Quantum Circuits for the Schur Transform

Oliver Thomas

University of Bristol
Quantum Engineering CDT

March 21, 2018

1 The Schur Transform

2 Streaming Scheme

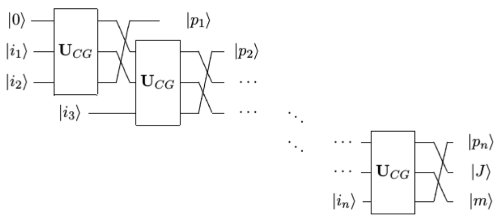
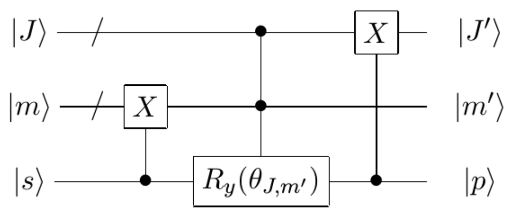


FIG. 2: Quantum circuit for the Schur transformation $\mathbf{U}_{\mathrm{Sch}}$, transforming between $|i_1 i_2 \cdots i_n\rangle$ and $|J, m, p\rangle$.

(a) Streaming structure. Taken from Bacon, Chaung, Harrow (2004) Arxiv /0407082v4



(b) U_{CG} block, Qadder, controlled rotation, Qadder. Taken from Bacon, Chaung, Harrow (2004) arXiv /0407082v4

The Rotation matrix, $R_y(\theta_{J,m'})$ is given by,

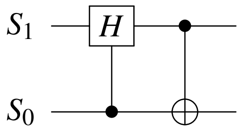
$$R_y(\theta_{J,m'}) = \begin{bmatrix} \cos(\theta_{J,m'}) & -\sin(\theta_{J,m'}) \\ \sin(\theta_{J,m'}) & \cos(\theta_{J,m'}) \end{bmatrix} \tag{1}$$

=

$$\frac{1}{\sqrt{2J+1}} \begin{bmatrix} \sqrt{J+\frac{1}{2}+m'} & -\sqrt{J+\frac{1}{2}-m'} \\ \sqrt{J+\frac{1}{2}-m'} & \sqrt{J+\frac{1}{2}+m'} \end{bmatrix} \tag{2}$$

Where primed variables means after the angular momentum addition so J is the total J that the spin is coupling to, the system will have J' total angular momentum after the coupling. m is the z component of the system before and m' is the total z component after the coupling.

The temporally multiplexed streaming scheme is shown in figures 4 & 5.



(3)

Figure 2: Schur transform for 2 qubits

3 Clebsch Gordan matrix transform

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{bmatrix} = \begin{bmatrix} |J=1,M=1\rangle \\ |J=1,M=0\rangle \\ |J=1,M=-1\rangle \\ |J=0,M=0\rangle \end{bmatrix} = (Intermsofspins) \begin{bmatrix} |00\rangle \\ \frac{1}{\sqrt{2}}(|01\rangle+|10\rangle) \\ |11\rangle \\ \frac{1}{\sqrt{2}}(|01\rangle-|10\rangle) \end{bmatrix} \tag{4}$$

Circuit for Clebsch-Gordan transform. Fig. 2

4 Spatially Multiplexed registered 2 qubit Schur transform- 12 Two-qubit gatesFig. 8Fig. 10

Spin values		Circuit output		M value
S_1	S_0	m_1	m_0	M
0	0	0	1	M=+1
0	1	0	0	M=0
1	0	0	0	M=0
1	1	1	0	M=-1

Figure 3: Table giving M register decoding for 2 qubit spatial multiplexing

Adding another CNOT allows you to encode the M' register using the same values for M'=0. Fig. 8 Fig. 8

This circuit can be minimised to 11 gates if the m register is not compressed. Spatially Multiplexed Minimal gate explicit J & M recording- 11 two qubits Fig. 10.

Circuit uses 11 two-qubit gates but only stores the final output values of J' & M'.

The HX gate triggers if $M=0$ meaning $S_0 \neq S_1$ which is implemented using an XOR between S_0 & S_1. The other gate (T) is triggered when $M=-1$ meaning $S_0 = S_1 = 1$ which is done using an AND (Toffoli) gate between, $S_0 = S_1$ AND $S_1 = 1$ which is decomposed into 5 two-qubit gates. T^2 is the ZX gate, meaning $T^2|S\rangle = XZ|S\rangle$ Fig. 10.

5 Reduced general gate circuit for up to the 2 qubit Schur transform Fig. 12

Circuit uses the encoding for $|S\rangle : |0\rangle \mapsto Spin = +\frac{1}{2}, |1\rangle \mapsto Spin = -\frac{1}{2}$ and the same for $|P\rangle$.

