Silicon qubits Delft

This device is based on reference: https://arxiv.org/abs/2202.09252

VQD setup

Set the main directory as the current directory

In[300]:=

```
SetDirectory[NotebookDirectory[]];
```

Load the QuESTLink package
One may also use the off-line questlink.m file, change it to the location of the local file

In[301]:=

```
Import["https://qtechtheory.org/questlink.m"]
```

This will download a binary file **quest_link** if there is no such file found
Otherwise, use a locally-compiled that called **quest_link***

In[302]:=

```
(* Search for existing files that match the pattern "quest_link*" *)
With[{questLinkFiles = Sort@FileNames["quest_link*", {NotebookDirectory[]}]}
 ,
 If[Length[questLinkFiles] > 0,
   (* If one or more matching files are found,
   use the first one alphabetically *)
   Print["Using the existing link file: ", First@questLinkFiles];
   CreateLocalQuESTEnv[First@questLinkFiles];
   ,
   (*If no matching files are found, download the link file*)
   Print["No link file found, download quest_link"];
   CreateDownloadedQuESTEnv[];
  ];
]
```

Load the **VQD** package; must be loaded after QuESTlink is loaded

In[303]:=

```
Get["../vqd.wl"]
```

Set the default configuration of the virtual Silicon device

frequency unit: ***MHz***
time unit: ***μs***

In[304]:=

```
Options[SiliconDelft] =
  {
   QubitNum → 6
   ,
   (* average of T1 *)
   T1 → 10^4
   ,
   (* In practice, T2 is obtained by echo RX[π/2]-τ-RX[π]-τ-RX[π/2],
   where τ=1μs. We assume the T2* is echoed out to T2 *)
   T2 → <|0 → 14, 1 → 21.1, 2 → 40.1, 3 → 37.2, 4 → 44.7, 5 → 26.7|>
   ,
   (* Qubit frequency of each qubit *)
   QubitFreq → <|0 → 15.62 * 10^3, 1 → 15.88 * 10^3,
     2 → 16.3 * 10^3, 3 → 16.1 * 10^3, 4 → 15.9 * 10^3, 5 → 15.69 * 10^3|>
   ,
   (* Rabi frequency of single rotations on each qubit *)
   RabiFreq → <|0 → 5, 1 → 5, 2 → 5, 3 → 5, 4 → 5, 5 → 5|>
   ,
   (* Set the noise form of off-resonant Rabi
     oscillation. This takes RabiFreq information to produce the noise.*)
   OffResonantRabi → True
   ,
   (* Set the standard depolarising
    and dephasing passive noise using T1 and T2 *)
   StdPassiveNoise → True
   ,
   (* Fidelities of X- and Y- rotations by random benchmarking *)
   FidSingleXY → <|0 → 0.9977,
     1 → 0.9987, 2 → 0.9996, 3 → 0.9988, 4 → 0.9991, 5 → 0.9989|>
   ,
   (* Error fraction/ratio {depolarising, dephasing} sum is either one or zero *)
   EFSingleXY → {0, 1}
```

```
,
(*  The rabi Frequency and fidelities of controlled-Z(C[Z]),
nearest-neighbors. Keys are the smallest qubit
  number. This applies to controlled-Ph gates *)
FreqCZ → <|0 → 12.1, 1 → 11.1, 2 → 6.6, 3 → 9.8, 4 → 5.4|>
,
(* Fidelity of controlled-Z; the numbers shown here are
 obtained from a simple optimisation via bell state fidelity *)
FidCZ → <|0 → 0.9374945614729504`, 1 → 0.9339691831083574`,
  2 → 0.9286379436705322`, 3 → 0.9967228426036524`, 4 → 0.9793017377403548`|>
,
(* Fidelity of CROT/Controlled-X rotation *)
FidCRot → 0.9988
,
(* Rabi frequency of CROT, obtained by conditional microwave drive *)
FreqCRot → 5
,
(* Error fraction/ratio {depolarising, dephasing} of controled-
 Ph(π) or controlled-Z. The error for other θ is scalled from π. *)
EFCZ → {0, 1}
,
(* Crosstalks error (C-Rz[ex])on the passive qubits when applying CZ gates;
square matrix with dims nqubit-2 *)
ExchangeRotOn → {{0, 0.023, 0.018 , 0.03, 0.04},
  {0.05 , 0, 0.03, 0.03, 0.04}, {0.05, 0.03 , 0, 0.07, 0.042},
  {0.038 , 0.03 , 0.031, 0 , 0.25}, {0.033, 0.03 , 0.02, 0.03, 0}}
,
(* Crosstalks error (C-Rz[ex])on the passive qubits when no CZ gates applied;
the qubits below indicate the controlled-qubit *)
ExchangeRotOff → <|0 → 0.039, 1 → 0.015 , 2 → 0.03 , 3 → 0.02, 4 → 0.028|>
,
(* Parity readout fidelity/charge readout fidelity between Q0,Q1 or Q5,Q6 *)
FidRead → 0.9997
,
(*Parity readout duration  *)
DurRead → 10
};
```

