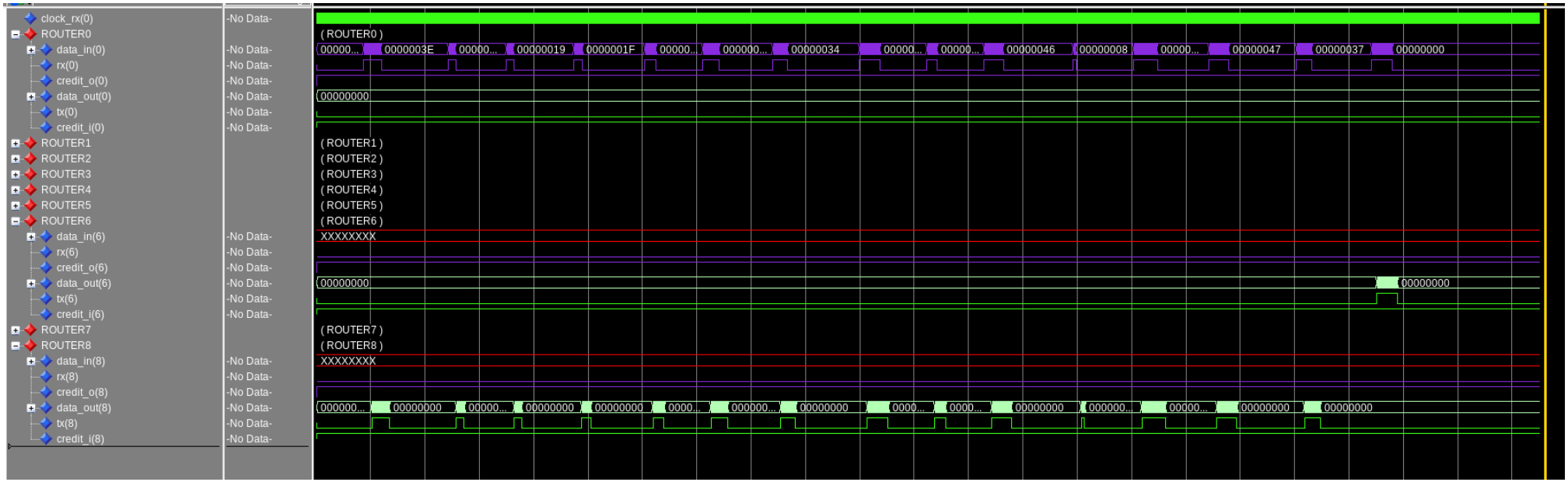
Pessoal

Na paxos (ou outra máquina qualquer):

- wget https://www.inf.pucrs.br/moraes/NOC_DESIGNS_SISCHIP.tar.gz
- tar -xzvf NOC_DESIGNS_SISCHIP.tar.gz
- ir na pasta: noc_generic_abr_22_le_arquivo
- vsim -do sim.do

"abrir" os roteadores 0 (0,0) , 6 (0,2) e 8 (2,2). O R0 envia 14 pacotes para o roteador R8 e 1 para o roteador R6.



O arquivo: topNoC.vhd:

- possui uma NoC X_ROUTERSxY_ROUTERS (noc1: Entity work.NOC)
- um port map que lê um arquivo com tráfego injetado no roteador 0
- um process que gera um arquivo de pacotes recebidos

Willam: substituir o arquivo *topNoC.vhd* por um *test bench* em SystemVerilog que leia um arquivo de especificação de tráfego por roteador.

Lembrando o sistema de coordenadas:

(0,2) (1,2) (2,2)

(0,1) (1,1) (2,1)

(0,0) (1,0) (2,0)

Prever um arquivo texto para cada roteador (se não houver arquivo não tem problema, pois o *test bench* deixa os sinais de tx zerados):

Cada pacote é definido para uma tupla {ti, target x, target y, size}

Exemplo hipotético para r0.txt (entrada para o test bench):

20 2 2 8
80 1 2 10

Neste caso, se a NoC for de 32 bits, o test bench irá gerar para o roteador 0, a partir do tempo 20, se houver crédito:

0000 0202 # destino
0000 **0008** # size
0000 **0014** # o primeiro flit do payload deve ser o memento de injeção, para cálculo de latência – nesta caso $(14)_{16}$ que é $(20)_{10}$
xxxx **xxxx** # 2º flit contador global do gerado de tráfego com o número do pacote
0000 **0003** # 3º flit
0000 **0004** # 4º flit
0000 **0005** # 5º flit
0000 **0006** # 6º flit
0000 **0007** # 7º flit
0000 **0008** # 8º flit

E a partir do tempo 80, se houver crédito:

0000 0102 # destino
0000 **000A** # size
0000 **0051** # o primeiro flit do payload deve ser o memento de injeção, para cálculo de latência – nesta caso $(50)_{16}$ que é $(80)_{10}$
xxxx **xxxx** # 2º flit contador global do gerado de tráfego com o número do pacote
0000 **0003** # 3º flit
0000 **0004** # 4º flit
0000 **0005** # 5º flit
0000 **0006** # 6º flit

0000 **0007** # 7º flit
0000 **0008** # 8º flit
0000 **0009** # 9º flit
0000 **000A** # 10º flit

- Gerar um arquivo de log por roteador, que gere para cada roteador **quantos** pacotes foram recebidos, e para cada pacote:

<source> <size> <latência>

- O *source* é a parte alta do primeiro flit (em vermelho acima), o size o segundo flit, e a latência é a diferença entre o tempo atual e o tempo do terceiro flit.

