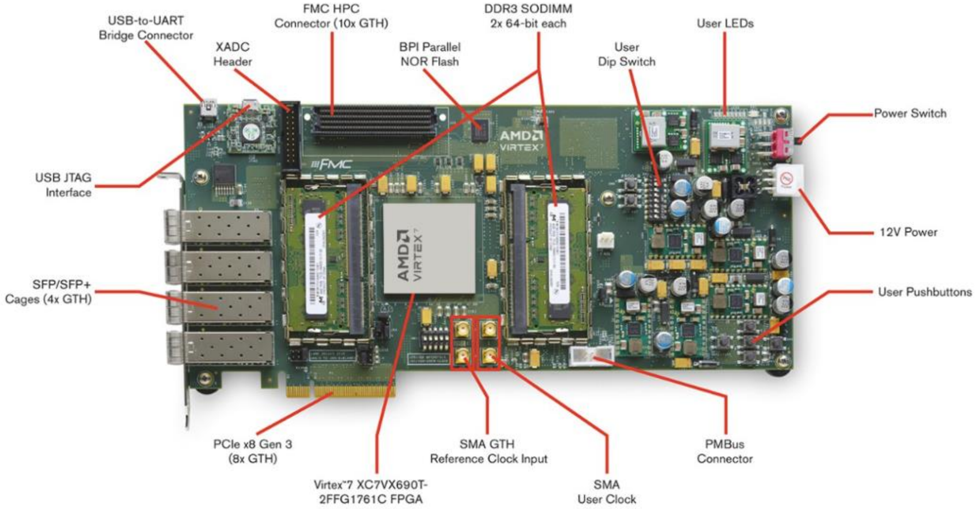
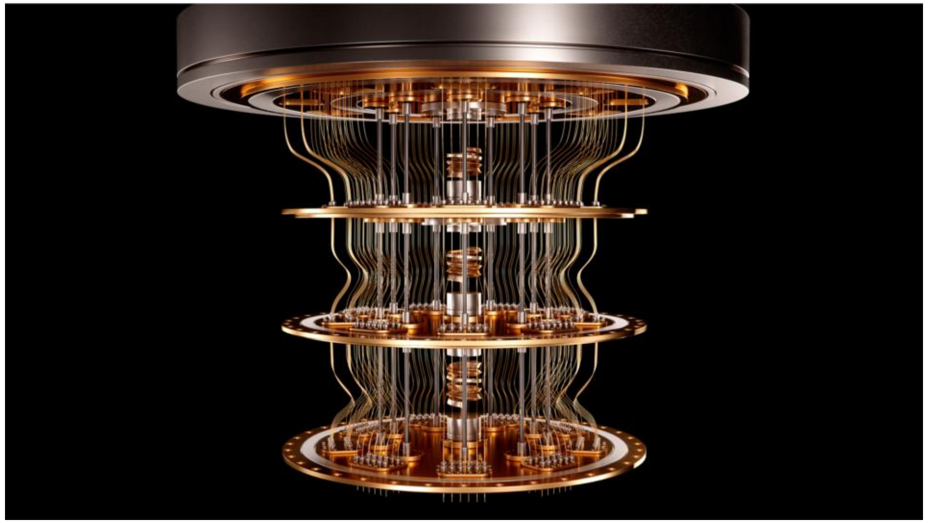
Project subject

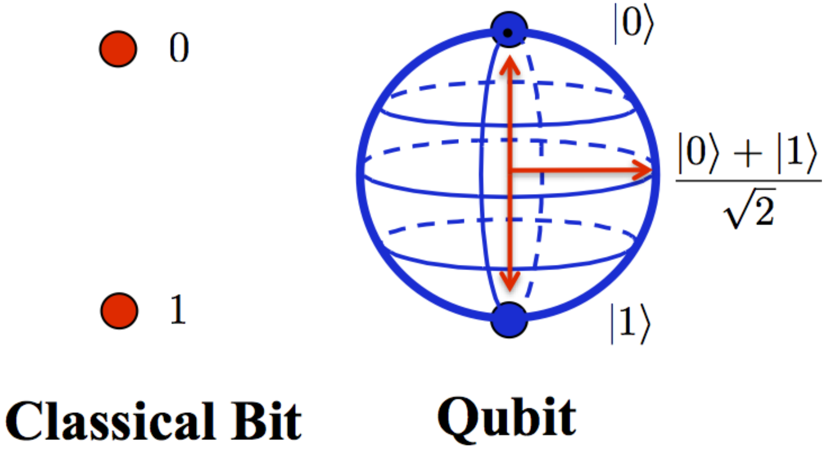
- Simulation and implementation of quantum transform fourier algorithm using FPGA device and classic PC while comparing the two.
- Using a new method, untested method to simulate the quantum gates while making translation into the classical world.



