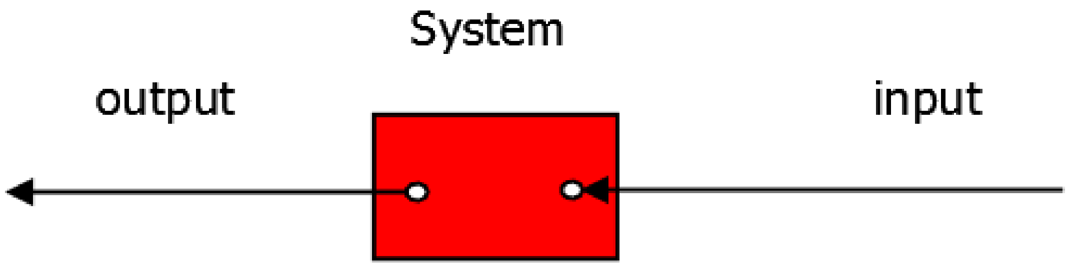
LECTURE 4
QUANTUM FEEDBACK NETWORKS

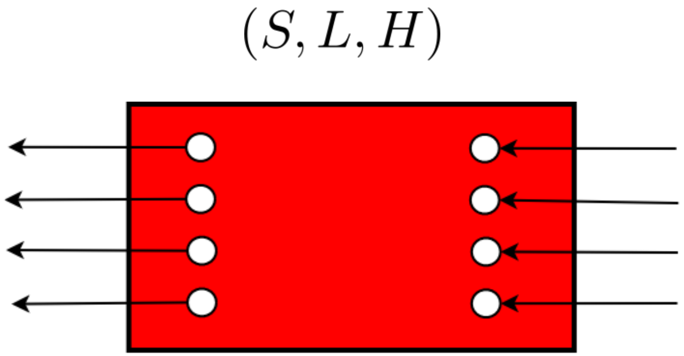
John Gough
Aberystwyth

CIRM, 16-20th April 2018

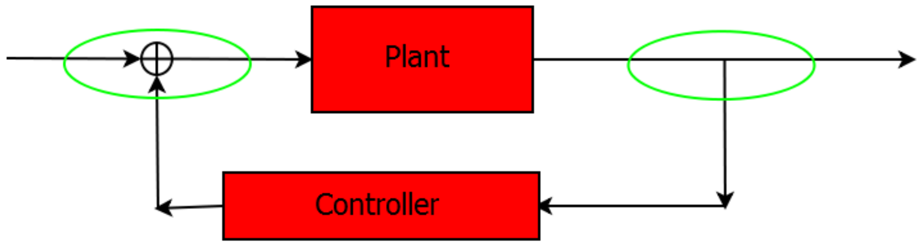
Quantum Markovian Models



The "wires" are quantum fields and may carry a multiplicity.



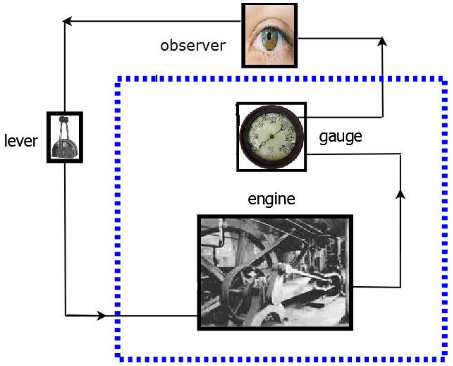
Networks and Feedback Control



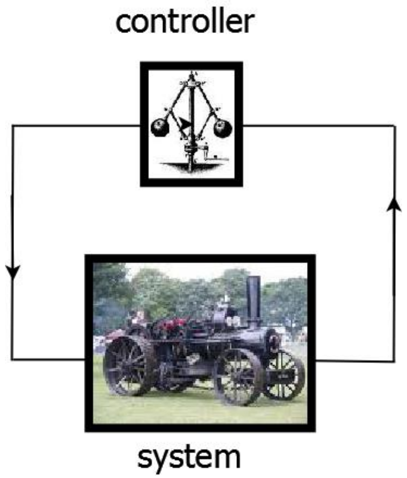
cannot happen in the quantum setting!!!

