This document demonstrates how to calculate the purity/entanglement values (aka purity of reduced state) for the qubits in a quantum computer.

The first thing that needs to be calculated is the reduced density matrix for each qubit.

The following equation is taken from Wikipedia. It is the reduced density matrix for system A of a larger system of qubits AB.

$$\rho_A \overset{\text{def}}{=} \sum_j^{N_B} (I_A \otimes \langle j|_B)\,(|\Psi\rangle\langle\Psi|)\,(I_A \otimes |j\rangle_B) = \mathrm{Tr}_B\;\rho_T.$$

For this Wikipedia equation:

|Ψ><Ψ| is the density matrix for the whole system AB

these are $<j|_B$:
$<0| = \begin{bmatrix} 1 & 0 \end{bmatrix}$

$<1| = \begin{bmatrix} 0 & 1 \end{bmatrix}$

these are $|j>_B$:
$|0> = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
$|1> = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

I_A (the identity matrix) = $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

The following examples describe how to find the reduced density matrices for each qubit (system A) in a three qubit system in a quantum computer.

The reduced density matrix for the first qubit (of three) in a quantum computer:
ρ_{A1} = eq 1a + eq 1b

eq 1a

$$\left(\begin{bmatrix}1 & 0\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \text{ x } \begin{bmatrix}\text{8x8}\\ \textit{density matrix}\end{bmatrix} \text{ x } \left(\begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$$

eq 1b

$$\left(\begin{bmatrix}0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \text{ x } \begin{bmatrix}\text{8x8}\\ \textit{density matrix}\end{bmatrix} \text{ x } \left(\begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$$

The reduced density matrix for the second qubit (of three) in a quantum computer:
ρ_{A2} = eq 2a + eq 2b

eq 2a

$$\left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \text{ x } \begin{bmatrix}\text{8x8}\\ \textit{density matrix}\end{bmatrix} \text{ x } \left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$$

eq 2b

$$\left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \text{ x } \begin{bmatrix}\text{8x8}\\ \textit{density matrix}\end{bmatrix} \text{ x } \left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$$

The reduced density matrix for the third qubit (of three) in a quantum computer:
ρ_{A3} = eq 3a + eq 3b

eq 3a

$$\left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\end{bmatrix}\right) \text{ x } \begin{bmatrix}\text{8x8}\\ \textit{density matrix}\end{bmatrix} \text{ x } \left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix}\right)$$

eq 3b

$$\left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}0 & 1\end{bmatrix}\right) \text{ x } \begin{bmatrix}\text{8x8}\\ \textit{density matrix}\end{bmatrix} \text{ x } \left(\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix}\right)$$

To calculate the amount of purity/entanglement for each qubit, simply take the trace of the square of each reduced density matrix.

Here is some Python code to calculate the amount of purity/entanglement for each qubit in a quantum computer.

```
import numpy as np

def calculateAndDisplayQubitsEntanglement(stateVector):
    # This function implements the Reduced Density Matrix formula found in the Wikipedia
article:
    # pA = summation(j -> NB)[ (IA kron <j|B) (|psi><psi|) (IA kron |j>B) ]
    #    where "kron" is the Kronecker product (aka "the tensor product")

    numberOfQubits = int(np.log2(len(stateVector)))

    identityMatrix = np.array ([[1,0],[0,1]]) # create the one qubit identity matrix

    qbZeroRow = np.array([1,0]) # create a ZERO one qubit state ROW vector
    qbOneRow = np.array([0,1]) # create a ONE one qubit state ROW vector
    qbZeroCol = np.array([[1],[0]]) # create a ZERO one qubit state COLUMN vector
    qbOneCol = np.array([[0],[1]]) # create a ONE one qubit state COLUMN vector


    # Implementation of (|psi><psi|) (the full non-reduced density matrix):
    #
    transposedConjugateSV = np.conj(stateVector)
    transposedConjugateSV = np.transpose(transposedConjugateSV)
    densityMatrix = np.kron(transposedConjugateSV,stateVector) # get the outer product of the
state vector and its conjugated transpose

    for qubitNumber in range(0,numberOfQubits): # process each qubit for its level of
entanglement in this outer loop
        # Initialize the variables that will hold the qubit-wide Kronecker (tensor) products.
        Ij_Zero_bra = 1
        Ij_One_bra = 1
        Ij_Zero_ket = 1
        Ij_One_ket = 1

        for Ij in range(0,numberOfQubits): # perform the Kronecker products in this inner loop
            # Implementation of (I kron I or <j|B) and (I kron I or |j>B):
             if (qubitNumber == Ij):
```

```
            # It's time to tensor-in the qb/Zero/One/Row/Col vectors with the four
Ij_One/Zero/bra/ket because we are processing the outer loop's current qubit now.
            # All of the other tensor positions will have, or have had, the one-qubit identity
matrix.
            Ij_Zero_bra = np.kron(qbZeroRow,Ij_Zero_bra)
            Ij_One_bra = np.kron(qbOneRow,Ij_One_bra)
            Ij_Zero_ket = np.kron(qbZeroCol,Ij_Zero_ket)
            Ij_One_ket = np.kron(qbOneCol,Ij_One_ket)
        else:
            # Tensor-in a one-qubit identity matrix into the other Ij* positions (i.e., not for the
current outer loop's qubit).
            Ij_Zero_ket = np.kron(identityMatrix,Ij_Zero_ket)
            Ij_One_ket = np.kron(identityMatrix,Ij_One_ket)
            Ij_Zero_bra = np.kron(identityMatrix,Ij_Zero_bra)
            Ij_One_bra = np.kron(identityMatrix,Ij_One_bra)

    rdmZero = np.matmul(Ij_Zero_bra,densityMatrix)
    rdmZero = np.matmul(rdmZero,Ij_Zero_ket)
    rdmOne = np.matmul(Ij_One_bra,densityMatrix)
    rdmOne = np.matmul(rdmOne,Ij_One_ket)

    reducedDensityMatrix = rdmZero + rdmOne

    # take the trace of the square of the reduced density matrix, then round it
    rdmSquared = np.dot(reducedDensityMatrix,reducedDensityMatrix)

    entanglementValue =
round(np.trace(np.dot(reducedDensityMatrix,reducedDensityMatrix)),3) # round to three
decimal places

    print("The purity-entanglement value for qubit",qubitNumber," =
",entanglementValue.real," ((where 1.0 = no entanglement : 0.5 = maximum entanglement))")
```