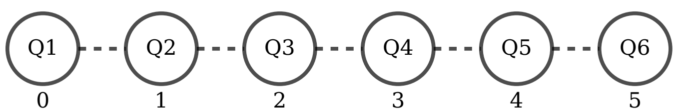
Elementary guide

Device connectivity

In[305]:=

```
With[{nq = OptionValue[SiliconDelft, QubitNum]},
 nodes =
  Labeled[#, Placed[{#, "Q" <> ToString[# + 1]}, {Below, Center}]] & /@ Range[0, nq - 1];
 Graph[nodes, Table[j ⟷ j + 1, {j, nodes[[All, 1]][[ ;; -2]]}],
  VertexSize → 0.6, VertexStyle → Directive[White, EdgeForm[Thick]],
  BaseStyle → {11, FontFamily → "Serif"}, ImageSize → Automatic,
  EdgeStyle → Directive[Black, Thick, Dashed]]
]
```

Out[ ]=



Native gates: $Rx_j[\theta]$, $Ry_j[\theta]$, $C_i[X_j]$, $C_i[Ph_j[\theta]]$, Read, Init, $Wait_i[\Delta t]$

Native gates

Initialisation must be done from edge qubits, for example:

$Init_{q1,q2}$, $Init_{q1,q2,q3}$, $Init_{q5,q6}$, $Init_{q4,q5,q6}$

Parity readout only on edge qubits, for example

$MeasP_{q1,q2}$, $MeasP_{q5,q6}$

Single-qubit gates

$Rx_q[\theta]$, $Ry_q[\theta]$, $Rz_q[\theta]$

Two-qubit gates

$C_p[Z_q]$, $C_p[Ph_q]$,

others: doing nothing

$Wait_q$[duration]

Basic operations

Doing nothing, observe the passive noise

The passive noise when no gates are applied.

The noise forms:

1) Cross-talk $C_i[Rz_{i+1}(\Delta t)]$ from input ExchangeRotOff, which is exchange rotation when no C[Z] gate is applied.
2) Standard dephasing from T2 input and depolarising from T1 inputs. We assume the T2* is echoed out to T2

The standard passivenoise (2) can be eliminated by setting **StdPassiveNoise → False**. The Cross-talk (1) can be set off by setting **ExchangeRotOff → False**

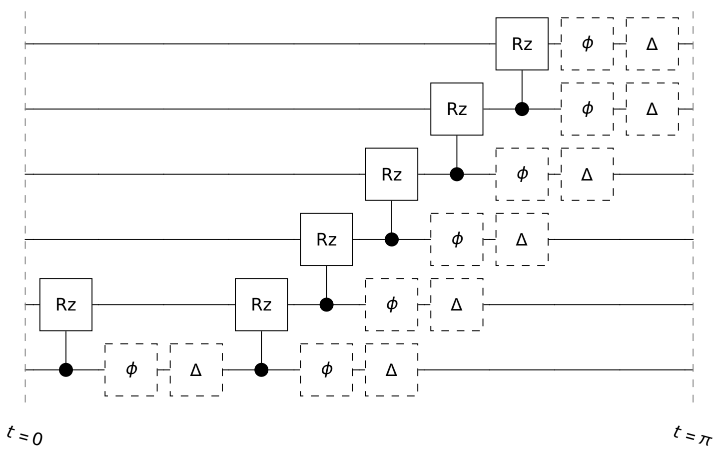
In[306]:=

```
InsertCircuitNoise[{Wait [π]}, SiliconDelft[]]
DrawCircuit[%]
```

Out[ ]=

```
{{0, {C [Rz [0.039]], Deph [0.100502], Depol [0.000235582]},
  {C [Rz [0.]], Deph [0.], Depol [0.], C [Rz [0.015]], Deph [0.0691683], Depol [0.000235582],
   C [Rz [0.03]], Deph [0.0376768], Depol [0.000235582], C [Rz [0.02]],
   Deph [0.0404918], Depol [0.000235582], C [Rz [0.028]], Deph [0.0339344],
   Depol [0.000235582], Deph [0.055502], Depol [0.000235582]}}, {π, {}, {}}}
```

Out[ ]=



Single qubit gates

Single qubit gates: x- and y- rotations have the same error
The noise forms:
1) Off-resonant Rabi Oscillation. This takes RabiFreq input and applied when **OffResonantRabi→ True**.
2) Standard Dephasing and Depolarising noise. This takes **FidSingleXY** and **EFSingleXY** inputs to estimate the error parameters. Set **EFSingleXY→{0,0}** to set this off.

