors.
   - The "Scaled Dot-Product Attention" is computed by:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

3. *Multi-Head-Attention (MHA):*
   - In the transformer, the Attention module repeats its computations multiple times in parallel.
   - MHA allows the model to jointly attend to information from different representation subspaces at different positions.
   - Performs attention function with multiple $head_i$, each with different, learned linear projections parameter matrices W_i^Q, K_i^K, and V_i^V.

$$MultiHead(Q,K,V) = Concat(head_1, head_2, \dots, head_h)W^O$$
$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

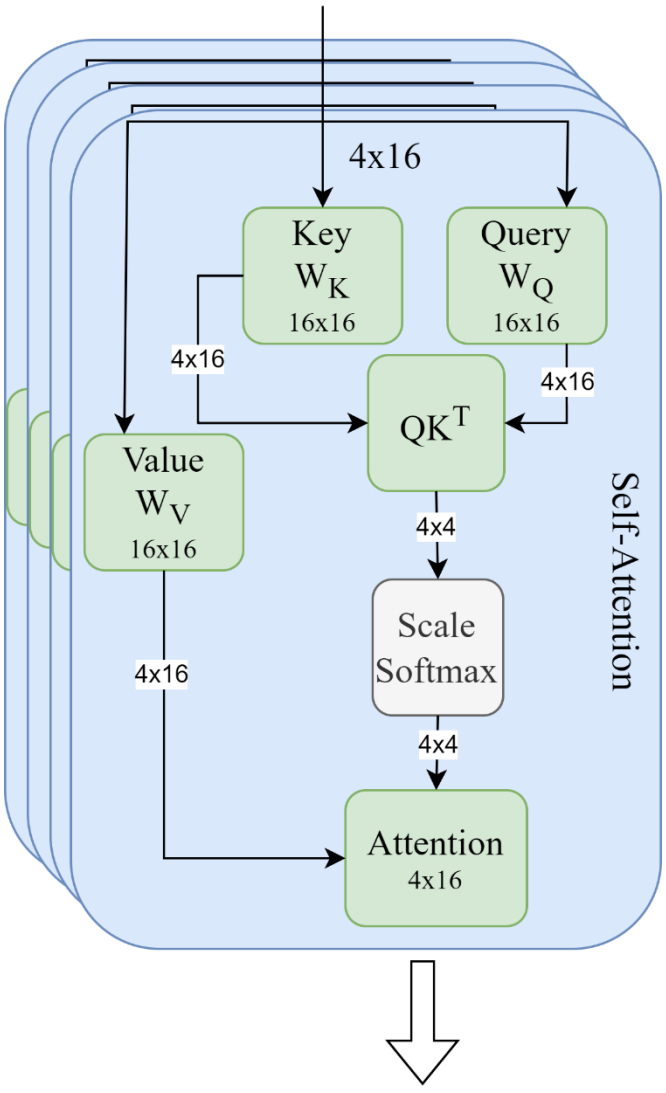


Figure 1: Transformer self-attention query, key, value, score and attention connection and matrix dimensions.

Matrix multiplication:

The matrix multiplication is performed on Matrix I (SRAM input) and Matrix W (SRAM project), and the results will be stored in SRAM Result. The equation below, shows the matrix multiplication performed:

$$\begin{bmatrix} Q_1 & Q_2 & \cdots & Q_{16} \\ Q_{17} & Q_{18} & \cdots & Q_{32} \\ \vdots & \ddots & \ddots & \vdots \\ Q_{49} & Q_{50} & \cdots & Q_{64} \end{bmatrix} = \begin{bmatrix} I_1 & I_2 & \cdots & I_{16} \\ I_{17} & I_{18} & \cdots & I_{32} \\ \vdots & \ddots & \ddots & \vdots \\ I_{49} & I_{50} & \cdots & I_{64} \end{bmatrix} * \begin{bmatrix} wq_1 & wq_{17} & wq_{33} & \cdots & wq_{241} \\ wq_2 & wq_{18} & wq_{34} & \cdots & wq_{242} \\ wq_3 & wq_{19} & wq_{35} & \cdots & wq_{243} \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ wq_{16} & wq_{32} & \cdots & \cdots & wq_{256} \end{bmatrix} \quad \text{(Eq: 1)}$$