J_2	J_1	J_0	J
0	0	0	0
0	0	1	$\frac{1}{2}$
0	1	0	1
0	1	1	$\frac{3}{2}$
1	0	0	-2
1	0	1	-$\frac{3}{2}$
1	1	0	-1
1	1	1	-$\frac{1}{2}$

m_2	m_1	m_0	M
0	0	0	0
0	0	1	$\frac{1}{2}$
0	1	0	1
0	1	1	$\frac{3}{2}$
1	0	0	-2
1	0	1	-$\frac{3}{2}$
1	1	0	-1
1	1	1	-$\frac{1}{2}$

Figure 4: Tables giving binary Two's complement encoding to spin values of the M and J registers

Where V is the phase gate, $V = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$, $V^\dagger V = I$ and $V^2 = Z$. V is used here to expand the double controlled Toffoli gate into single control gates in the quantum adder subroutine.

The W gate, $W^2 = HX$ with $W^\dagger = I$, is used to expand the HX gate into single control gates in the spin transform region.

The circuit checks that if (m_1 XNOR m_0) AND (m_0 XOR S_0) and will then change m_2. Then m_1 is updated using $m_1 = m_0$ XOR S_0. m_0 is always incremented by 1, if $|S\rangle = |0\rangle$ increment only m_0 by 1 corresponding to adding $\frac{1}{2}$ to the M register. $|S\rangle = |1\rangle$ corresponds to subtracting $\frac{1}{2}$ from the M register by adding the string 111 bitwise to M.

For the most positive values of M the Identity is performed on the spin corresponding to the strings $M = 001(J = \frac{1}{2}, M' = \frac{1}{2})$ for the first spin and $M = 010(J = 1, M' = 1)$ for the second coupled in spins.

The most negative values of M performs $XZ|S\rangle$ corresponding to the strings $M = 111(J = \frac{1}{2}, M' = -\frac{1}{2})$ for the first spin and $M = 110(J = 1, M' = -1)$ for the second spin.

If $M = 000(J = 0, M' = 0)$ do $XH|S\rangle$ **??**.

6 Clebsch-Gordan coefficients for 3 qubits

matrix for transform is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & 0 & \sqrt{\frac{1}{3}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{\frac{1}{3}} & 0 & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \sqrt{\frac{2}{3}} & -\sqrt{\frac{1}{6}} & 0 & -\sqrt{\frac{1}{6}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{\frac{1}{6}} & 0 & \sqrt{\frac{1}{6}} & -\sqrt{\frac{2}{3}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 \end{bmatrix} \begin{bmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{bmatrix} = \begin{bmatrix} |J = 3/2, M = 3/2\rangle \\ |J = 3/2, M = 1/2\rangle \\ |J = 3/2, M = -1/2\rangle \\ |J = 3/2, M = -3/2\rangle \\ \hline |J = 1/2, M = 1/2, P = 0\rangle \\ |J = 1/2, M = -1/2, P = 0\rangle \\ \hline |J = 1/2, M = 1/2, P = 1\rangle \\ |J = 1/2, M = -1/2, P = 1\rangle \end{bmatrix} \tag{5}$$

$J=\frac{3}{2}$ (P=000 j=1/2, j=1, j=3/2)	$S=\frac{3}{2}$
000	$M=\frac{3}{2}$
$\sqrt{\frac{1}{3}}(001+010+100)$	$M=\frac{1}{2}$
$\sqrt{\frac{1}{3}}(110+011+101)$	$M=-\frac{1}{2}$
111	$M=-\frac{3}{2}$
$J=\frac{1}{2}$ (P=001 j=1/2, j=1, j=1/2)	$S=\frac{1}{2}$
$\sqrt{\frac{2}{3}}(001)-\sqrt{\frac{1}{6}}(010+100)$	$M=\frac{1}{2}$
$-\sqrt{\frac{2}{3}}(110)+\sqrt{\frac{1}{6}}(011+101)$	$M=-\frac{1}{2}$
$J=\frac{1}{2}$ (P=010 j=1/2, j=0, j=1/2)	$S=\frac{1}{2}$
$\frac{1}{\sqrt{2}}(010-100)$	$M=\frac{1}{2}$
$\frac{1}{\sqrt{2}}(011-101)$	$M=-\frac{1}{2}$

Figure 5: J & M values for 3 qubits using encoding 0=spin up, 1=spin down

Rearranging,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sqrt{\frac{2}{3}} & -\sqrt{\frac{1}{6}} & -\sqrt{\frac{1}{6}} & 0 & 0 & 0 & 0 \\ 0 & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{3}} & 0 \\ 0 & 0 & 0 & 0 & \sqrt{\frac{1}{6}} & \sqrt{\frac{1}{6}} & -\sqrt{\frac{2}{3}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 000 \\ 001 \\ 010 \\ 100 \\ 011 \\ 101 \\ 110 \\ 111 \end{bmatrix} \tag{6}$$

have to swap 011 & 100 to get block form.