In[308]:=

```
InsertCircuitNoise[CircSiliconDelft[{Rx0[π/2], Wait0[1]}, Parallel → False],
 SiliconDelft[StdPassiveNoise → True, OffResonantRabi → True]]
DrawCircuit[%]
```

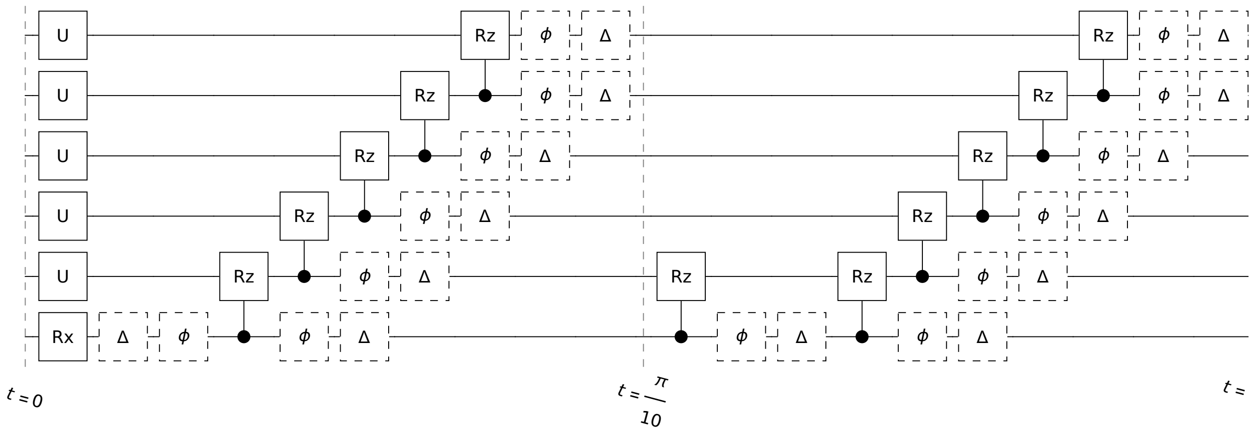
Out[ ]=

$$\left\{\left\{0, \left\{Rx_0\left[\frac{\pi}{2}\right], Depol_0[0.], Deph_0[0.001725], U_1\left[\left\{\left\{0.999287 - 0.0377401 i, -1.35127\times10^{-17} - 0.000725771 i\right\}, \left\{-1.35127\times10^{-17} - 0.000725771 i, 0.999287 + 0.0377401 i\right\}\right\}\right], U_2\left[\left\{\left\{0.999896 - 0.0144364 i, -1.45569\times10^{-18} - 0.00010615 i\right\}, \left\{-1.45569\times10^{-18} - 0.00010615 i, 0.999896 + 0.0144364 i\right\}\right\}\right], U_3\left[\left\{\left\{0.999791 - 0.02045 i, 3.05645\times10^{-16} - 0.000213021 i\right\}, \left\{3.05645\times10^{-16} - 0.000213021 i, 0.999791 + 0.02045 i\right\}\right\}\right], U_4\left[\left\{\left\{0.999385 - 0.0350469 i, -9.81184\times10^{-18} - 0.000625837 i\right\}, \left\{-9.81184\times10^{-18} - 0.000625837 i, 0.999385 + 0.0350469 i\right\}\right\}\right], U_5\left[\left\{\left\{0.98769 - 0.139614 i, 0.0705493 - 2.76517\times10^{-16} i\right\}, \left\{0.0705493 - 2.76517\times10^{-16} i, -0.98769 - 0.139614 i\right\}\right\}\right]\right\},$$

$$\{C_0[Rz_1[0.]], Deph_0[0.], Depol_0[0.], C_1[Rz_2[0.0015]], Deph_1[0.00738939], Depol_1[0.0000235616], C_2[Rz_3[0.003]], Deph_2[0.00390189], Depol_2[0.0000235616], C_3[Rz_4[0.002]], Deph_3[0.00420479], Depol_3[0.0000235616], C_4[Rz_5[0.0028]], Deph_4[0.00350177], Depol_4[0.0000235616], Deph_5[0.00584866], Depol_5[0.0000235616]\}\right\},$$

$$\left\{\frac{\pi}{10}, \{C_0[Rz_1[0.0124141]], Deph_0[0.0344686], Depol_0[0.0000749963]\}, \{C_0[Rz_1[0.]], Deph_0[0.], Depol_0[0.], C_1[Rz_2[0.00477465]], Deph_1[0.0231439], Depol_1[0.0000749963], C_2[Rz_3[0.0095493]], Deph_2[0.0123146], Depol_2[0.0000749963], C_3[Rz_4[0.0063662]], Deph_3[0.0132618], Depol_3[0.0000749963], C_4[Rz_5[0.00891268]], Deph_4[0.0110615], Depol_4[0.0000749963], Deph_5[0.0183802], Depol_5[0.0000749963]\}\right\}, \left\{1 + \frac{\pi}{10}, \{\}, \{\}\right\}\right\}$$

Out[•]=



Controlled - Z, Coltrolled-Phase

The noisy forms:

1) Standard 2-qubits dephasing and depolarising noise. This takes information **FidCZ** (fidelities) and **EFCZ** (error fraction). Set it off by **EFCZ→{0,0}**

2) Exchange rotation when a two-qubit gate is on. Takes information **ExchangeRotOn**. Set **ExchangeRotOn → False** to turn it off.