must use unitary junctions (e.g., beamsplitters)

- Measurement Based Feedback Control

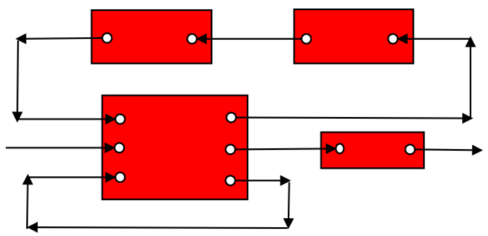


- Coherent Feedback Control

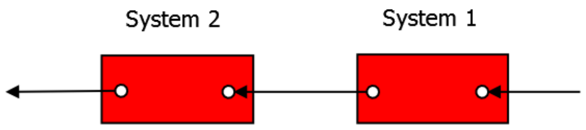


Quantum Networks

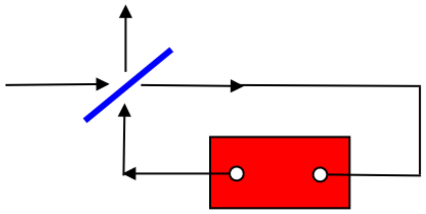
- How to connect models?



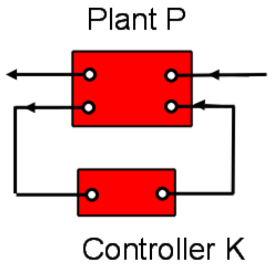
- Cascaded models



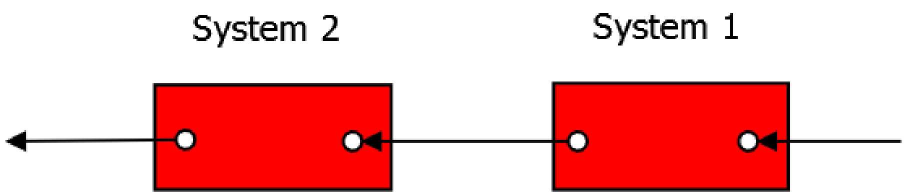
- Algebraic loops



- Feedback Control



The Series Product

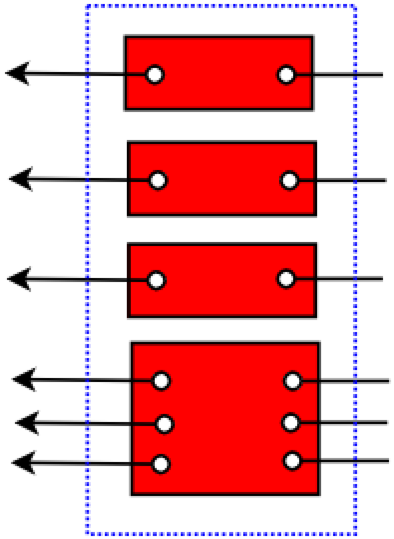


The cascaded system in the **instantaneous feedforward** limit is equivalent to the single component

$$(S_2, L_2, H_2) \triangleleft (S_1, L_1, H_1) = \left(S_2 S_1, L_2 + S_2 L_1, H_1 + H_2 + \mathrm{Im}\left\{ L_2^\dagger S_2 L_1 \right\} \right).$$

J. G., M.R. James, *The Series Product and Its Application to Quantum Feedforward and Feedback Networks* IEEE Transactions on Automatic Control, 2009.