$J=\frac{3}{2}$	$S=\frac{3}{2}$
000	$M=\frac{3}{2}$
$\sqrt{\frac{1}{3}}(001+010+100)$	$M=\frac{1}{2}$
$\sqrt{\frac{1}{3}}(110+011+101)$	$M=-\frac{1}{2}$
111	$M=-\frac{3}{2}$
$J=\frac{1}{2}, P=0$	$S=\frac{1}{2}$
$\frac{1}{\sqrt{3}}(e^{2\pi i/3}001+e^{4\pi i/3}010+100)$	$M=\frac{1}{2}$
$\frac{1}{\sqrt{3}}(e^{2\pi i/3}011+e^{4\pi i/3}101+110)$	$M=-\frac{1}{2}$
$J=\frac{1}{2}, P=1$	$S=\frac{1}{2}$
$\frac{1}{\sqrt{3}}(e^{2\pi i/3}001+e^{4\pi i/3}010+100)$	$M=\frac{1}{2}$
$\frac{1}{\sqrt{3}}(e^{4\pi i/3}011+e^{2\pi i/3}101+110)$	$M=-\frac{1}{2}$

Figure 6: Schur transform with Phase encoding?

Using the Givens rotation method for unitary decomposition into a gateset this matrix can be expressed as a product of 19 CC-U gates which is 100 C-U gates.

$$\frac{1}{\sqrt{3}}\begin{bmatrix} \sqrt{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{3} \\ 0 & e^{2\pi i/3} & e^{4\pi i/3} & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{2\pi i/3} & 0 & e^{4\pi i/3} & 1 & 0 \\ 0 & e^{4\pi i/3} & e^{2\pi i/3} & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & e^{4\pi i/3} & 0 & e^{2\pi i/3} & 1 & 0 \end{bmatrix}\begin{bmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{bmatrix} = \begin{bmatrix} |J=3/2, M=3/2\rangle \\ |J=3/2, M=1/2\rangle \\ |J=3/2, M=-1/2\rangle \\ |J=3/2, M=-3/2\rangle \\ \hline |J=1/2, M=1/2, P=0\rangle \\ |J=1/2, M=-1/2, P=0\rangle \\ \hline |J=1/2, M=1/2, P=1\rangle \\ |J=1/2, M=-1/2, P=1\rangle \end{bmatrix} \tag{7}$$

This matrix will have a different decomposition which may be more or less efficient.

7 Circuit for 3 qubit transform

---

see github for Fortran code, using Givens rotations gives 19 cc-unitary gates which is about 100 c-unitaries.

8 General circuit for the Quantum Schur transform ($|S\rangle$) Fig. 13

J_2	J_1	J_0	J
0	0	0	0
0	0	1	$\frac{1}{2}$
0	1	0	1
0	1	1	$\frac{3}{2}$
1	0	0	-2
1	0	1	-$\frac{3}{2}$
1	1	0	-1
1	1	1	-$\frac{1}{2}$

m_2	m_1	m_0	M
0	0	0	0
0	0	1	$\frac{1}{2}$
0	1	0	1
0	1	1	$\frac{3}{2}$
1	0	0	-2
1	0	1	-$\frac{3}{2}$
1	1	0	-1
1	1	1	-$\frac{1}{2}$

Figure 7: Tables giving binary Two's complement encoding to spin values of the M and J registers

Circuit uses the encoding for $|S\rangle : |0\rangle \mapsto Spin = +\frac{1}{2}, |1\rangle \mapsto Spin = -\frac{1}{2}$ and the same for $|P\rangle$.

The circuit adds the value of the spin to be added, $|S\rangle$, to the M register to calculate the M' register value. This is done by implementing the quantum reversible equivalent to the digital full adder.

The case where $|S\rangle = |0\rangle$ means the spin is $+\frac{1}{2}$ so to add $\frac{1}{2}$ to M one is added to the m_0 bit. The very first Quantum Adder (QAdd) uses Toffoli gates controlled on $|0\rangle$ on $|s\rangle$ (denoted by the white control circle) with the current m_0 value and C_0 (an ancilla carry) so that in the case $m_0 = 1$ and we try and add 1 to it, m_0 goes to 0 and m_1 is increased using the carry as $001 + 1 = 010$. The rest of the QAdd stages then just check the carry of the previous qubit to complete to M $+\frac{1}{2}$ addition as $|S\rangle = |0\rangle$ does not trigger any of the rest of the control gates.

The case where $|S\rangle = |1\rangle$ means the spin is $-\frac{1}{2}$ we do M $-\frac{1}{2}$ which is done by adding the binary string for $-\frac{1}{2}$ which is the all 1's string, 111. This time the very first Quantum Adder does not trigger and $|s\rangle$ is then added to all of the bits of M using C-NOT gates with carries to check for overflow.

The Unitary is then performed on $|S\rangle$ depending on the values of the newly calculated M' and J registers. The Identity is shown in the circuit for completeness on all the J and M' values. The J register is then updated to J' by adding the value of $|P\rangle$ to J using the QAdd sequence of gates.