As regular matrix multiplication, you will multiply and accumulate each row element of matrix I with column elements of matrix W. The expression below shows the example computation:

$$Q_1 = (I_1 * wq_1) + (I_2 * wq_2) + (I_3 * wq_3) + \cdots + (I_{16} * wq_{16})$$

$$Q_2 = (I_1 * wq_{17}) + (I_2 * wq_{18}) + (I_3 * wq_{19}) + \cdots + (I_4 * wq_{32}) \cdots$$

$$Q_{17} = (I_{17} * wq_1) + (I_{18} * wq_2) + (I_{19} * wq_3) + \cdots + (I_{32} * wq_{16})$$

$$Q_{18} = (I_{17} * wq_{17}) + (I_{18} * wq_{18}) + (I_{19} * wq_{19}) + \cdots + (I_{32} * wq_{32}) \cdots$$

$$Q_{63} = (I_{49} * wq_{241}) + (I_{50} * wq_{242}) + (I_{51} * wq_{243}) + \cdots + (I_{64} * wq_{256})$$

ECE464 project:

The 464 project will realize the QK^T part of the self-attention equation. This involves multiple matrix multiplication operations to obtain the results. The inputs and weight parameters are loaded by the testbench in "sram_input" and "sram_weight". The results are expected to be written by the DUT in "sram_result". Refer to the "SRAM contents mapping" section for the corresponding mapping schemes of different parameters.

1. Calculating the query (Q) and key (K) matrices.
    - Query (Q) is obtained by multiplying input embedding (I) with weight matrices (W^Q).
        i. $Q = I * W^Q$
    - Key (K) is obtained by multiplying input embedding (I) with weight matrices (W^K).
        i. $K = I * W^K$

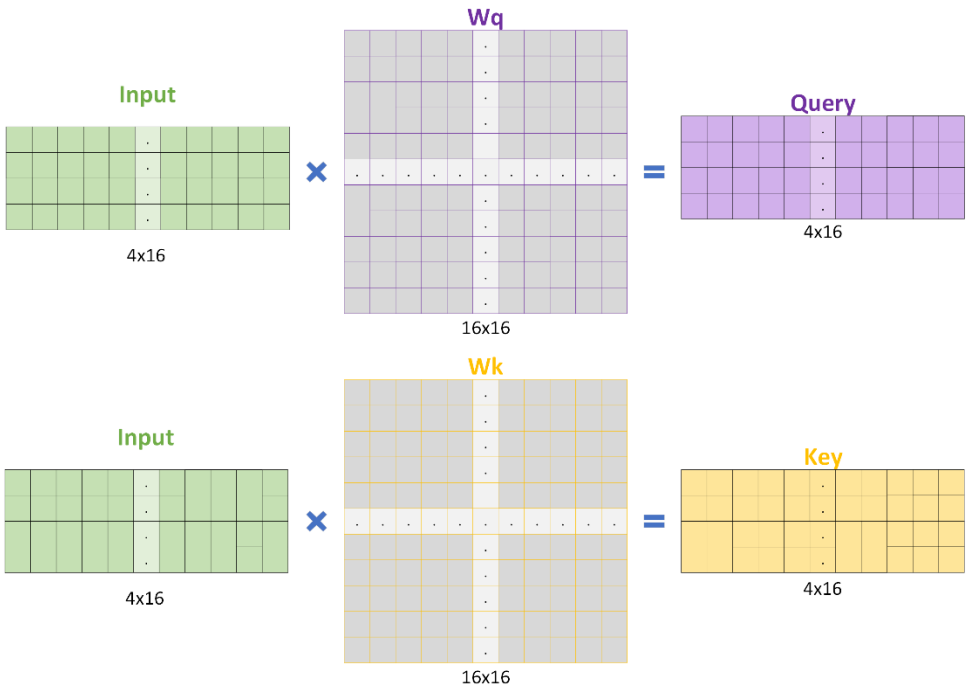
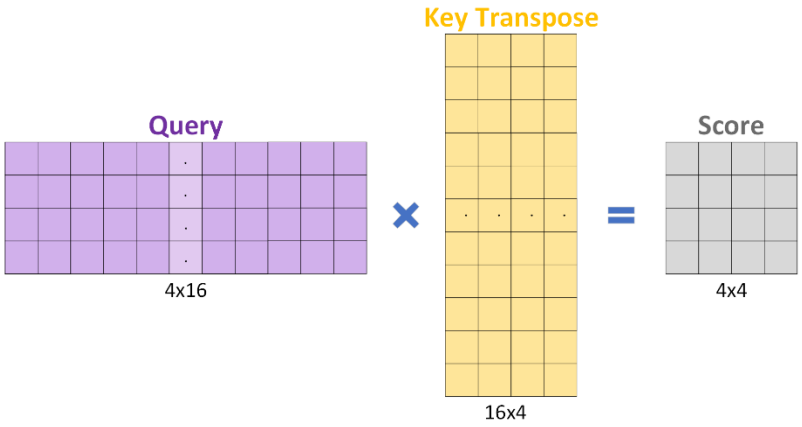