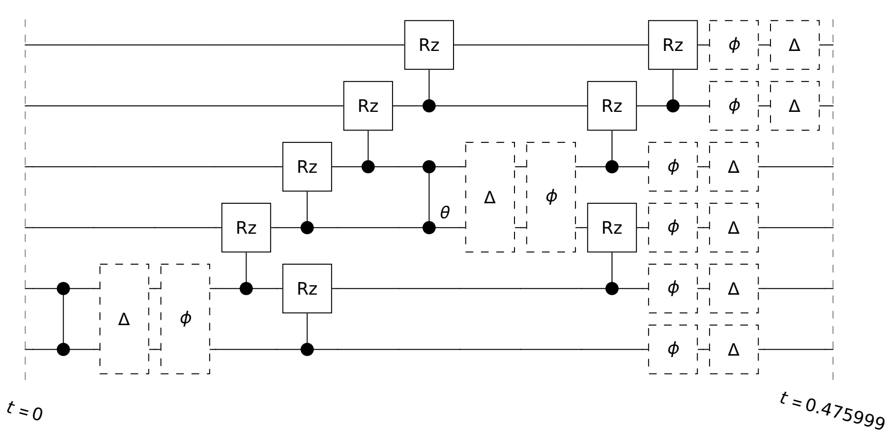
In[310]:=

```
InsertCircuitNoise[{C_0[Z_1], C_2[Ph_3[π]]},
 SiliconDelft[StdPassiveNoise → True, EFCZ → {0, 0}]]
DrawCircuit[%]
```

Out[•]=

{{0, {$C_0[Z_1]$, $Depol_{0,1}[0]$, $Deph_{0,1}[0]$, $C_1[Rz_2[0.023]]$, $C_2[Rz_3[0.018]]$, $C_3[Rz_4[0.03]]$, $C_4[Rz_5[0.04]]$, $C_2[Ph_3[\pi]]$, $Depol_{2,3}[0]$, $Deph_{2,3}[0]$, $C_0[Rz_1[0.05]]$, $C_1[Rz_2[0.03]]$, $C_3[Rz_4[0.07]]$, $C_4[Rz_5[0.042]]$},
{$Deph_0[0.00766785]$, $Depol_0[0.0000162271]$, $Deph_1[0.00510089]$, $Depol_1[0.0000162271]$, $Deph_2[0.]$, $Depol_2[0.]$, $Deph_3[0.]$, $Depol_3[0.]$, $Deph_4[0.00529612]$, $Depol_4[0.0000356991]$, $Deph_5[0.00883485]$, $Depol_5[0.0000356991]$}}, {0.475999, {}, {}}}

Out[•]=



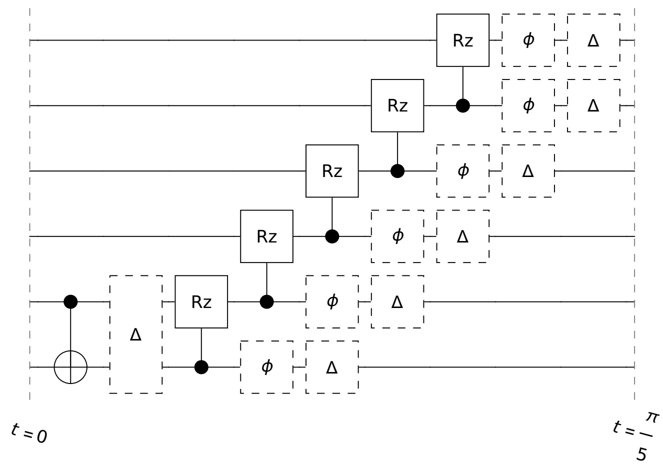
In[312]:=

```
InsertCircuitNoise[{CRot_{1,0}}, SiliconDelft[], ReplaceAliases → False]
DrawCircuit[%]
```

Out[ ]=

$\{\{0, \{C_1[X_0], Depol_{1,0}[0.0012]\}, \{C_0[Rz_1[0.]], Deph_0[0.], Depol_0[0.], C_1[Rz_2[0.]], Deph_1[0.], Depol_1[0.], C_2[Rz_3[0.006]], Deph_2[0.00777334], Depol_2[0.0000471224], C_3[Rz_4[0.004]], Deph_3[0.00837422], Depol_3[0.0000471224], C_4[Rz_5[0.0056]], Deph_4[0.00697901], Depol_4[0.0000471224], Deph_5[0.0116289], Depol_5[0.0000471224]\}\}, \{\frac{\pi}{5}, \{\}, \{\}\}\}$

Out[ ]=



Readout: parity measurement

Note: see *supplement/BellsonSiliconDelft/SiDelftReadInit.nb* for further details on this measurement model