To add the second qubit in the values of J' and M' are passed in as the initial register values. It is easy to extend this to many qubits being streamed in one at a time by carefully conditioning the controls on the unitaries, I think in the general case you need at most N controls for coupling up to N qubits in one at a time. The circuit written here has redundancy in the Identity and ZX gates appearing twice Fig. 13.

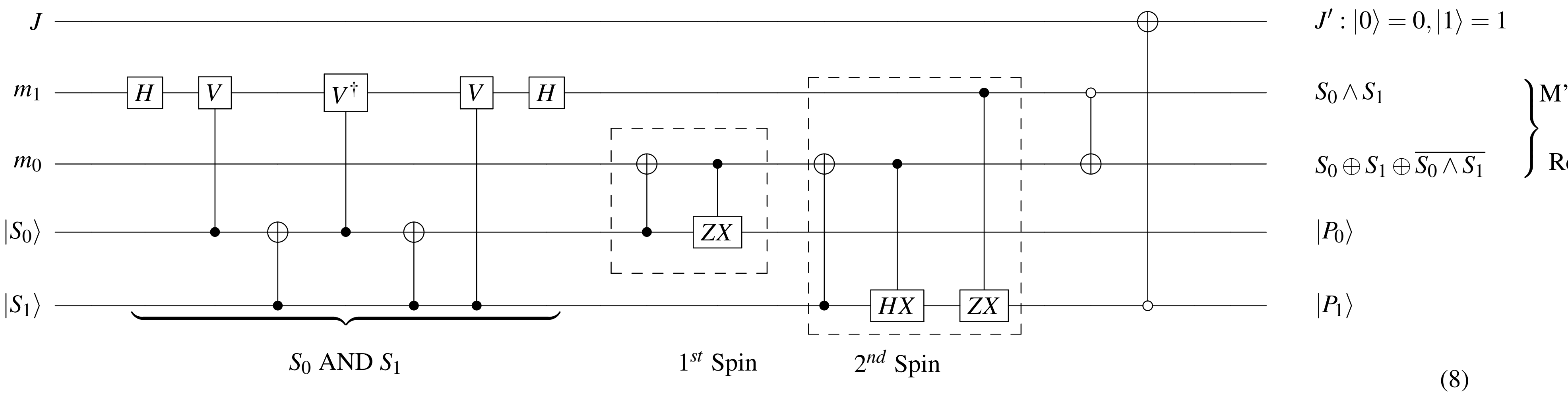
9 4 Qubit CG coefficients

$J' = 2$, (P=0000 j=1/2, j=1, j=3/2, j=2)	$S = 2$
0000	$M = 2$
$\frac{1}{2}(0001 + 0010 + 0100 + 1000)$	$M = 1$
$\sqrt{\frac{1}{6}}(0011 + 0101 + 1001 + 1100 + 1010 + 0110)$	$M = 0$
$\frac{1}{2}(1110 + 1101 + 1011 + 0111)$	$M = -1$
1111	$M = -2$
$J' = 1$, (P=0001 j=1/2, j=1, j=3/2, j=1)	$S = 1$
$\sqrt{\frac{3}{4}}(0001) - \sqrt{\frac{1}{12}}(0010 + 0100 + 1000)$	$M = 1$
$\sqrt{\frac{1}{6}}((0011 + 0101 + 1001) - (1100 + 1010 + 0110))$	$M = 0$
$-\sqrt{\frac{3}{4}}(1110) + \sqrt{\frac{1}{12}}(1101 + 1011 + 0111)$	$M = -1$
$J' = 1$, (P=0010 j=1/2, j=1, j=1/2, j=1)	$S = 1$
$\sqrt{\frac{2}{3}}(0010) - \sqrt{\frac{1}{6}}(0100 + 1000)$	$M = 1$
$\sqrt{\frac{1}{3}}(0011 - 1100) + \sqrt{\frac{1}{12}}(0110 + 1010 - 0101 - 1001)$	$M = 0$
$-\sqrt{\frac{2}{3}}(1101) + \sqrt{\frac{1}{6}}(1011 + 0111)$	$M = -1$
$J' = 1$, (P=0100 j=1/2, j=0, j=1/2, j=1)	$S = 1$
$\sqrt{\frac{1}{2}}(0100 - 1000)$	$M = 1$
$\frac{1}{2}(0101 - 1001 + 0110 - 1010)$	$M = 0$
$-\sqrt{\frac{1}{2}}(0111 - 1011)$	$M = -1$
$J' = 0$, (P=0011 j=1/2, j=1, j=1/2, j=0)	$S = 0$
$\sqrt{\frac{1}{3}}(0011 + 1100) - \sqrt{\frac{1}{12}}(0101 + 1001 + 0110 + 1010)$	$M = 0$
$J' = 0$, (P=0101 j=1/2, j=0, j=1/2, j=0)	$S = 0$
$\frac{1}{2}(0101 - 1001 - 0110 + 1010)$	$M = 0$