Figure 2: Calculation of Query and Key matrices by multiplying Input embedding with corresponding weight parameters (W^Q, W^K).

2. Compute the score matrix.
    - Transpose the Key matrix from the previous step to obtain K^T.
    - Multiply Query (Q) with Key transpose (K^T).
        i. $Score = QK^T$



ECE564 project:

The 564 project will realize the "Scaled Dot-Product Attention" $[(\frac{QK^T}{\sqrt{d_k}})V]$.

1. Calculating the query (Q), key (K), and value (V) matrices.
    - Query (Q) is obtained by multiplying input embedding (I) with weight matrices (W^Q).
        i. $Q = I * W^Q$
    - Key (K) is obtained by multiplying input embedding (I) with weight matrices (W^K).
        i. $K = I * W^K$
    - Value (V) is obtained by multiplying input embedding (I) with weight matrices (W^V).
        i. $V = I * W^V$
2. Compute the score matrix (S).
    - Transpose the Key matrix from the previous step to obtain K^T.
    - Multiply Query (Q) with Key transpose (K^T).
        i. $S = QK^T$
3. Compute the scaled dot-product attention (Z).
    - Multiply the score (S) with the value (V).
        i. $\mathrm{Z} = \mathrm{S} * \mathrm{V}$

System signals:

- *reset_n* is used to reset the logic into a known state,
- *clk* is used to drive the flip-flop logic in the design.

Control signals:

- *dut_valid:* used as part of a hand shack between the test fixture and the dut. Valid is used to signal that a valid input can be computed from the SRAM.
- *dut_ready:* used to signal that the dut is ready to receive new input from the SRAM.
- Together these two signals tell the test fixture the state of the dut. So, the dut should assert *dut_ready* on reset and wait for the *dut_valid* to be asserted.
- Once *dut_valid* is asserted by the test fixture, the dut should set *dut_ready* to low and can start reading from the SRAM.
- Dut should hold the *dut_ready* low until it has populated the result values in the SRAM. Once the results are stored in the SRAM the *dut_ready* will be asserted high, signaling that the result is valid, and is ready to be read from SRAM. Fig 2 shows the expected behavior.