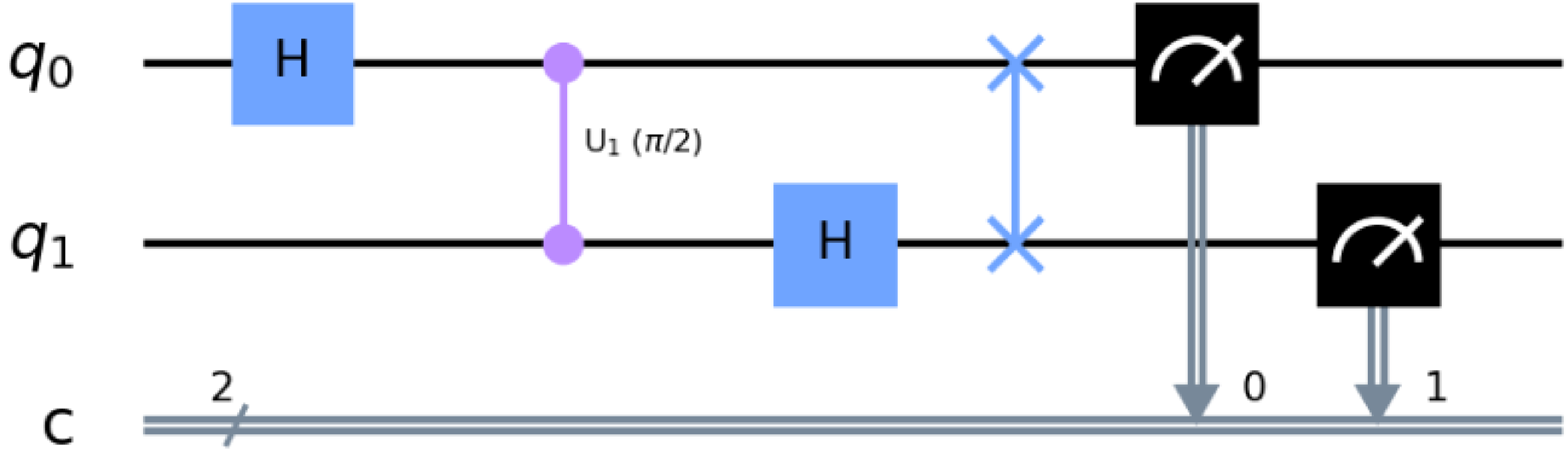


A little bit on quantum computing

- Qubits are bits which are built from phase and size. They represent bits of quantum computers.
- The main advantage of quantum computers, is that because of quantum theory nature they allow to solve some problems more efficiently than their classical counterpart.
- QFT – O($\log^2(n)$) , FFT – O($nlog(n)$)



QFT



$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, CP(\varphi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\varphi} \end{pmatrix}, SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Theoretical background – translating to the classical world

$$\varphi|0\rangle_{quantum} = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}_{classical} , \quad \varphi|1\rangle_{quantum} = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}_{classical}$$

$$|k\rangle = c_0|0\rangle + c_1|1\rangle = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} x_0 + iy_0 \\ x_1 + iy_1 \end{pmatrix} = \vec{x} + i\vec{y}$$

$$\varphi|k\rangle_{quantum} = \frac{1}{8}(\vec{u}_{classical} + \vec{p}_{classical}), \qquad \vec{u} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \qquad \vec{p} = \begin{pmatrix} \vec{x} \\ -\vec{x} \\ \vec{y} \\ -\vec{y} \end{pmatrix}$$