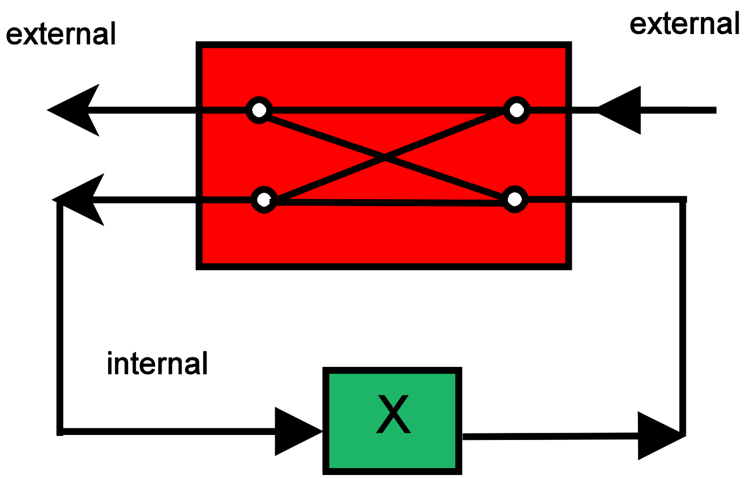
Network Rule # 1
Open loop systems in parallel



Models $(S_j, L_j, H_j)_{j=1}^{n}$ in parallel

$$\left(\begin{bmatrix} S_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & S_n \end{bmatrix}, \begin{bmatrix} L_1 \\ \vdots \\ L_n \end{bmatrix}, H_1 + \cdots + H_n \right).$$

Network Rule # 2
Feedback Reduction Formula



$$S = \begin{bmatrix} \mathsf{S}_{\mathtt{ii}} & \mathsf{S}_{\mathtt{ie}} \\ \mathsf{S}_{\mathtt{ei}} & \mathsf{S}_{\mathtt{ee}} \end{bmatrix}, \mathsf{L} = \begin{bmatrix} \mathsf{L}_{\mathtt{i}} \\ \mathsf{L}_{\mathtt{e}} \end{bmatrix}$$

The reduced model obtained by eliminating all the internal channels (instantaneous feedback) is determined by the operators $(\mathsf{S}^{\mathrm{fb}}, \mathsf{L}^{\mathrm{fb}}, \mathsf{H}^{\mathrm{fb}})$ given by

$$\begin{aligned} \mathsf{S}^{\mathrm{fb}} &= \mathsf{S}_{\mathtt{ee}} + \mathsf{S}_{\mathtt{ei}} X \left(1 - \mathsf{S}_{\mathtt{ii}} X\right)^{-1} \mathsf{S}_{\mathtt{ie}}, \\ \mathsf{L}^{\mathrm{fb}} &= \mathsf{L}_{\mathtt{e}} + \mathsf{S}_{\mathtt{ei}} X \left(1 - \mathsf{S}_{\mathtt{ii}} X\right)^{-1} \mathsf{L}_{\mathtt{i}}, \\ \mathsf{H}^{\mathrm{fb}} &= \mathsf{H} + \sum_{i=\mathtt{i},\mathtt{e}} \mathrm{Im} \mathsf{L}_j^{\dagger} X \mathsf{S}_{j\mathtt{i}} \left(1 - \mathsf{S}_{\mathtt{ii}} X\right)^{-1} \mathsf{L}_{\mathtt{i}}. \end{aligned}$$

J. G., M.R. James, *Quantum Feedback Networks: Hamiltonian Formulation*, Commun. Math. Phys., 1109-1132, Volume 287, Number 3 / May, 2009.

Properties of the Feedback Reduction Formula

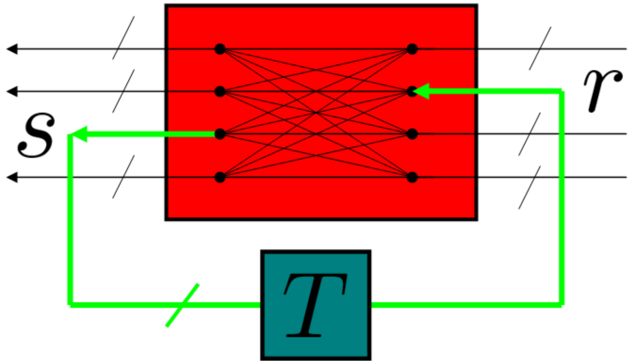
- Mathematically a Schur complement of the matrix of coefficient operators:

$$\mathbf{G} = \begin{bmatrix} -\frac{1}{2}L^*L - iH & -L^*S \\ L & S - I \end{bmatrix}.$$