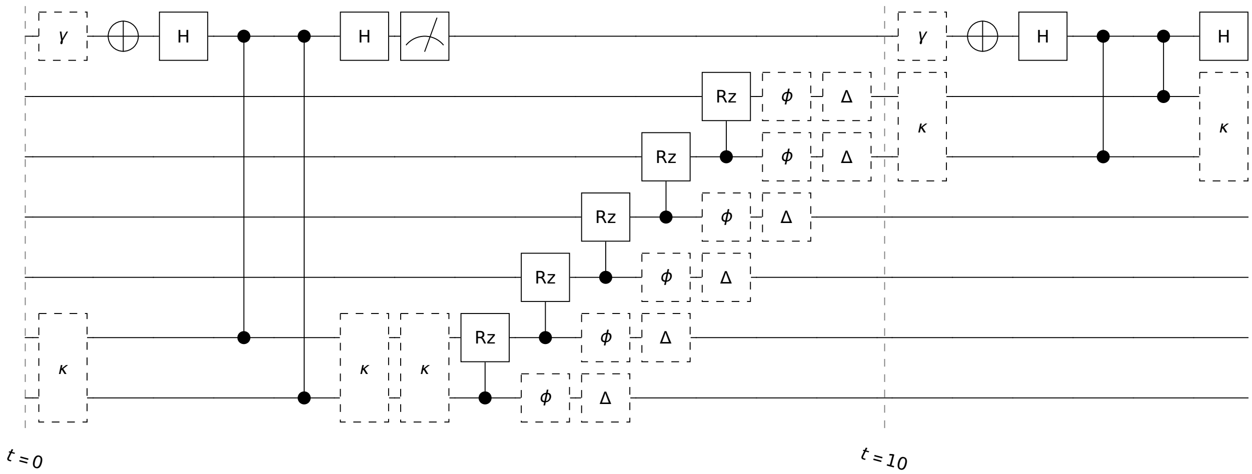
In[314]:=

```
InsertCircuitNoise[List /@ {MeasP_{1,0}, MeasP_{4,5}}, SiliconDelft[], ReplaceAliases → True];
```

In[315]:=

```
DrawCircuit@%
```

Out[ ]=



Reproducing results from the paper

Initialisation is obtained by 2 readouts and a partial swap with single qubit errors
The qubits are initialised to state 100 ... 001

Realtime feedback initialisation to |10⟩ and |100⟩

In[316]:=

```
ρ = CreateDensityQureg[7];
```

The initialisation process done in the experiment paper

In[317]:=

```
SetAttributes[readInit, HoldFirst]
readInit[ρ_, q0_, q1_, opt_ : {}] := Module[{m1, m2},
   m1 = First@Flatten@ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
          {MeasP_{q0,q1}}, SiliconDelft[Sequence @@ opt], ReplaceAliases → True]];
   If[m1 == 1, ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
        {Rx_{q0}[π]}, SiliconDelft[Sequence @@ opt], ReplaceAliases → True]]];
   m2 = First@Flatten@ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
          {MeasP_{q0,q1}}, SiliconDelft[Sequence @@ opt], ReplaceAliases → True]];
   {m1, m2}
 ]
```

Initialise the qubits to mixed state for a proper test

In[319]:=

```
SetQuregMatrix[ρ, RandomMixState[7]];
readInit[ρ, 0, 1]
Re@PartialTrace[ρ, 2, 3, 4, 5, 6][[2, 2]]
```

Out[ ]=

```
{1, 0}
```

Out[ ]=

```
0.9997
```

Readout (QND) on middle qubits Q2,Q3 and initialising it at the same time to state $|100\rangle$

Only works if the output the last measurement is 0; repeat the initialisation process otherwise

In[322]:=

```
readInit3[ρ_, q0_, q1_, q2_, opt_ : {}] := Module[{m1, m2, m3},
  m1 = First@Flatten@ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
        {MeasP_{q0,q1}}, SiliconDelft[Sequence @@ opt], ReplaceAliases → True]];
  If[m1 == 1, ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
      {Rx_{q0}[π]}, SiliconDelft[Sequence @@ opt], ReplaceAliases → True]]];
  m2 = First@Flatten@ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
        {MeasP_{q0,q1}}, SiliconDelft[Sequence @@ opt], ReplaceAliases → True]];
  m3 = First@Flatten@ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
        {CRot_{q2,q1}, MeasP_{q0,q1}}, SiliconDelft[Sequence @@ opt], ReplaceAliases → True]];
  If[m3 == 1, ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
      {Rx_{q2}[π]}, SiliconDelft[Sequence @@ opt], ReplaceAliases → True]]];
  {m1, m2, m3}
 ]
```

In[323]:=

```
(* initialise the qubits to mixed state *)
SetQuregMatrix[ρ, RandomMixState[7]];
```

In[324]:=

```
(* Keep doing it until the fidelity is high: Readout repetition in practice *)
```

In[325]:=

```
readInit3[ρ, 0, 1, 2]
Re@PartialTrace[ρ, 3, 4, 5, 6][[2, 2]]
```

Out[ ]=

```
{0, 0, 0}
```

Out[ ]=

```
0.998154
```

In[327]:=

```
readInit3[ρ, 5, 4, 3]
Re@PartialTrace[ρ, 0, 1, 2, 6][[5, 5]]
```

Out[ ]=

```
{1, 0, 1}
```

Out[ ]=

```
0.000137176
```

Full device initialisation to $|100\,001\rangle$

In[329]:=

```
ψ5 = CreateQureg[7];
ApplyCircuit[InitZeroState@ψ5, {X0, X5}];

(* initialise the qubits to mixed state *)
SetQuregMatrix[ρ, RandomMixState[7]];
(* repeat the measurement process: ideally all outputs are zeros *)
repeat = 4;

opt = {};
Table[readInit3[ρ, 5, 4, 3, opt], {repeat}]
Table[readInit3[ρ, 0, 1, 2, opt], {repeat}]
CalcFidelity[ρ, ψ5]
```

