Sincronizador multi-domínio para uma rede-em-chip hierárquica

Jonathan Martinelli, Cezar Reinbrecht, Débora Matos, Altamiro Susin

- Devido ao número crescente de elementos de processamento que tem sido inserido nos atuais MPSoCs, se faz extremamente necessário que cada dispositivo permita a utilização de um determinado domínio de *clock*. A nossa proposta difere das demais uma vez que as interfaces de rede para sincronismo de *clocks* foram desenvolvidas para uma topologia de rede hierárquica chamada HiCIT.

- Entre os métodos verificados na literatura, o que nos mostrou ser mais eficaz foi a proposta de um sincronizador que propõe a implementação de uma FIFO bi-síncrona para multi-domínios de *clock*.

Arquitetura Proposta

✓HiCIT é composta por clusters baseados em crossbars no nível local e no nível global utiliza roteadores.

✓No nível global é usado uma topologia da rede mesh com um mecanismo de *wormhole*. Os roteadores da *mesh* possuem cinco portas, quatro para conexão com roteadores vizinhos e um para conexão com o cluster baseado em *crossbar*.

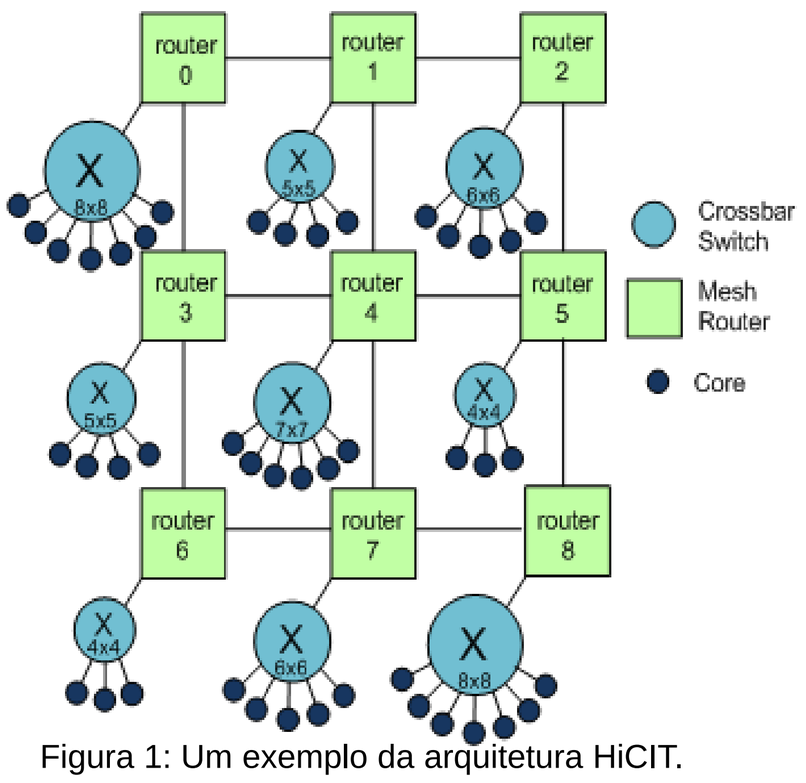

Figura 1: Um exemplo da arquitetura HiCIT.

- A arquitetura do sincronizador implementado consiste em cinco módulos:
  - **Buffer:** composto por registradores de dados, mulitiplexadores e portas AND.
  - **Write e Read pointers:** são ponteiros que apontam para a próxima posição de escrita ou leitura, respectivamente do buffer. Utilizam o algoritmo Bubble Enconding para impedir a ocorrência de metaestabilidade.
  - **Full e Empty Detector:** sinaliza a atual situação do buffer, e indica os estados de cheio ou vazio. Primeiramente realizam a sincronização dos clocks de escrita e leitura e logo após, a comparação entre ponteiros.