- Equivalently formulated as a fractional linear transformation.
- Independent of the order of edge-elimination.

$$\mathsf{V} = \begin{bmatrix} -\frac{1}{2}L^*L - iH & -L^*S \\ L & S \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\sum_j L_j^* L_j - iH & -\sum_j L_j^* S_{j1} & \cdots & -\sum_j L_j^* S_{jm} \\ L_1 & S_{11} & \cdots & S_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ L_n & S_{n1} & \cdots & S_{nn} \end{bmatrix}$$

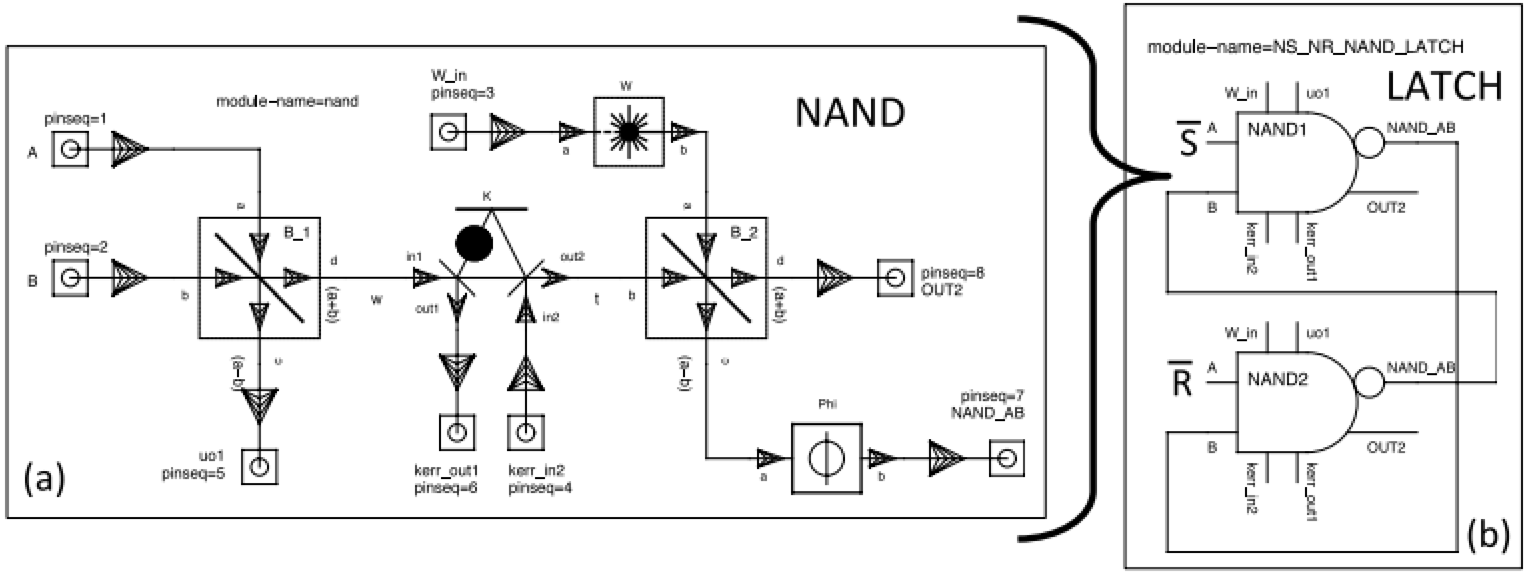
$$= \begin{bmatrix} \mathsf{V}_{00} & \mathsf{V}_{01} & \cdots & \mathsf{V}_{0m} \\ \mathsf{V}_{10} & \mathsf{V}_{11} & \cdots & \mathsf{V}_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{V}_{n0} & \mathsf{V}_{n1} & \cdots & \mathsf{V}_{nn} \end{bmatrix}.$$