Out[ ]=

```
{{1, 0, 1}, {1, 0, 0}, {0, 0, 0}, {0, 0, 0}}
```

Out[ ]=

```
{{1, 0, 1}, {1, 0, 0}, {0, 0, 1}, {1, 0, 1}}
```

Out[ ]=

```
0.
```

Obtaining T1: X180-τ- with perfect measurement

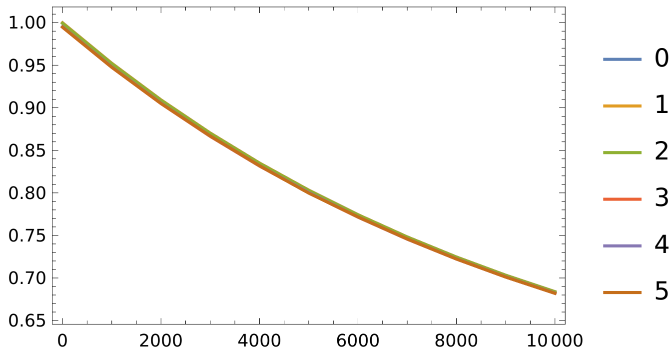
In[337]:=

```
RelaxationExperiment[dev_, qubit_, initrep_ : 3] := Module[{init},
  Table[
   (* initialise the qubits to mixed state *)
   SetQuregMatrix[ρ, RandomMixState[7]];
   Table[readInit3[ρ, 5, 4, 3], {initrep}];
   Table[readInit3[ρ, 0, 1, 2], {initrep}];
   ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
      If[MemberQ[{0, 5}, qubit], {Wait0[t]}, List/@{Rxqubit[π], Wait0[t]}], dev], dev]];
   {t, CalcProbOfOutcome[ρ, qubit, 1]}, {t, 0, 10^4, 1000}]
  ]
```

In[338]:=

```
ListPlot[RelaxationExperiment[SiliconDelft[], #] & /@ Range[0, 5],
 PlotLegends → Range[0, 5], Joined → True,
 ImageSize → 300, AxesLabel → {"τ", "p(0)"}, Frame → True]
```

Out[ ]=



Obtaining T_2 : X90 - tau - X180 - tau - X90, tau = 1 μs

In[95]:=

```
HahnEchoExperiment[dev_, qubit_, initrep_ : 3] := Module[{},
  Table[
   (* initialise the qubits to mixed state *)
   SetQuregMatrix[ρ, RandomMixState[7]];
   Table[readInit3[ρ, 5, 4, 3], {initrep}];
   Table[readInit3[ρ, 0, 1, 2], {initrep}];
   ApplyCircuit[ρ, ExtractCircuit@InsertCircuitNoise[
      List /@ {Rx_qubit[π / 2], Wait_0[t / 2], Rx_qubit[π], Wait_0[t / 2], Rx_qubit[π / 2]}, dev]];
   {t, CalcProbOfOutcome[ρ, qubit, If[MemberQ[{5, 0}, qubit], 1, 0]]}, {t, 0, 60, 4}]
 ]
```

In[96]:=

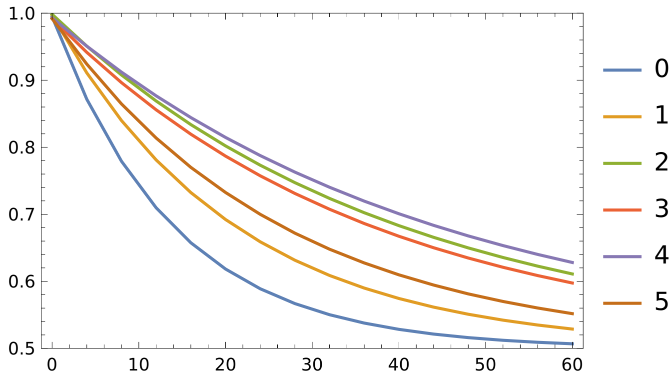
```
OptionValue[SiliconDelft, T2]
```

Out[ ]=

```
<|0 → 14, 1 → 21.1, 2 → 40.1, 3 → 37.2, 4 → 44.7, 5 → 26.7|>
```

In[97]:=
```
ListPlot[HahnEchoExperiment[SiliconDelft[], #] & /@ Range[0, 5],
 PlotLegends → Range[0, 5], PlotRange → {0.5, 1}, Joined → True,
 ImageSize → 300, AxesLabel → {"τ", "p(|+⟩)"}, Frame → True]
```

Out[ ]=



Paper supplement: Bell states

In[98]:=
```
chartstyle[label_] := {ImageSize → 200, BarSpacing → 0.1`,
   ColorFunction → Function[{height}, If[height < 0.1, ColorData["Rainbow"][10 height],
      ColorData["DeepSeaColors"][(height - 0.9) * 10]]], ChartElementFunction → "Cube",
   ChartStyle → EdgeForm[Thick], PlotTheme → "Business", Ticks →
    {{{1, "00"}, {2, "01"}, {3, "10"}, {4, "11"}}, {{1, "00"}, {2, "01"}, {3, "10"}, {4, "11"}},
     Automatic}, LabelStyle → Directive[Bold, Black],
   Epilog → Inset[Style[label, Thick, 17], ImageScaled[{.2, .7}]], PlotRange → All
  };
```

The CZ gates' fidelities are unknown. We set it according to the results on the Bell states fidelity.