$$s_{12\dots n} = s_{1,n} = \frac{1}{8^n}\left(\vec{u}^{\otimes n} + \vec{p}_1 \otimes \vec{p}_2 \otimes \dots \otimes \vec{p}_n\right)$$

$$M[Gate] = \begin{pmatrix} \mathcal{Re}(Gate) & 0 & 0 & \mathcal{Im}(Gate) \\ 0 & \mathcal{Re}(Gate) & \mathcal{Im}(Gate) & 0 \\ \mathcal{Im}(Gate) & 0 & \mathcal{Re}(Gate) & 0 \\ 0 & \mathcal{Im}(Gate) & 0 & \mathcal{Re}(Gate) \end{pmatrix}$$

And if $Gate = A \otimes B + C$ we get

$$M[Gate] = M[A] \otimes M[B] + M[C]$$

$$Cgate_{1,n} = P_0 \otimes \mathbb{I}_{2^{n-1},2^{n-1}} + P_1 \otimes \mathbb{I}_{2^{n-2},2^{n-2}} \otimes gate$$

$$Cgate_{n,1} = \mathbb{I}_{2^{n-1},2^{n-1}} \otimes P_0 + \mathbb{I}_{2^{n-2},2^{n-2}} \otimes gate \otimes P_1$$

And therefore:

$$M\left[Cgate_{1,n}\right] = M[P_0] \otimes M\left[\mathbb{I}_{2^{n-1},2^{n-1}}\right] + M[P_1] \otimes M\left[\mathbb{I}_{2^{n-2},2^{n-2}}\right] \otimes M[gate]$$

$$M\left[Cgate_{n,1}\right] = M\left[\mathbb{I}_{2^{n-1},2^{n-1}}\right] \otimes M[P_0] + M\left[\mathbb{I}_{2^{n-2},2^{n-2}}\right] \otimes M[gate] \otimes M[P_1]$$

$$j < k\colon\ \ Cgate_{1,j,k,n} = \mathbb{I}_{2^{j-1},2^{j-1}} \otimes Cgate_{1,k-j+1} \otimes \mathbb{I}_{2^{n-k},2^{n-k}}$$

$$j > k\colon\ \ Cgate_{i,j,k,n} = \mathbb{I}_{2^{j-1},2^{j-1}} \otimes Cgate_{j-k+1,1} \otimes \mathbb{I}_{2^{n-k},2^{n-k}}$$

And therefore:

$$M\left[Cgate_{1,j,k,n}\right] = M\left[\mathbb{I}_{2^{j-1},2^{j-1}}\right] \otimes M\left[Cgate_{1,k-j+1}\right] \otimes M\left[\mathbb{I}_{2^{n-k},2^{n-k}}\right]$$