The feedback reduction formula is

$$\left[\mathscr{F}_{(r,s)}(\mathsf{V}, T)\right]_{\alpha\beta} = \mathsf{V}_{\alpha\beta} - \mathsf{V}_{\alpha r} T \left(1 - \mathsf{V}_{rs} T\right)^{-1} \mathsf{V}_{s\beta}$$

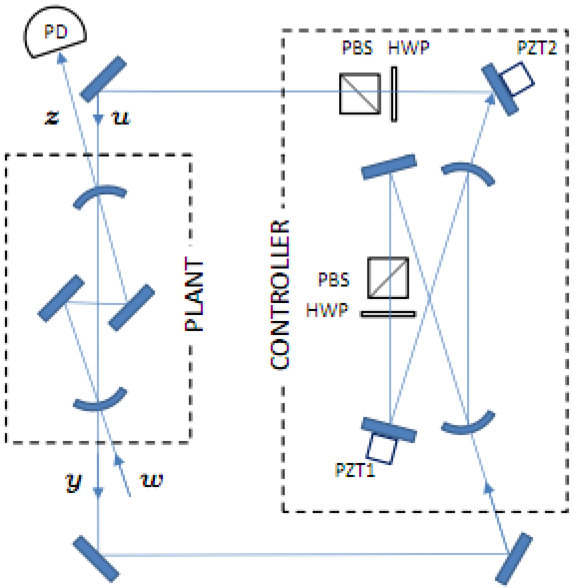
The Network Rules are implemented in a workflow capture package QHDL



QHDL (MabuchiLab)