p is defined as $J' - J$

References

Appendix- Circuits



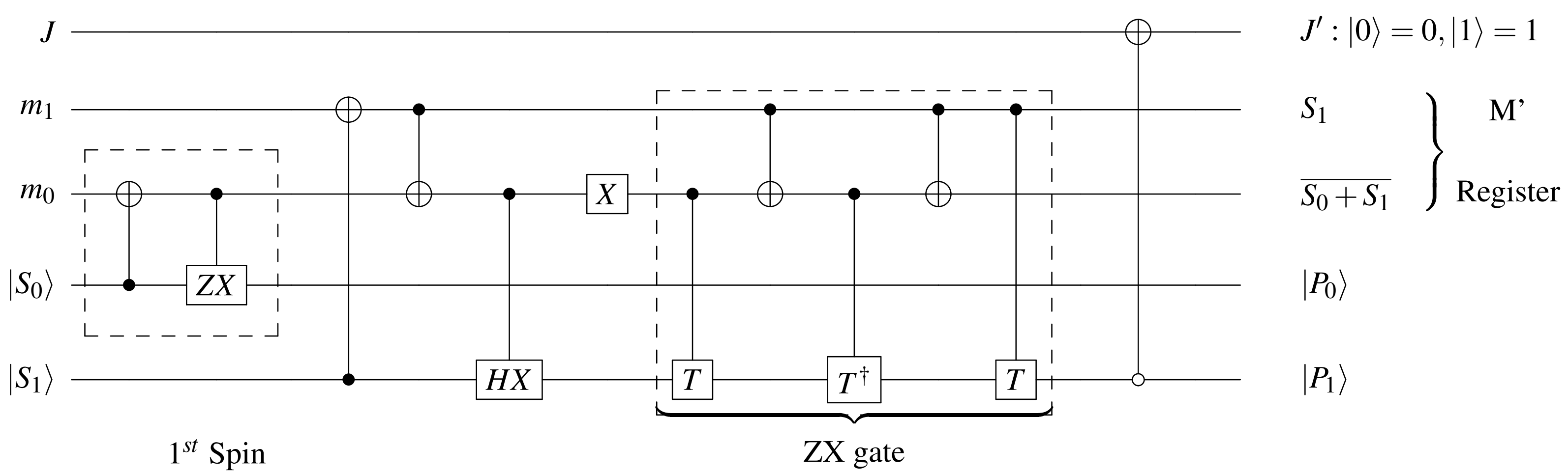
(8)

∞

Figure 8: Spatial multiplexed 2 qubit

Spin values		Circuit output		M value
S_1	S_0	m_1	m_0	M
0	0	0	1	M=+1
0	1	0	0	M=0
1	0	0	0	M=0
1	1	1	0	M=-1

Figure 9: Table giving M register decoding for 2 qubit spatial multiplexing



(9)

Figure 10: minimal gate spatial multiplexing

Spin values		Circuit output		M value
S_1	S_0	S_1	$\overline{S_0 + S_1}$	M
0	0	0	1	M=+1
0	1	0	0	M=0
1	0	1	0	M=0
1	1	1	1	M=-1