Example

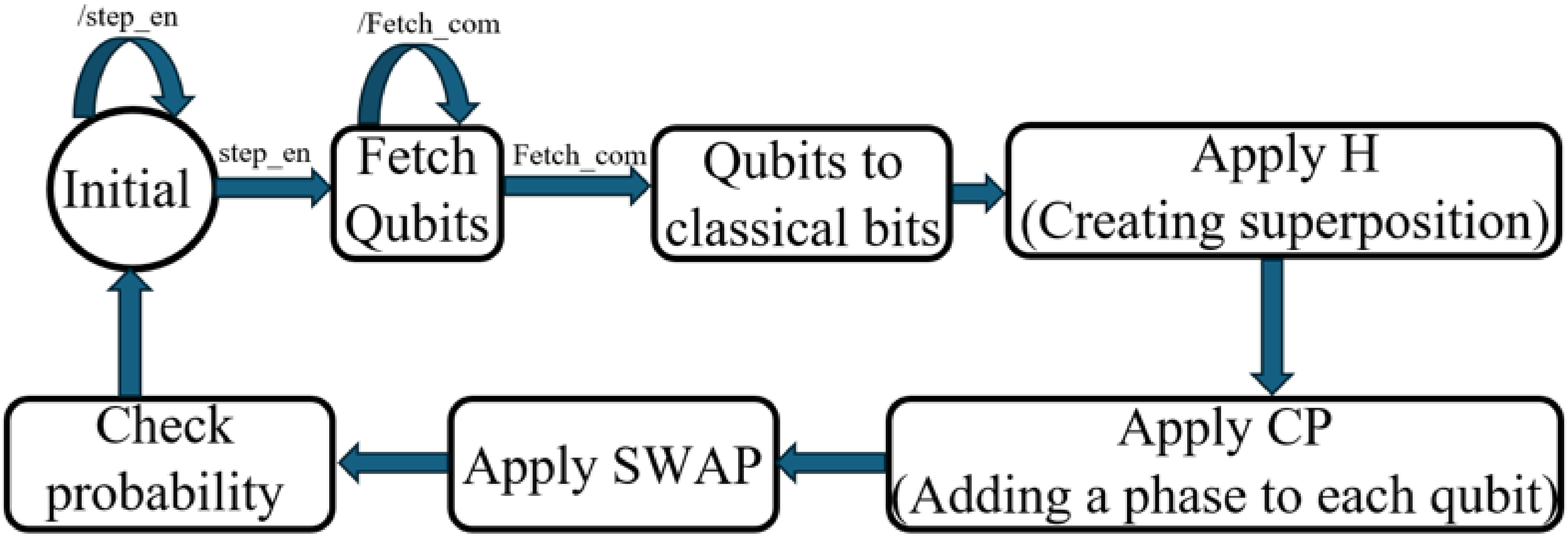
$$CP\left(\frac{\pi}{8}\right)_{1,4}^{4Qbits} = P_0 \otimes \mathbb{I}_{8x8} + P_1 \otimes \mathbb{I}_{4x4} \otimes P\left(\frac{\pi}{8}\right)$$

$$CNOT_{1,2,3,4}^{4Qbits} = \mathbb{I}_{2x2} \otimes (P_0 \otimes \mathbb{I}_{2x2} + P_1 \otimes NOT) \otimes \mathbb{I}_{2x2}$$

Data representation

- 2’s complement and floating point
- 24 bits per word to save memory while keeping high precision
- Broken down to 8 bits part as part of the UART-USB limitations
- Read from the serial port using python and rebuilt as the original 24 bits words