N. Tezak, et al., (2012) Phil. Trans. Roy. Soc. A, 370, 5270.

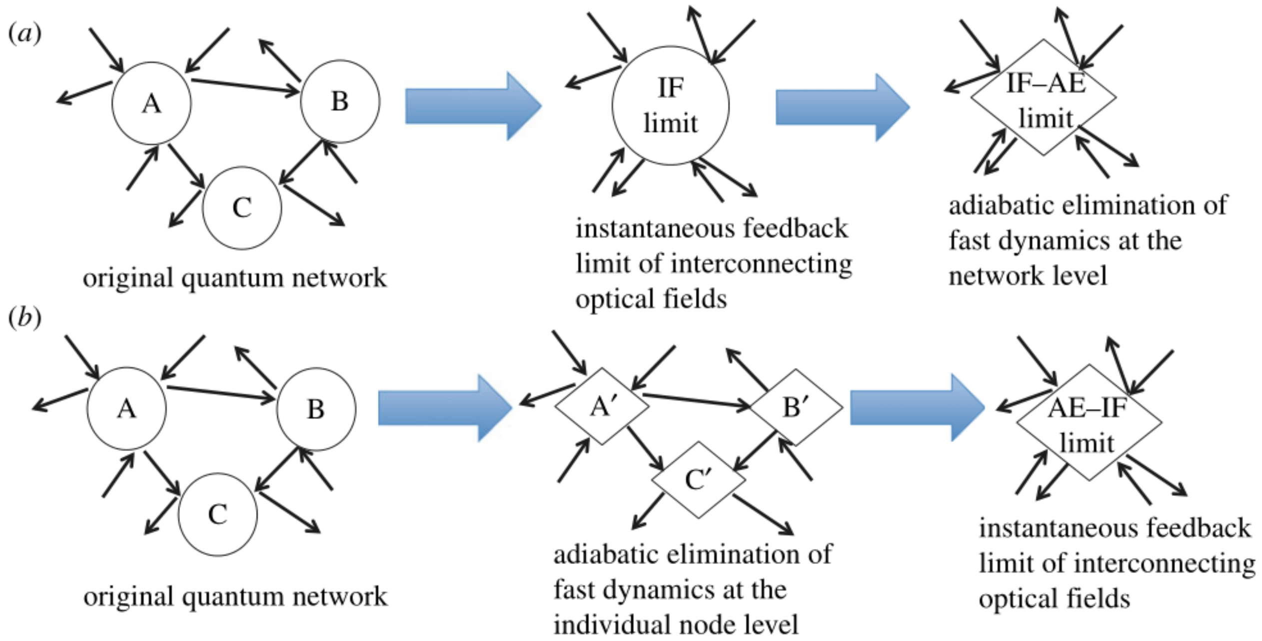
Coherent Quantum Feedback Control



H. Mabuchi, *Coherent-Feedback Quantum Control With a Dynamic Compensator*, Phys. Rev. A 78, 032323 (2008).

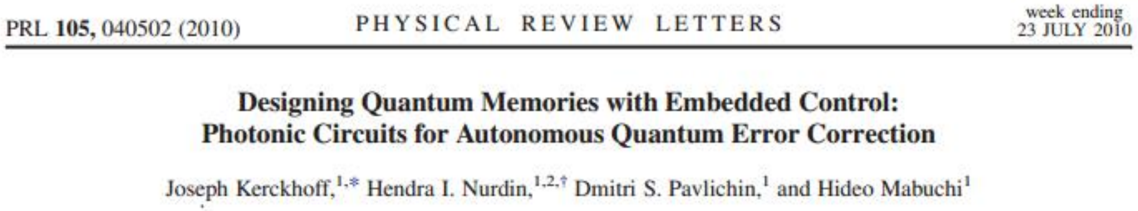
Adiabatic Elimination

- An important model simplification split the systems into slow and fast subspaces
- Mathematical this is also a Schur complement of the model matrix ***G***



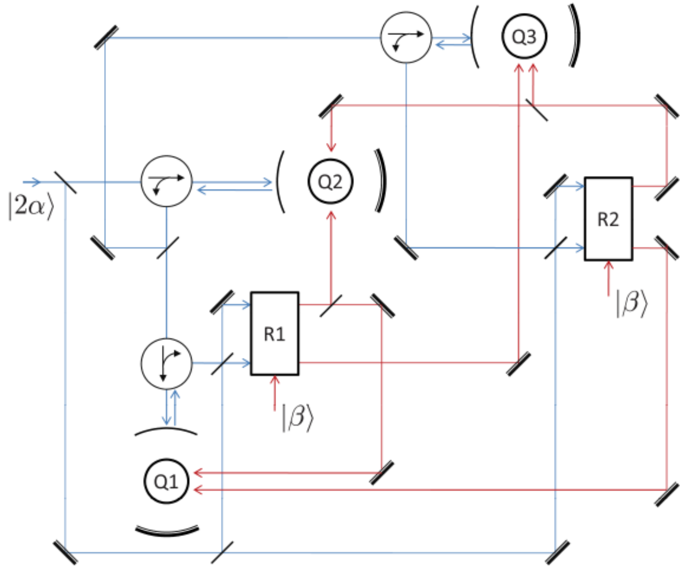
- It commutes with feedback reduction!

Autonomous Quantum Error Correction


PRL 105, 040502 (2010) PHYSICAL REVIEW LETTERS week ending 23 JULY 2010

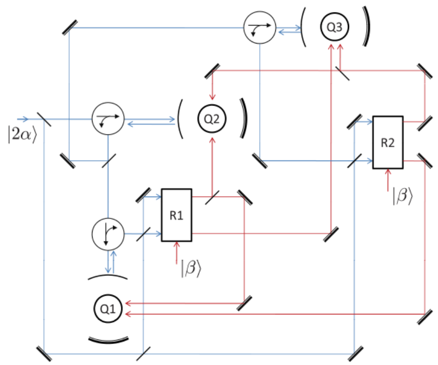
Designing Quantum Memories with Embedded Control: Photonic Circuits for Autonomous Quantum Error Correction

Joseph Kerckhoff,[1,*] Hendra I. Nurdin,[1,2,†] Dmitri S. Pavlichin,[1] and Hideo Mabuchi[1]