- Instrumentalizar o buffer.vhd para logar o endereço do roteador quando passar o segundo flit de payload, e logar esta informação em um arquivo texto único – isto permitirá traçar os caminhos tomados pelos pacotes

Elisa – Geração de tráfego para validar a NoC. Gerar no formato aceito pelo William (pode ser em Phyton, C, ou qq outra linguagem).

Voltando ao exemplo hipotético para R0 {ti, target x, target y, size}
20 2 2 8
80 1 2 10

Observar que temos:
(1) distribuição temporal, isto é, **quando** os pacotes devem ser injetados (*ti* – tempo inicial, primeiro campo)
(2) distribuição espacial, isto é, para **onde** os pacotes devem ser enviados (*target* – destino do pacote)
(3) **volume** de dados, que é o tamanho dos pacotes (*size*)

➔ A distribuição **temporal** pode ser (neste trabalho): uniforme ou normal (Gaussiana)

Distribuição temporal Uniforme. Exemplo:

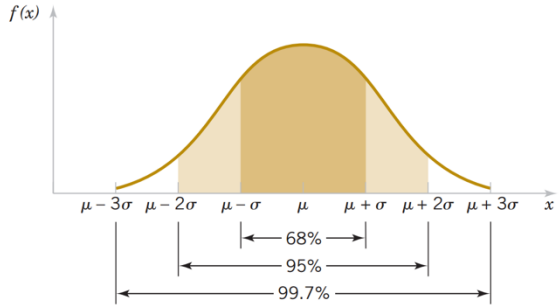
- Frequência: 1GHz
- Largura do flit: 32 bits
- Pacote de tamanho 64 (2 de cabeçalho e 62 de payload)

Temos uma banda por link de **32 Gbps**

Se quisermos uma taxa uniforme de **1 Gbps** ($\frac{1}{32}$ $da\ banda\ total$) temos que transmitir um pacote de 64 flits a cada 2048 ciclos de clock.

Distribuição temporal Normal (Gaussiana). Exemplo:

- Frequência: 1GHz
- Largura do flit: 32 bits
- Pacote de tamanho 64 (2 de cabeçalho e 62 de *payload*)
- **Taxa média 9 Gbps, desvio padrão 2 Gbps e 100 pacotes.**



Deve gerar uma tabela com número de pacotes no pontos: [μ-3σ], [μ-2σ], [μ-σ], [μ], [μ+σ], [μ+2σ] e [μ+3σ]. Depois se injeta na rede aleatoriamente os pacotes usando a tabela.

Exemplo (mais ou menos para μ=9 e σ=2):

Taxa (Gbps)	Número de pacotes
μ-3σ	2
μ-2σ	6
μ-1σ	23
μ	37
μ+1σ	23
μ+2σ	6
μ+3σ	2
Total de pacotes:	**100**

➔ **Distribuição espacial:**

- **Uniforme**: todos os roteadores têm a mesma probabilidade de receber pacotes

  Em uma rede 4x4, há 16 roteadores. Para 100 pacotes, cada roteador pode receber entre 6 e 7 pacotes, até fechar 100 pacotes

- **N-hot-spot**: N roteadores recebem o tráfego. Por exemplo, se N=2 (dois roteadores hot spot), cada roteador envia 50% do tráfego para estes dois roteadores.

Sugestão de entrada:

```
.global  T E S N

.temp <média> {<desv pad>}

.hot <N> <coord 1> <cood 2> <coord n>

.R[x,y]
    .D T S
   . temp <média>  {<desv pad>}
.R[x,y]
   .D T S
  . temp <média>  {<desv pad>}
.....
```

Onde

.global – configuração para todos os roteadores

T: pode ser U ou N
E: pode ser U ou H
S: tamanho do pacote
N: número de pacotes por roteador

.temp– configuração da distribuição temporal, média e desvio padrão para a distribuição normal

.hot – parâmetros para a distribuição N-hot-spot

.R[x,y] D T S – sobrepõe-se ao global, com tráfego específico para um dado roteador, onde D é o destino do pacote, T é a distribuição temporal (U ou N) e S tamanho do pacote. Logo depois do .target deve vir um . temp, que especifica a taxa constante ou normal.