In[99]:=
```
ψ2 = CreateQureg[2];
ρ2 = CreateDensityQureg[2];
```

In[101]:=
```
concurence[ρ_] := Module[{eigv, ρm, ρms, ρt, nq, pauliy},
  ρm = GetQuregMatrix@ρ;
  nq = IntegerPart@Log2@Length@ρm;
  ρms = MatrixPower[ρm, 1 / 2];
  pauliy = CalcCircuitMatrix[Y# & /@ Range[0, nq - 1]];
  ρt = pauliy.Conjugate[ρm].pauliy;
  eigv = Reverse@Sort[Chop@Eigenvalues[MatrixPower[ρms.ρt.ρms, 1 / 2]]];
  Max[0, eigv[[1]] - Total@eigv[[2 ;;]]]
 ]
```

In[102]:=

```
bellcircρ = <|
    "01" → {Rx0[π / 2], Rx1[-π / 2], C0[Z1], Rx1[π / 2]},
    "12" → {Rx1[π / 2], Rx2[π / 2], C1[Z2], Rx2[π / 2]},
    "23" → {Rx2[π / 2], Rx3[π / 2], C2[Z3], Rx3[π / 2]},
    "34" → {Rx3[π / 2], Rx4[π / 2], C3[Z4], Rx4[π / 2]},
    "45" → {Rx4[π / 2], Rx5[π / 2], C4[Z5], Rx5[π / 2]}|>;
bellcircψ = <|
    "01" → {X0, Rx0[π / 2], Rx1[-π / 2], C0[Z1], Rx1[π / 2]},
    "12" → {Rx0[π / 2], Rx1[π / 2], C0[Z1], Rx1[π / 2]},
    "23" → {Rx0[π / 2], Rx1[π / 2], C0[Z1], Rx1[π / 2]},
    "34" → {Rx0[π / 2], Rx1[π / 2], C0[Z1], Rx1[π / 2]},
    "45" → {X1, Rx0[π / 2], Rx1[π / 2], C0[Z1], Rx1[π / 2]}|>;
```

In[104]:=

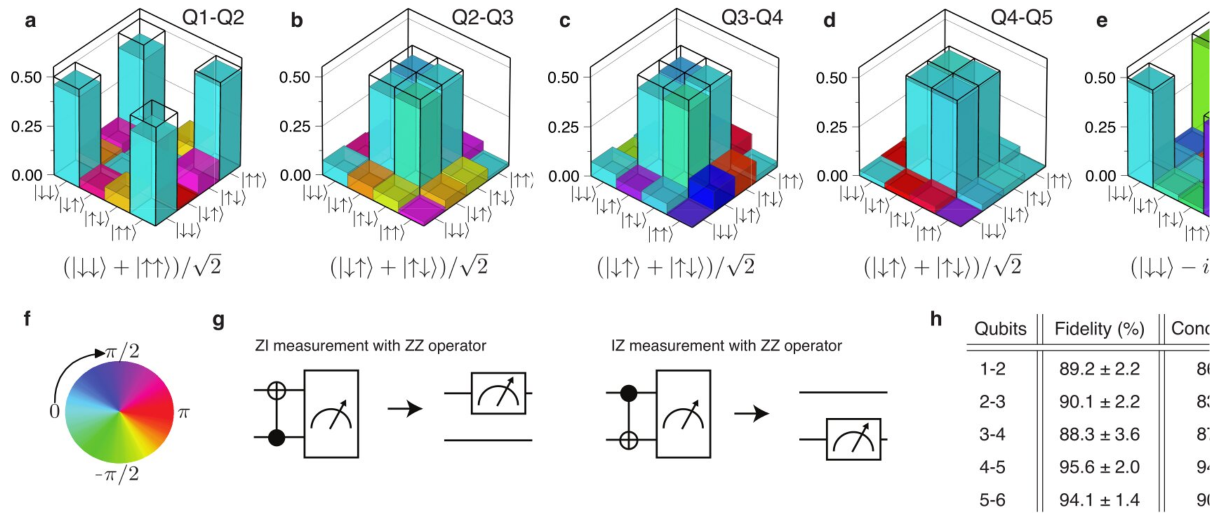
```
bell[code_, opt_, initrep_ : 4] := Module[{qubits, fid, plot, conc, str, out1, out2},
  qubits = ToExpression@StringSplit[code, ""];

  SetQuregMatrix[ρ, IdentityMatrix[2^7]];
  out1 = Table[readInit3[ρ, 5, 4, 3, opt], {initrep}];
  out2 = Table[readInit3[ρ, 0, 1, 2, opt], {initrep}];

  ApplyCircuit[ρ, ExtractCircuit@
    InsertCircuitNoise[Serialize@{bellcircρ[code]}, SiliconDelft[Sequence @@ opt]]];
  SetQuregMatrix[ρ2, PartialTrace[ρ, Sequence @@ Complement[Range[0, 5], qubits], 6]];
  conc = concurence[ρ2] * 100;
  ApplyCircuit[InitZeroState@ψ2, bellcircψ[code]];
  fid = CalcFidelity[ρ2, ψ2] * 100;
  str = "Q" <> ToString[1 + qubits[[1]]] <> "-Q" <> ToString[1 + qubits[[2]]];
  plot = PlotDensityMatrix[ρ2, ψ2, Sequence @@ chartstyle[str]];
  {str, plot, fid, conc}
 ]
```