Block diagram



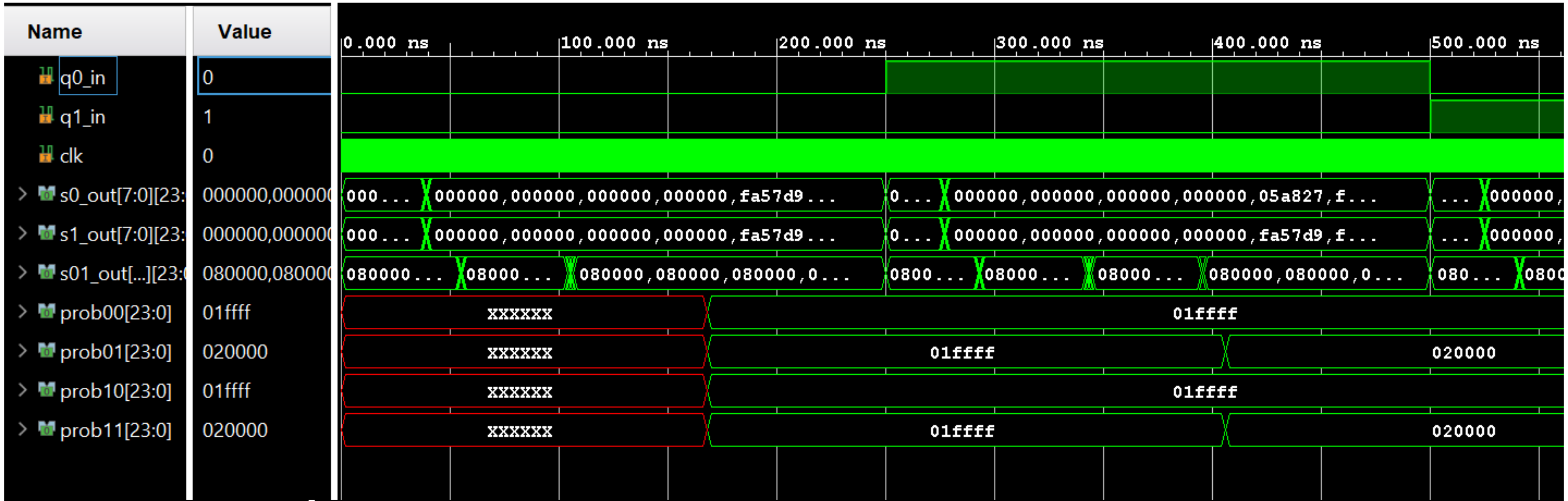
Xilinx VC709 connectivity board- החומרה

- 128 Mb memory flash
- 200Mhz default clock speed
- User switches/LEDs/buttons

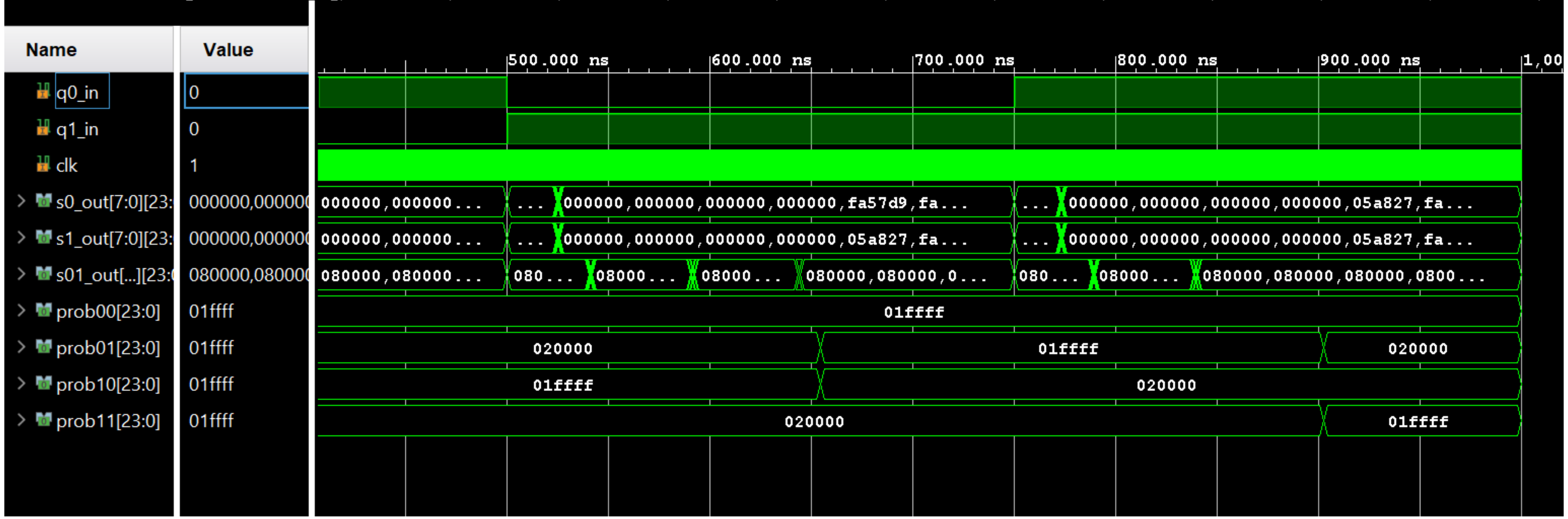
Implementation

- Python simulation
- System Verilog simulation
- Creating a communication bridge between the FPGA and PC using UART-USB bridge
- Creating the final FPGA firmware, burning and running it