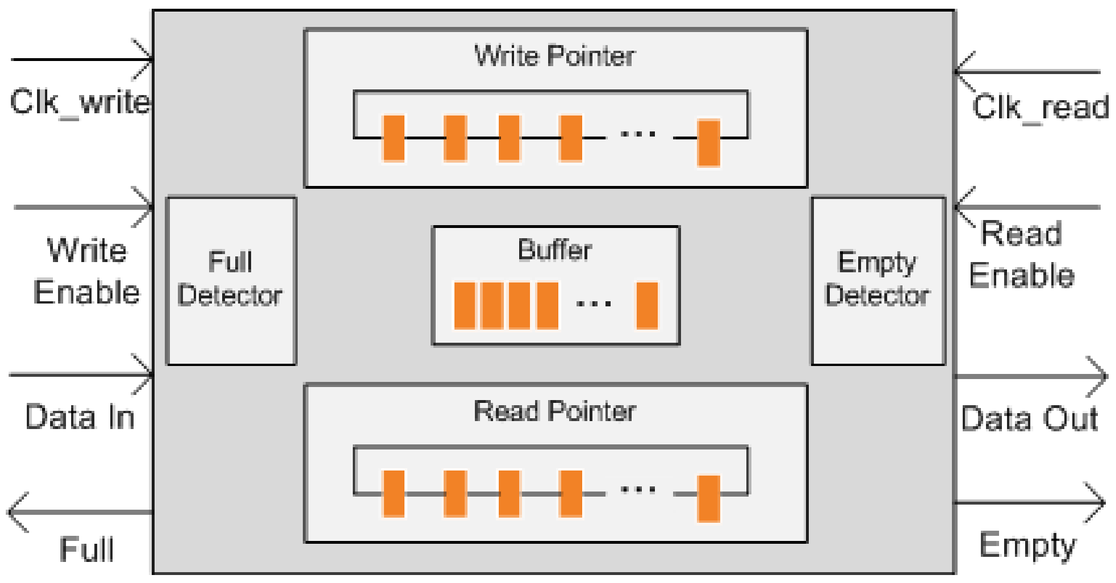

Figura 2: Módulo sincronizador Multi-domínio de *clocks*.

Integração

- A ideia da integração é utilizar o sincronizador entre o *crossbar* e o roteador ao qual o *crossbar* esteja conectado. Assim há a possibilidade de utilização de um *clock* diferente para cada *crossbar* e um *clock* único para a *mesh*.

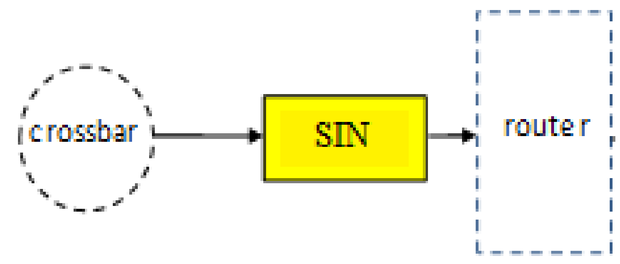

Figura 3: Integração crossbar com roteador.

- Também há a possibilidade de utilizar o sincronizador entre roteadores. Nesse caso, é útil utilizar um sincronizador entre roteadores quando se faz o uso de uma rede muito grande, já que podem ocorrer problemas de *clock skew* e *jitter*.

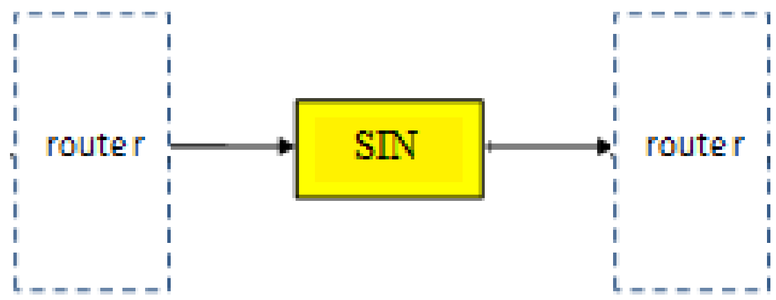

Figura 4: Integração roteador com roteador.

Resultados

- Como o transmissor e o receptor possuem diferentes *clocks*, a latência da nossa FIFO dependerá da relação entre esses 2 sinais. Essa latência será a diferença de um período de *clock* do receptor após constar que o dado foi escrito.

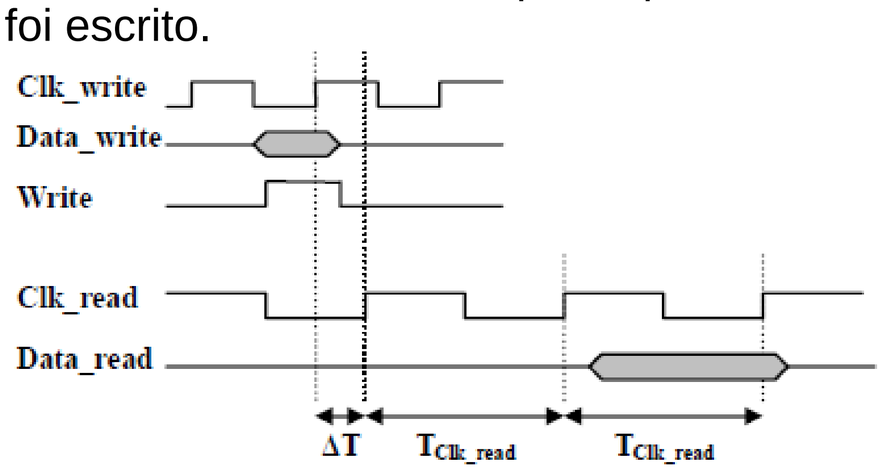

Figura 5: Análise de latência.

Conclusão

- Com o sincronizador é possível que cada região opere na sua frequência ideal. Sendo assim, algumas regiões podem operar com valores mais baixos e assim permite-se **reduzir a potência do sistema sem comprometer o desempenho do mesmo**.
- O sincronizador pode ser utilizado para interfacear diferentes regiões que precisem operar em domínios de *clock* e/ou fases diferentes, além de solucionar problemas de distribuição de clock, como *skew* e *jitter*.