Figure 11: Table giving M register decoding for minimal gate number

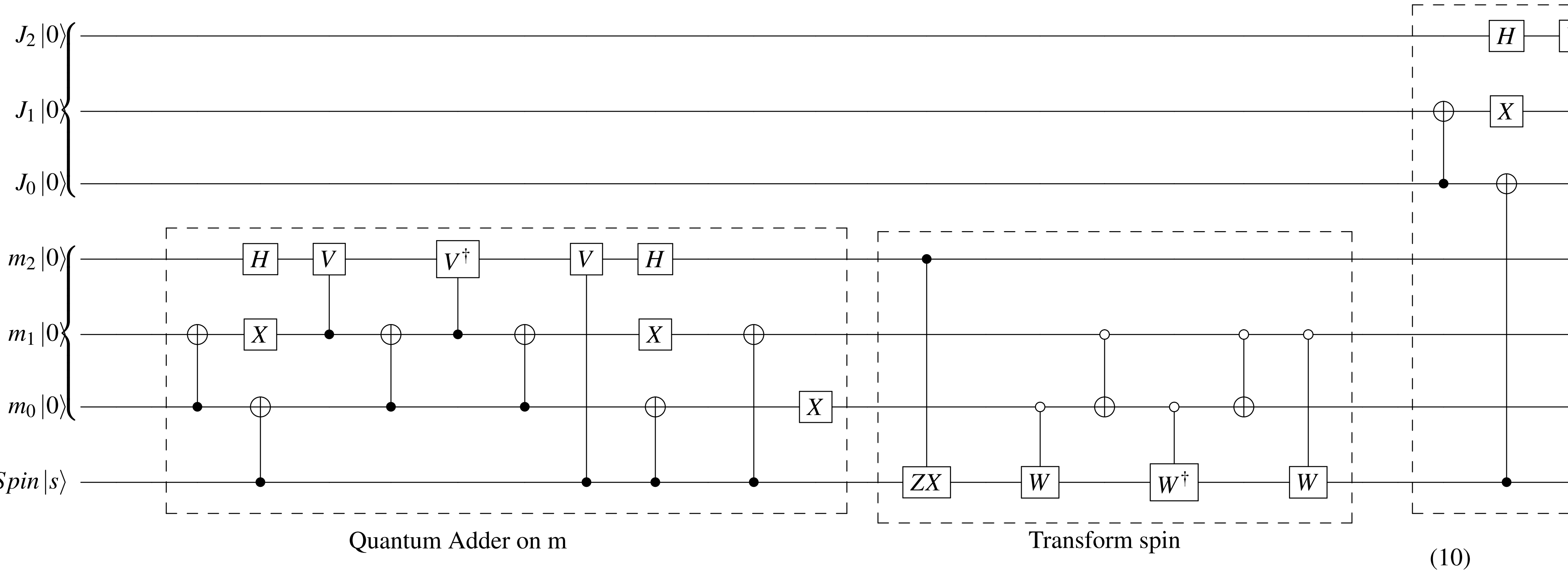


Figure 12: temporal multiplexed streaming

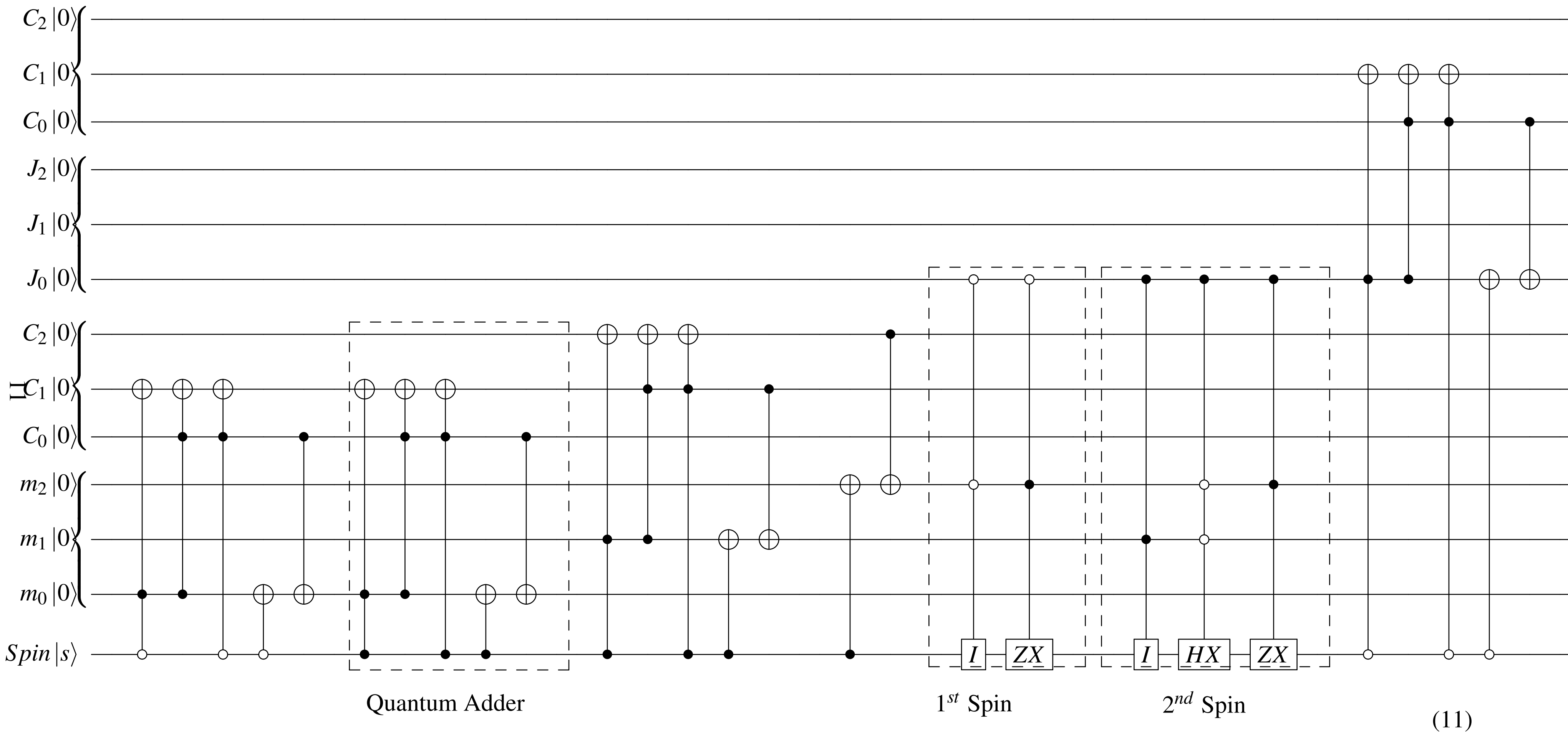


Figure 13: general streaming circuit

J_2	J_1	J_0	J
0	0	0	0
0	0	1	$\frac{1}{2}$
0	1	0	1
0	1	1	$\frac{3}{2}$
1	0	0	-2
1	0	1	-$\frac{3}{2}$
1	1	0	-1
1	1	1	-$\frac{1}{2}$