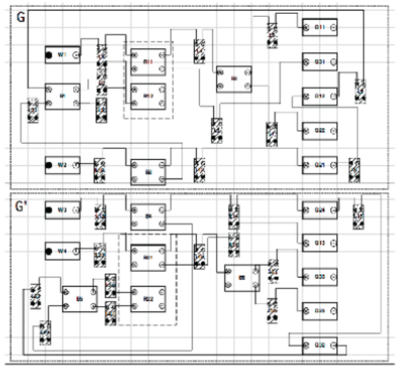
J. Kerckhoff, H. I. Nurdin, D. Pavlichin and H. Mabuchi, *Designing quantum memories with embedded control: photonic circuits for autonomous quantum error correction*, Phys. Rev. Lett. 105, 040502 (2010)

Network rules yield the overall "*SLH*"
From which we deduce the master equation



Set up in QHDL

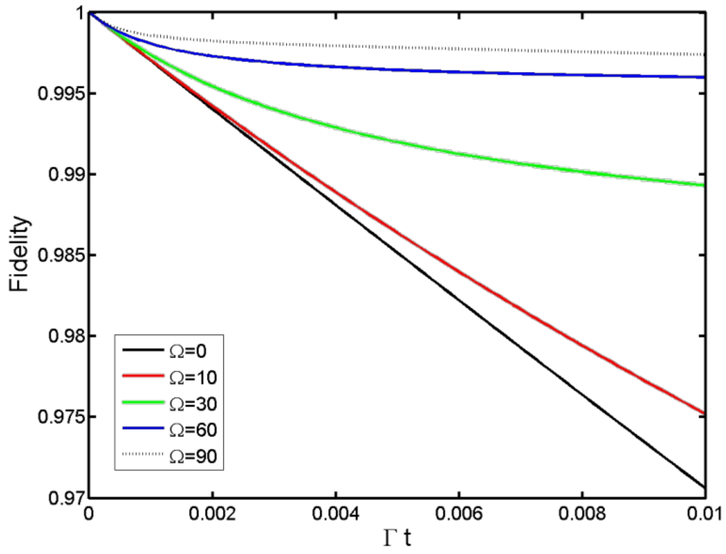
```
entity QECBitflip is
    generic (sqrt2Alpha, beta: complex);
    port    (pvac1, pvac2, pvac3, pvac4,
            fvac1, fvac2, fvac3, fvac4, fvac5, fvac6: in fieldmode;
            pvo1, pvo2, pvo3, pvo4,
            fvo1, fvo2, fvo3, fvo4, fvo5, fvo6: out fieldmode);
end QECBitflip;

architecture netlist of QECBitflip is

    -- Beamsplitter model
    component Beamsplitter
        port (In1, In2: in fieldmode;
            Out1, Out2: out fieldmode);
    end component;

    -- Z-probe cavity model to store the QBits
    component SingleSidedCavityZProbe
        port (ip1, if1, if2: in fieldmode;
            op1, of1, of2: out fieldmode);
    end component;

    -- Relay model to route feedback signals
    component Relay
        port ( if1, if2, ip1, ip2: in fieldmode;
                of1, of2, op1, op2: out fieldmode);
    end component;

    -- Coherent displacement / laser source
    component Displace
        generic (alpha: complex);
        port (VacIn: in fieldmode;
            Out1: out fieldmode);
    end component;

    -- signals for probe network
    signal pLaser, pbs1q2, pbs1bs2, pbs2bs3, pbs2bs4,
            pbs3r1_1, pbs3r1_2, pbs4r2_1, pbs4r2_2,
            pq1bs3, pq2bs5, pq3bs4 : fieldmode;

    -- signals for feedback network
    signal fLaser1, fLaser2, fr1q3, fr1q1q2, fr2q2q3, fr2q1,
            fbs1q1, fbs1q2, fbs2q2, fbs2q3,
            pbs5q1, pbs5q3: fieldmode;

begin
    -- probe source
    pSOURCE: Displace
                generic map (alpha => sqrt2Alpha);
                port map(pvac1, pLaser);

    -- Beamsplitters in Probe network
    pBS1: Beamsplitter
        port map(pvac2, pLaser, pbs1q2, pbs1bs2);

    pBS2: Beamsplitter
        port map(pbs1bs2, pvac3, pbs2bs3, pbs2bs4);

    pBS3: Beamsplitter
        port map(pq1bs3, pbs2bs3, pbs3r1_1, pbs3r1_2);

    pBS4: Beamsplitter
        port map(pq3bs4, pbs2bs4, pbs4r2_1, pbs4r2_2);
```




```
model LossyBitFlip
  /*The first half of a QHDL/M model consists of the component declaration.
  *We take advantage of Modelica's object-oriented syntax with components inheriting
  *properties from their respective classes. Component parameters are specified as
  *arguments, for example, a cavity type, coherent field amplitude, or Hilbert space */

  /*One-port cavity declarations*/
  Photonics.BlockComponents.SingleCavity Q11(CavityType=Ramang, HilbertSpace=Q1)
  Photonics.BlockComponents.SingleCavity Q12(CavityType=Zprobe, HilbertSpace=Q1)
  Photonics.BlockComponents.SingleCavity Q13(CavityType=Ramanh, HilbertSpace=Q1)
  Photonics.BlockComponents.SingleCavity Q21(CavityType=Zprobe, HilbertSpace=Q2)
  Photonics.BlockComponents.SingleCavity Q22(CavityType=Ramanh, HilbertSpace=Q2)
  Photonics.BlockComponents.SingleCavity Q23(CavityType=Ramang, HilbertSpace=Q2)
  Photonics.BlockComponents.SingleCavity Q24(CavityType=Zprobe, HilbertSpace=Q2)
  Photonics.BlockComponents.SingleCavity Q31(CavityType=Ramang, HilbertSpace=Q3)
  Photonics.BlockComponents.SingleCavity Q32(CavityType=Zprobe, HilbertSpace=Q3)
  Photonics.BlockComponents.SingleCavity Q33(CavityType=Ramanh, HilbertSpace=Q3)

  /*Coherent field declarations*/
  Photonics.BlockComponents.CoherentField W2(Amplitude=sqrt2alpha)
  Photonics.BlockComponents.CoherentField W1(Amplitude=beta)
  Photonics.BlockComponents.CoherentField W3(Amplitude=sqrt2alpha)
  Photonics.BlockComponents.CoherentField W4(Amplitude=beta)

  /*Cavity-QED relay declarations*/
  Photonics.BlockComponents.RelayPi R11(HilbertSpace=R1)
  Photonics.BlockComponents.RelaySigma R12(HilbertSpace=R1)
  Photonics.BlockComponents.RelayPi R21(HilbertSpace=R2)
  Photonics.BlockComponents.RelaySigma R22(HilbertSpace=R2)

  /*Beamsplitter declarations*/
  Photonics.BlockComponents.BeamSplitter B1
```