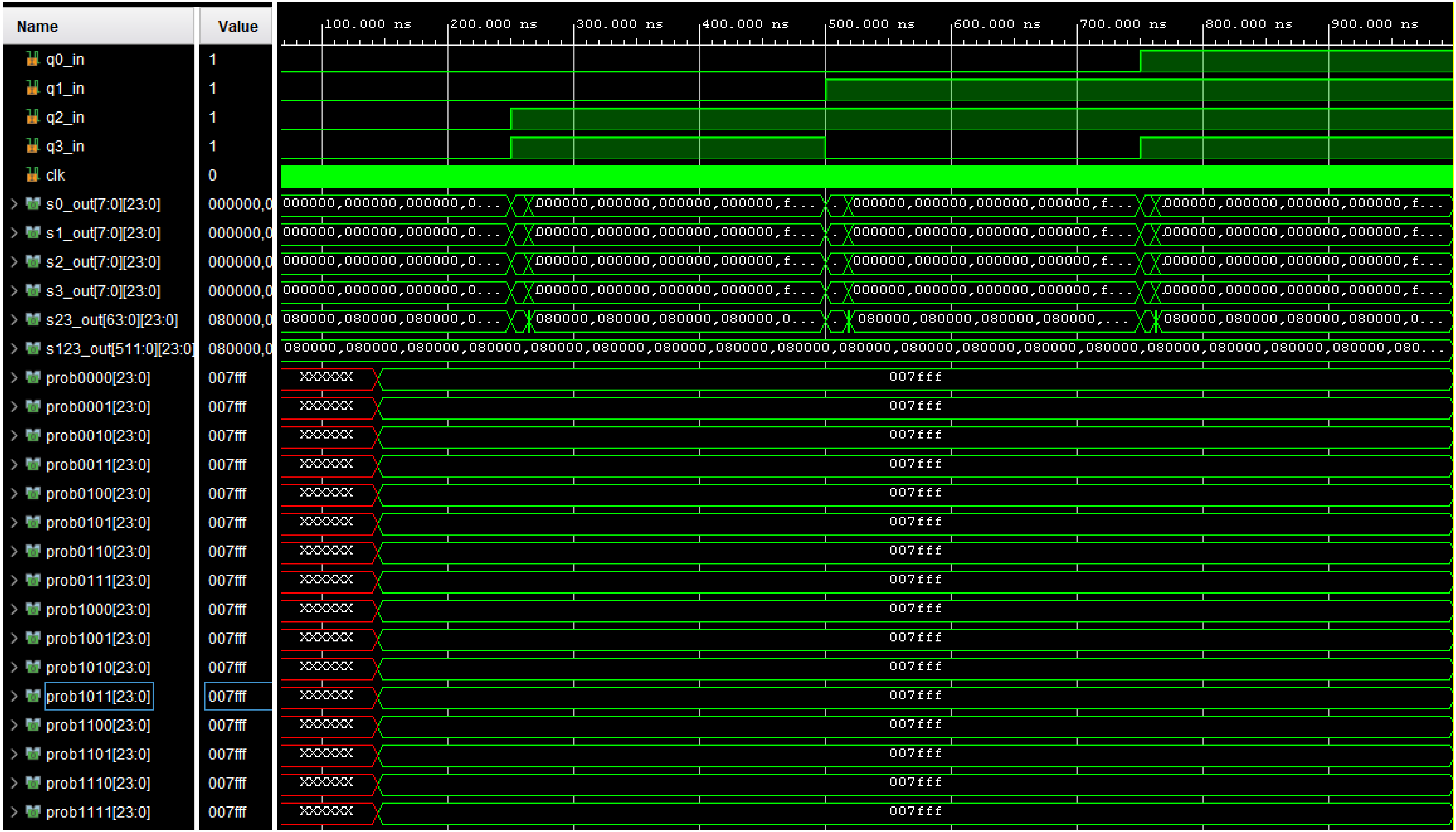
System Verilog simulation results



Name	Value
q0_in	0
q1_in	1
clk	0
s0_out[7:0][23:	000000,000000
s1_out[7:0][23:	000000,000000
s01_out[...][23:(	080000,080000
prob00[23:0]	01ffff
prob01[23:0]	020000
prob10[23:0]	01ffff
prob11[23:0]	020000