m_2	m_1	m_0	M
0	0	0	0
0	0	1	$\frac{1}{2}$
0	1	0	1
0	1	1	$\frac{3}{2}$
1	0	0	-2
1	0	1	-$\frac{3}{2}$
1	1	0	-1
1	1	1	-$\frac{1}{2}$

Figure 14: Tables giving binary Two's complement encoding to spin values of the M and J registers

Appendix- Fortran code

```
!Oliver Thomas 2018 Bristol
program matrixmul
implicit none

integer, parameter :: dp=selected_real_kind(15,300)
integer :: n, qubits, numofdecomp, i, j, counter
integer, allocatable, dimension(:) :: p

real(kind=dp), parameter :: invr2=1/sqrt(real(2,kind=dp)), invr3=1/sqrt(real(3,kind=d
real(kind=dp), parameter :: r2=sqrt(real(2,kind=dp))

real(kind=dp), allocatable, dimension(:,:) :: unitary, ident, uprod
real(kind=dp), allocatable, dimension(:,:,:)  :: u, gateseq

counter=1
print*, 'Enter number of qubits, 2 or 3'
read*, qubits
n= 2**qubits
numofdecomp=int(n*(n-1)/2.0_dp)

allocate(ident(n,n))
allocate(unitary(n,n))
allocate(u(n,n,n*n))
allocate(uprod(n,n))
allocate(p(n))
allocate(gateseq(n,n,n*n))


ident=0.0_dp
unitary=0.0_dp
u=0.0_dp
gateseq=0.0_dp

!#make identity
ident=identity(n)

!#make u's ident
do i=1, size(u,3)
  u(:,:,i)=identity(n)
end do
gateseq=u
!#init uprod as ident
uprod=identity(n)
```

```
!#write(*,*) int(ident)

!#make unitary
if (qubits==2) then
  p(1:n)=(/1,2,4,3/)

  unitary(1:n,1)=(/1.0_dp, 0.0_dp, 0.0_dp, 0.0_dp/)
  unitary(1:n,2)=(/0.0_dp, invr2, invr2, 0.0_dp/)
  unitary(1:n,3)=(/0.0_dp, 0.0_dp, 0.0_dp, 1.0_dp/)
  unitary(1:n,4)=(/0.0_dp, invr2, -invr2,0.0_dp/)

else if (qubits==3) then

  p(1:n)=(/1,2,4,3,7,8,6,5/)
!# col,row
  unitary(1,1)=1.0_dp

  unitary(2,2)=invr3
  unitary(2,5)=r2*invr3

  unitary(3,2)=invr3
  unitary(3,5)=-invr6
  unitary(3,7)=invr2

  unitary(4,3)=invr3
  unitary(4,6)=invr6
  unitary(4,8)=invr2

  unitary(5,2)=invr3
  unitary(5,5)=-invr6
  unitary(5,7)=-invr2

  unitary(6,3)=invr3
  unitary(6,6)=invr6
  unitary(6,8)=-invr2

  unitary(7,3)=invr3
  unitary(7,6)=-r2*invr3

  unitary(8,4)=1.0_dp
end if

write(*,*) unitary

!#!!! make unitary gates

!#print*, p(n), p(n-1), p(1)
```

```
!#write(*,*) uprod
!#write(*,*) u(:,:,1)
uprod=unitary
do i=1,n !#col
  do j=1,n-1 !#row
    !#write(*,*)
    !#write(*,*) uprod
!#print*,
!#print*, p(n+1-j
    if(p(n-j+1).ne.p(i)) then
      call makeunitary(p(n-j),p(n-j+1),p(i), uprod, u(:,:,(i-1)*n+j))
    end if
  end do
end do

print*,
print*, 'unitaries'

uprod=unitary
do i=1, n*n
  if (icheck(u(:,:,i))==0) then
    print*, 'u non identity',i
    write(*,*) u(:,:,i)
    print*,
    uprod=matmul(uprod(:,:),u(:,:,i))
    write(*,*) uprod
    print*,
  end if
end do

!uprod=ident


print*,
print*, unitarycheck(u,unitary)
!if (unitarycheck(u,unitary)==1) then
  print*, 'THIS IS UNITARY'
  call invert(u,gateseq,counter)
  call gateset(gateseq)
!end if

do i=1,n*n
if (icheck(gateseq(:,:,i))==0) then

!write(*,*) gateseq(:,:,i)
print*,
end if
```

```
end do

do i=1, counter
  uprod=matmul(uprod,gateseq(:,:,i))
end do

print*, '------------------------------------------------------------------'
print*, 'Unitary matrix from', counter-1, 'gates'
print*, '------------------------------------------------------------------'

print*,
write(*,*) uprod
!write(*,*) icheck(uprod)
!#write(*,*) matmul(gateseq(:,:,1),gateseq(:,:,2))


deallocate(ident)
deallocate(unitary)