Eu começaria por um:

```
.R[0,0]
    .[3,3] U 64
    .temp 2
```

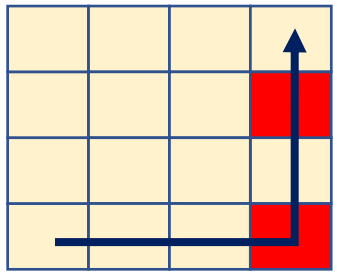
- Significa roteador [0,0] enviando 64 pacotes a uma taxa de 2 Gbps para o roteador [3,3] (estou supondo uma NoC 4x4). Este exemplo de entrada deve gerar para o *test bench* do William o arquivo r0.txt:

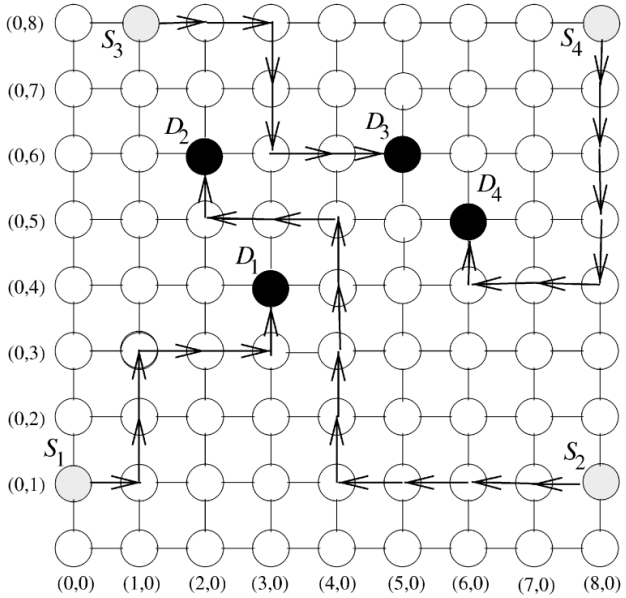
0	3	3	64
2048	3	3	64
4096	3	3	64
6144	3	3	64
8192	3	3	64
10240	3	3	64
12288	3	3	64
14336	3	3	64
16384	3	3	64
18432	3	3	64
20480	3	3	64
22528	3	3	64

Gustavo – roteamento adaptativo, para diminuir a latência e sair de *hotspots*. Um tráfego XY pode ser perturbado para um *hotspot* como abaixo.

```
.global  U H 64 300
.temp 6
.hot 2 [3,0]  [3,2]

.R[0,0]
    .[3,3] U 200
    .temp 6
```





- Todos os roteadores, **exceto** o [0,0], geram 150 pacotes para o [3,0] e 150 pacotes para o [3,2], com taxa uniforme de 6 Gbps, com pacotes tendo tamanho 64 flits.

- Sugestão de algoritmo de roteamento: ODD-EVEN: https://www.inf.pucrs.br/moraes/sichip/2noc/odd-even.pdf (código C na página 4, e abaixo). Basicamente o algoritmo calcula 2 opções para onde um pacote deve ir, e decide pela opção que tem porta livre – linha antes do *end*.

- Obs:
    - Precisa colocar o endereço *source* no pacote para testar, como em vermelho para o William
    - Vai escrever código (pouco) no **switch control** – certamente vai aumentar algum(ns) estado(s)

- Abaixo comentários sobre o código fonte.

- Teste – usa o **noc_generic_abr_22_le_arquivo** que é simples, e gera bloqueios como no artigo forçando alguns sinais de crédito para zero (página 5).

```
Algorithm ROUTE
/* Source node: (s_0, s_1); destination node: (d_0, d_1); current node: (c_0, c_1). */
begin
  Avail_Dimension_Set ⟵ ∅;
  c_0 ← d_0 − c_0;
  c_1 ← d_1 − c_1;
  if (e_0 = 0 and e_1 = 0)
     Deliver the packet to the local node and exit;
  if (e_0 = 0)      /* currently in the same column as destination */
      if (e_1 > 0)
          Add North to Avail_Dimension_Set;
      else
          Add South to Avail_Dimension_Set;
  else
      if (e_0 > 0)      /* eastbound messages */
          if (e_1 = 0)
              Add East to Avail_Dimension_Set;
          else {
                if (c_0 is odd or c_0 = s_0)
                    if (e_1 > 0)
                        Add North to Avail_Dimension_Set;
                    else
                        Add South to Avail_Dimension_Set;
                if (d_0 is odd or e_0 ≠ 1)    /* odd destination column or ≥ 2 columns to destination */
                    Add East to Avail_Dimension_Set;
               }
      else {     /* westbound messages */
              Add West to Avail_Dimension_Set;
              if (c_0 is even)
                  if (e_1 > 0)
                      Add North to Avail_Dimension_Set;
                  else
                      Add South to Avail_Dimension_Set;
            }
  Select a dimension from Avail_Dimension_Set to forward the packet;
end
```

Duas dimensões possíveis de saída

Delta x e delta y (como na Hermes)

Chegou na porta local (como na Hermes)

Alinhou na vertical, só pode subir ou descer

Para a direita

Neste else é o odd-even funcionando, vai para a leste (direita) e nas colunas ímpares permite subir ou descer (cuidar exceções)

Para a esquerda

Neste outro else é o odd-even funcionando, vai para oeste, e nas colunas pares permite subir ou descer