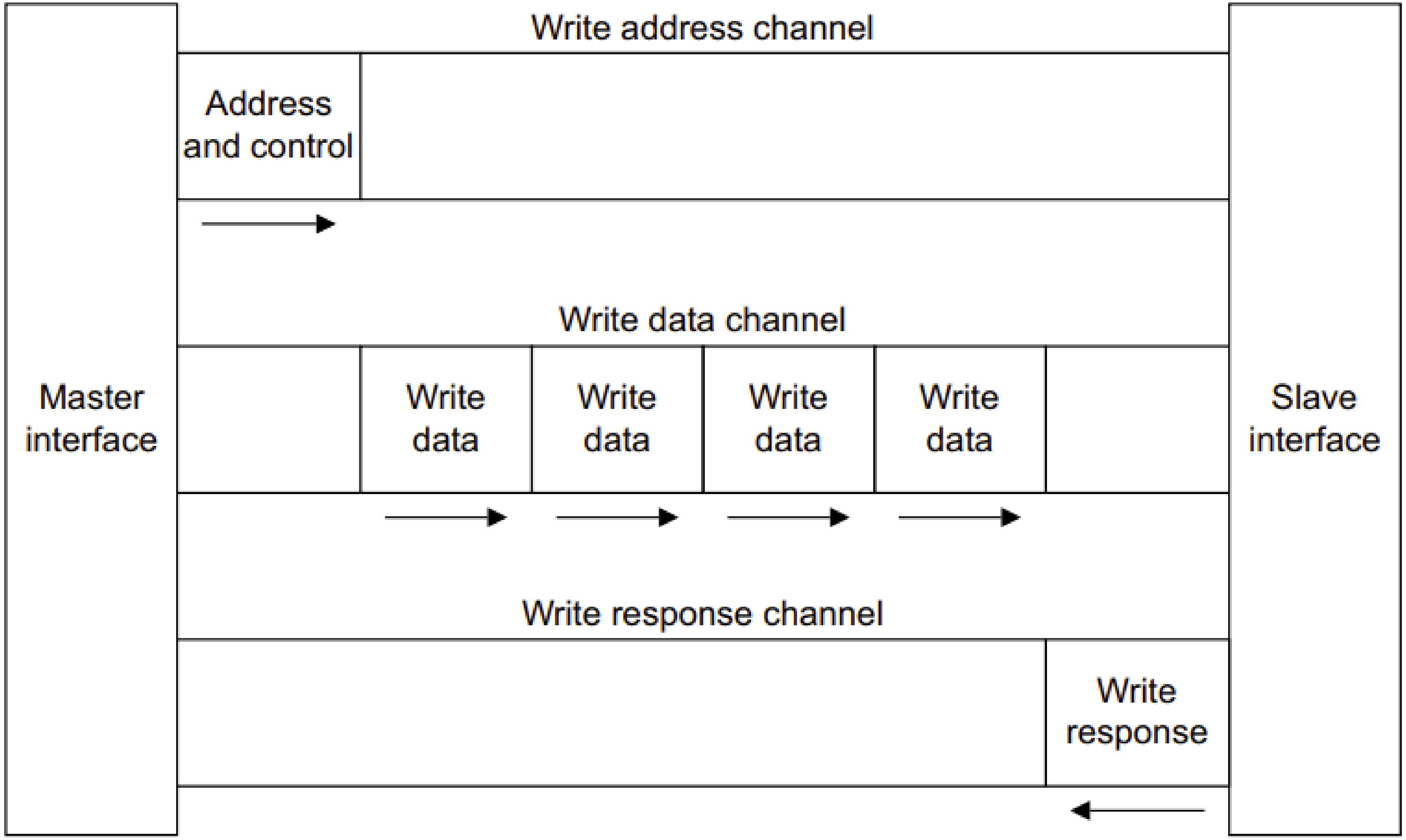


Name	Value
q0_in	0
q1_in	0
clk	1
s0_out[7:0][23:	000000,000000
s1_out[7:0][23:	000000,000000
s01_out[...][23:(	080000,080000
prob00[23:0]	01ffff
prob01[23:0]	01ffff
prob10[23:0]	01ffff
prob11[23:0]	01ffff