deallocate(u)
deallocate(uprod)
deallocate(p)
deallocate(gateseq)

!#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
contains

!# print non identity elements
subroutine gateset(matrix)
  real(kind=dp), dimension(:,:,:), intent(in) :: matrix
  integer :: n, i
  n=size(matrix,3)
  do i=1, n
    if (icheck(matrix(:,:,i))==0) then
      print*, 'gate', i
      write(*,*) matrix(:,:,i)
      print*,
    end if
  end do
end subroutine gateset

!# transpose and invert array
subroutine invert(matrix,inverted,count)
  real(kind=dp), dimension(:,:,:), intent(inout) :: inverted
  real(kind=dp), dimension(:,:,:), intent(in) :: matrix
  integer :: i, n, count
  n=size(matrix,3)
  count=1
```

```
  do i=1,n
    if (icheck(matrix(:,:,n-i+1))==0) then
      inverted(:,:,count) = transpose(matrix(:,:,n-i+1))
      count=count+1
    end if
  end do

end subroutine invert

!#!!!! Check product gives identity
function unitarycheck(umatrices, uni)
  real(kind=dp) :: unitarycheck
  real(kind=dp), dimension(:,:,:), intent(in) :: umatrices
  real(kind=dp), dimension(:,:), intent(in) :: uni
  real(kind=dp), dimension(:,:), allocatable :: uprod
  integer :: i, j

unitarycheck=1
uprod=uni
!#write(*,*) uprod
!#print*,
!#do u_n*u_n-1*...*u1*Unitary=Ident
do i=1, size(umatrices,3)-1
  if (icheck(umatrices(:,:,n*n-i))==0) then
    uprod=matmul(uprod(:,:), umatrices(:,:,i))
   !# write(*,*) uprod
   !# write(*,*)
  end if
end do

unitarycheck=icheck(uprod)

end function unitarycheck

!#!!! Calc givens rotation
subroutine givensrot(a, b, c, s, r)
  real(kind=dp) :: a, b, c, s, r, h, d
h=0.0
d=0.0
!#print*,
 !#write(*,*) 'a',a,'b',b,'c',c,'s',s
if (abs(b)>=1e-1) then
  h=hypot(a,b)
  d=1.0_dp/h
  c=abs(a)*d
  s=sign(d,a)*b
  r=sign(1.0_dp,a)*h
```

```
else
  c=1.0_dp
  s=0.0_dp
  r=a

end if
end subroutine givensrot

!#!!! find type of u
subroutine makeunitary(row1,row2,col,ucurrent, ugate)
  real(kind=dp) :: c,s,r
  real(kind=dp), dimension(:,:), intent(inout) :: ucurrent, ugate
  integer, intent(in) :: row1, row2, col

  ugate=identity(size(ucurrent,1))
  c=0.0
  s=0.0
  r=0.0

!#print*, 'r1',row1,'r2',row2,'col',col
call givensrot(ucurrent(col,row1), ucurrent(col,row2), c,s,r)
!#print*, 'c=', c, 's=', s
  ugate(row1,row1)=c
  ugate(row1,row2)=-s
  ugate(row2,row1)=s
  ugate(row2,row2)=c
 if (icheck(ugate)==0) then
  !#print*, 'ugate'
  !#write(*,*) ugate
  !#print*,
end if
ucurrent=matmul(ucurrent,ugate)

end subroutine makeunitary

!#!!!!! make identiy matrix dim n
function identity(n)
  real(kind=dp), dimension(n,n) :: identity
  integer :: n, i

identity=0.0_dp
do i=1, n
  identity(i,i) =1.0_dp
end do
end function identity
```

```
!#!!!! Check product gives identity
function icheck(uni)
  real(kind=dp) :: icheck
  real(kind=dp), dimension(:,:), intent(in) :: uni
  integer :: i, j

icheck=1

!#check ident
itest:do i=1, size(uni,1)
  do j=1, size(uni,1)
    if (i.ne.j) then
      if (abs(uni(i,j))>=1e-10) then
        print*, 'not ident off diag'
        icheck=0
        exit itest
      end if
    else if (i.eq.j) then
      if ((abs(uni(i,j))-1)>=1e-10) then
        print*, 'not identity diag'
        icheck=0
        exit itest
      end if
    end if
  end do
end do itest

end function icheck

end program matrixmul
```