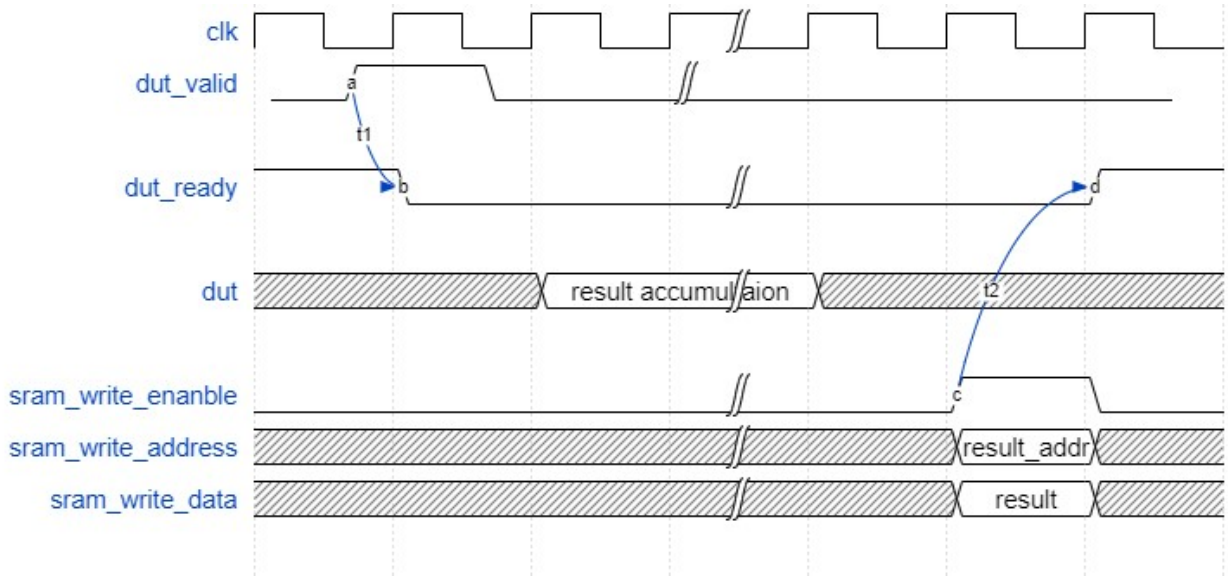


Fig 2. Test fixture and DUT handshake behavior

SRAM contents mapping:

The SRAM holds the 32-bit data in each address. The table below shows the memory mapping of various SRAM addresses. The color schemes corresponding to Equation 1. The input SRAM's contains the input matrix dimension at address 12'h00, while the matrix data from 12'h01 onwards.

SRAM input: Address	SRAM input: Content [31:0]
12'h00	[31:16] – Number of matrix A rows, [15:0] – Number of matrix A columns
12'h01	I_1
12'h02	I_2
.	.
.	.
12'h40	I_{64}

SRAM weight: Address	SRAM weight: Content [31:0]
12'h00	[31:16] – Number of matrix B rows, [15:0] – Number of matrix B columns
12'h01	wq_1
12'h02	wq_2
.	.
.	.
12'h100	wq_{256}
12'h101	wk_1
12'h102	wk_2
.	.
.	.
12'h200	wk_{256}
12'h201	wv_1
12'h202	wv_2
.	.
.	.
12'h300	wv_{256}

SRAM Result: Address	SRAM Result: Content [31:0]
12'h00	Q_1
12'h01	Q_2
.	.
.	.
12'h3F	Q_{64}
12'h40	K_1
12'h41	K_2
.	.
.	.
12'h7F	K_{64}
12'h80	V_1

12’h81	V_2
.	.
.	.
12’hBF	V_{64}
12’hC0	S_1
12’hC1	S_2
.	.
.	.
12’hCF	S_{16}
12’hD0	Z_1
12’hD1	Z_2
.	.
.	.
12’h10F	Z_{64}

SRAM:

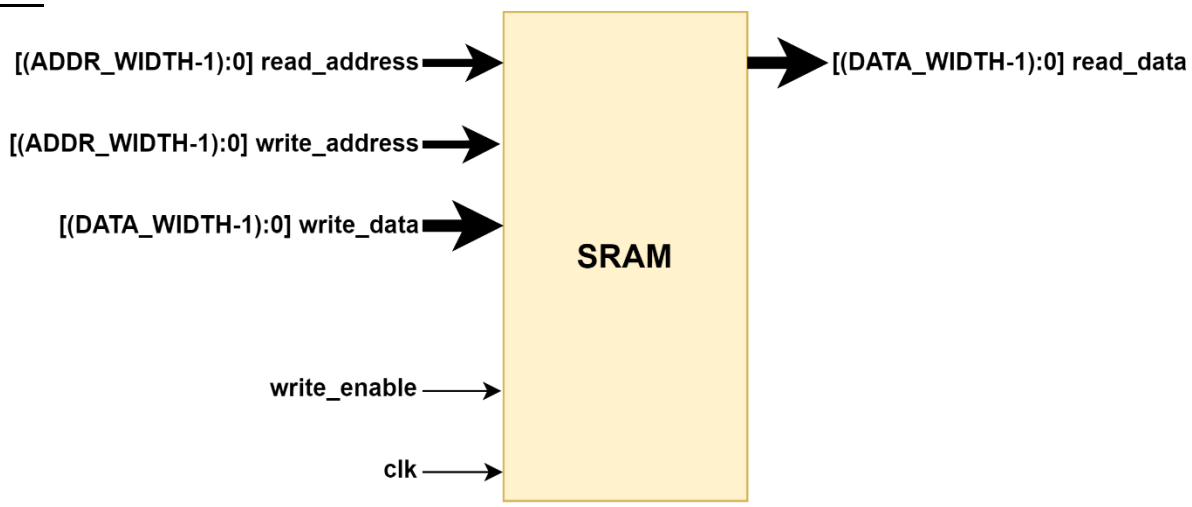


Fig 3. SRAM interface ports

The SRAM is word addressable and has a one cycle delay between address and data. When writing to the SRAM, you would have to set the "write_enable" to high. The SRAM will write the data in the next cycle. "read_write_select" is not used for this implementation.

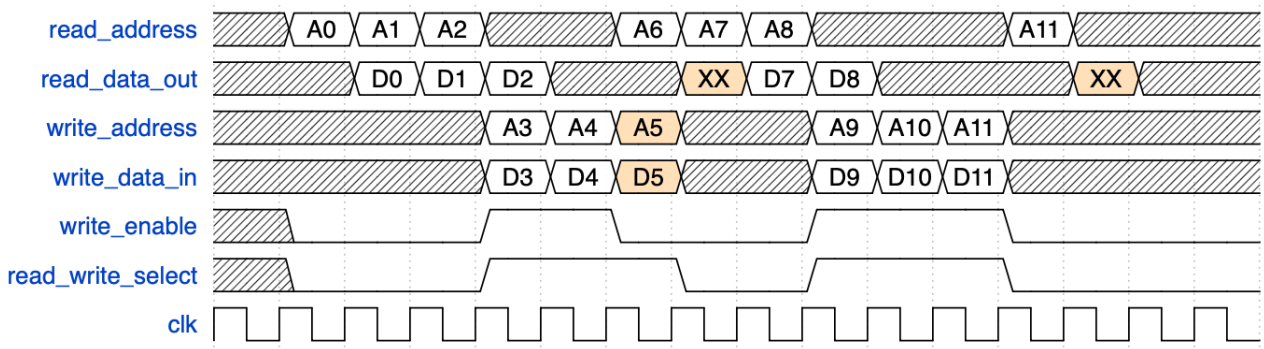


Fig 4. SRAM timing behavior

As shown in the Fig 4, since "write_enable" is set to low when A5 and D5 is on the write bus, D5 will not be written to the SRAM. Also, because "read_write_select" is set to high, the read request for A6 will not be valid.

Note that the SRAM cannot handle consecutive read after write (RAW) to the same address (shown as A11 and D11 in the timing diagram). You would have to either manage the timing of your access or write the data forwarding mechanism yourself. As long as the read and write address are different, the request can be pipelined.

Important Notes:

1. Please read the README and look at the dut.sv/testbench.sv file.
2. Design should not have any major/minor synthesis errors pointed out in the Standard Class Tutorial (Appendix C). This includes but is not limited to latches, wired-OR, combination feedback, etc.

Design, verify, synthesize a module that meets these specifications. **Use at least one coding feature unique to System Verilog.**

Submission Instruction:

- **Project Verilog and synthesis files.** Submitted electronically on the date indicated in the class schedule. Please turn in the following:
    - All Verilog files AS ONE FILE in submission.
    - Zipped modelsim simulation results file showing correct functionality. Logs from '/run/logs/*.log'
    - Synopsys view_command.log file from complete synthesis run
    - **Project Report.** Complete report to be turned in electronically with project files. It must follow the format attached. There is a 10% penalty for not following the format.
- **DUT ONLY submission.** Submit your dut.sv file without any folders or zip files, just the .sv file on its own to the DUT ONLY project submission link on Moodle.

[50 points]