Name	Value
q0_in	1
q1_in	1
q2_in	1
q3_in	1
clk	0
s0_out[7:0][23:0]	000000,0
s1_out[7:0][23:0]	000000,0
s2_out[7:0][23:0]	000000,0
s3_out[7:0][23:0]	000000,0
s23_out[63:0][23:0]	080000,0
s123_out[511:0][23:0]	080000,0
prob0000[23:0]	007fff
prob0001[23:0]	007fff
prob0010[23:0]	007fff
prob0011[23:0]	007fff
prob0100[23:0]	007fff
prob0101[23:0]	007fff
prob0110[23:0]	007fff
prob0111[23:0]	007fff
prob1000[23:0]	007fff
prob1001[23:0]	007fff
prob1010[23:0]	007fff
prob1011[23:0]	007fff
prob1100[23:0]	007fff
prob1101[23:0]	007fff
prob1110[23:0]	007fff
prob1111[23:0]	007fff

UART-USB Bridge explained



Master interface

Write address channel

Address and control

Write data channel

Write data

Write data

Write data

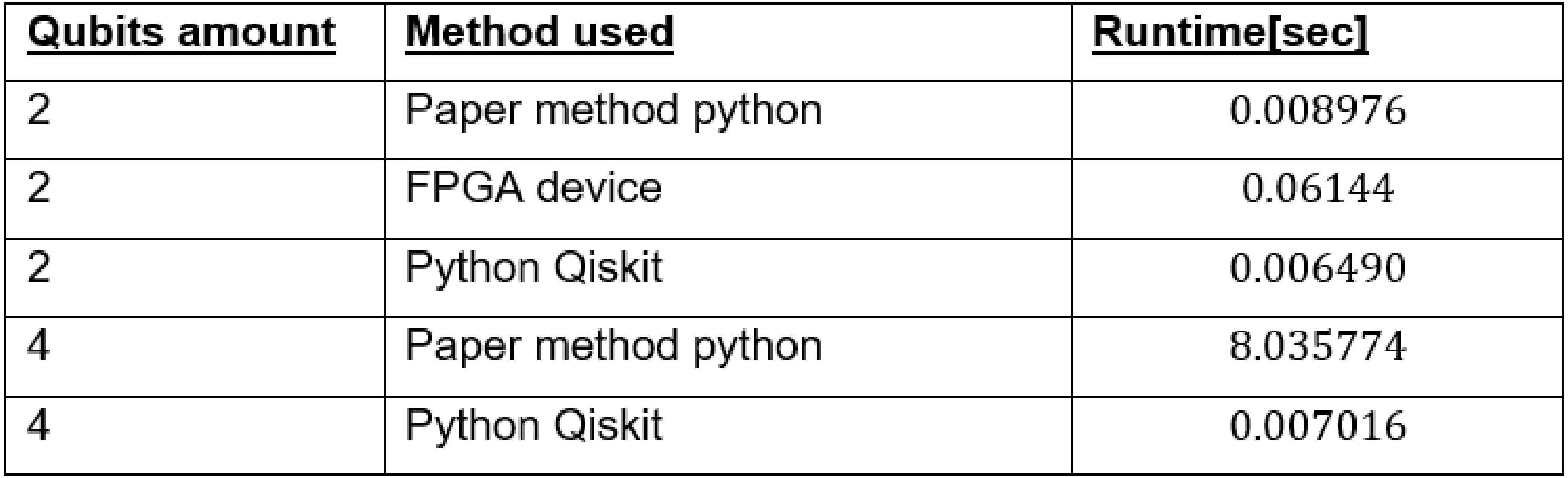
Write data

Write response channel

Write response

Slave interface

Project results



Qubits amount	Method used	Runtime[sec]
2	Paper method python	0.008976
2	FPGA device	0.06144
2	Python Qiskit	0.006490
4	Paper method python	8.035774
4	Python Qiskit	0.007016

Memory limitations

- 2 qubits - ~Kb per gate
- 4 qubits - ~150Kb per gate
- 6 qubits – ~10Mb per gate
- 8 qubits - ~650-750Mb per gate

.

.

.

.

Conclusion and future work

- Higher qubits implementations using the FPGA
- Testing alternative methods to simulate quantum circuits using an FPGA device
- Testing the effects on different quantum circuits, implementing different algorithms.
- Testing implementation with GPUs and comparing to FPGAs