Reference from the experiment

In[∘]:=



Qubits	Fidelity (%)	Conc
1-2	89.2 ± 2.2	8
2-3	90.1 ± 2.2	8
3-4	88.3 ± 3.6	8
4-5	95.6 ± 2.0	9
5-6	94.1 ± 1.4	9

The simulation

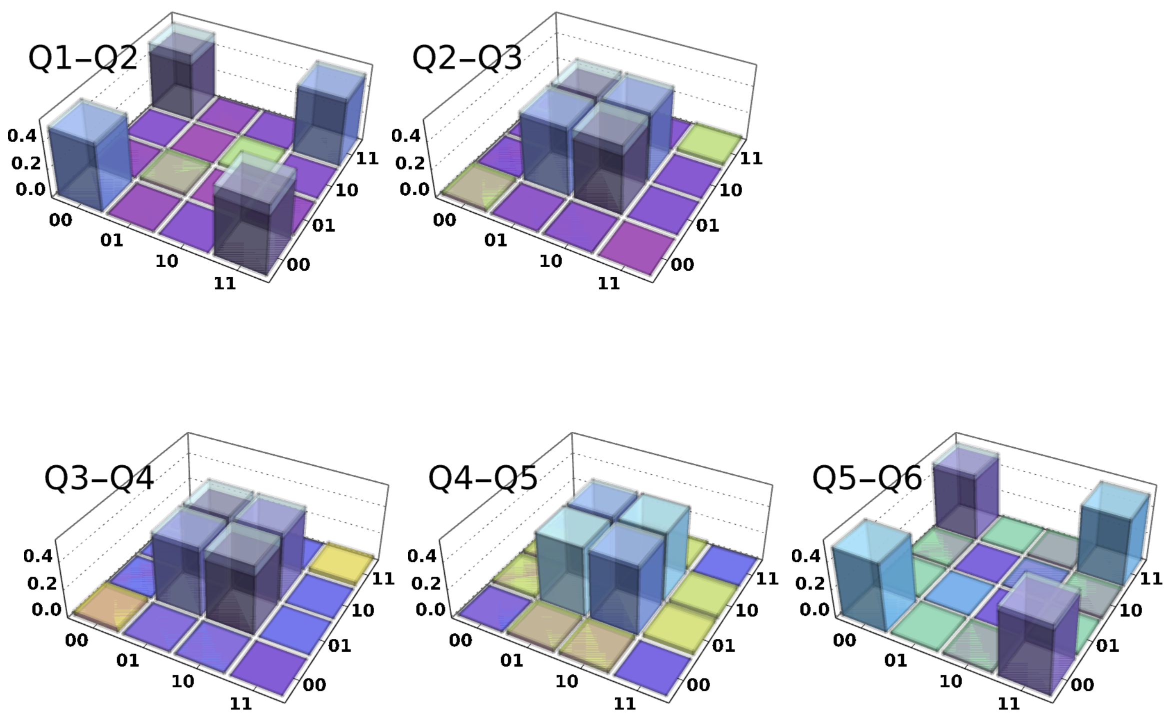
In[105]:=

```
plots = Transpose[bell[#, {}, 4] & /@
    ReplaceList[Sort@Range[0, 5], {p___, a_, b_, q___} :→ StringRiffle[{a, b}, ""]]];
TableForm[Transpose@{DecimalForm[#, 4] & /@ plots[[3]], DecimalForm[#, 4] & /@ plots[[4]]},
 TableHeadings → {plots[[1]], {"Fidelity", "Concurence"}}]
Row@plots[[2]]
```

Out[∘]//TableForm=

	Fidelity	Concurence
Q1-Q2	89.4	79.75
Q2-Q3	90.18	80.42
Q3-Q4	88.69	79.05
Q4-Q5	95.95	94.46
Q5-Q6	94.18	90.43

Out[∘]=



In[108]:=

```
(*Table[
  Export[plots[[1, i]] <> ".pdf", plots[[2, i]]],
  {i, Length@plots[[1]]}]*)
```

In[109]:=

```
(* average obtained fidelity *)
Round[#, 0.01] & /@ plots[[3]]
Mean@%
```

Out[ ]=

```
{89.4, 90.18, 88.69, 95.95, 94.18}
```

Out[ ]=

```
91.68
```