$$\dot{\rho}_t = -i[H,\rho_t] + \sum_{i=1}^{7}\left(L_i\rho_t L_i^* - \frac{1}{2}\{L_i^*L_i,\rho_t\}\right)$$

$$H = \sqrt{2}\Omega\Pi_g^{(R_1)}\Pi_h^{(R_2)}X_1 + \sqrt{2}\Omega\Pi_h^{(R_1)}\Pi_g^{(R_2)}X_3 - \Omega\Pi_g^{(R_1)}\Pi_g^{(R_2)}X_2$$



$$L_1 = \frac{\alpha}{\sqrt{2}}\{\sigma_{hg}^{(R_1)}(1+Z_1Z_2) + \Pi_h^{(R_1)}(1-Z_1Z_2)\}$$

$$L_2 = \frac{\alpha}{\sqrt{2}}\{\sigma_{gh}^{(R_1)}(1-Z_1Z_2) + \Pi_g^{(R_1)}(1+Z_1Z_2)\}$$

$$L_3 = \frac{\alpha}{\sqrt{2}}\{\sigma_{hg}^{(R_2)}(1+Z_3Z_2) + \Pi_h^{(R_2)}(1-Z_3Z_2)\}$$

$$L_4 = \frac{\alpha}{\sqrt{2}}\{\sigma_{gh}^{(R_2)}(1-Z_3Z_2) + \Pi_g^{(R_2)}(1+Z_3Z_2)\}$$