Fast Fourier Transforms in FPGAs

Fourier Transformation

- Continuous

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\omega) \cdot e^{j(\omega t)} \cdot d\omega$$

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) \cdot e^{j(-\omega t)} \cdot dt$$

- Discrete

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} X(\omega) \cdot e^{j(\omega T_s n)} \cdot d(\omega T_s)$$

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) \cdot e^{j(-\omega T_s n)}$$

Discrete Fourier Transform (DFT)

- Windowed
  - (N time-points in: N freq-points out at fs/N)

$$N\delta = \omega_s \quad X(k\delta) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\delta T_s kn\right)}$$

$$X_N(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi kn}{N}\right)}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_N(k) \cdot e^{j\left(\frac{2\pi kn}{N}\right)}$$

Fast Fourier Transform (FFT)

$$\forall_{k=0}^{N-1} \qquad X_N(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi kn}{N}\right)}$$

- DFT
  - Requires N^2 multiplications
- (Fast Fourier Transform) FFT
  - uses repetitive nature of twiddle-factor
  - (N/2)log(N) multiplications

Fast Fourier Transform (FFT)

$$X_N(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\Pi kn}{N}\right)}$$

$$W_N = e^{-j\left(\frac{2\Pi}{N}\right)}$$

$$X_N(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn}$$

- W_N^k is called the twiddle factor

(FFT) Decimation in time

- Decimate time samples in odd and even groups
- Partition frequency samples into bottom and top halves

$$X_N(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn}$$

$$X_N(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) \cdot W_N^{2rk} + \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) \cdot W_N^{(2r+1)k}$$

$$X_N(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) \cdot \left(W_N^2\right)^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) \cdot \left(W_N^2\right)^{rk}$$

Decimation in time (contd.)

$$W_N = e^{j\left(-\frac{2\Pi}{N}\right)}$$

$$W_N^2 = e^{j\left(-\frac{2\Pi\times 2}{N}\right)} = e^{j\left(-\frac{2\Pi}{N/2}\right)} = W_{N/2}$$

$$X_N(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r)\cdot W_{N/2}^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x(2r+1)\cdot W_{N/2}^{rk}$$

$$X_N(k) = G_{N/2}(k) + W_N^k H_{N/2}(k)$$

Decimation in time (contd.)

- Decimate time samples in odd and even groups
- Partition frequency samples into bottom and top halves

$$X_N(k) = G_{N/2}(k) + W_N^k H_{N/2}(k)$$

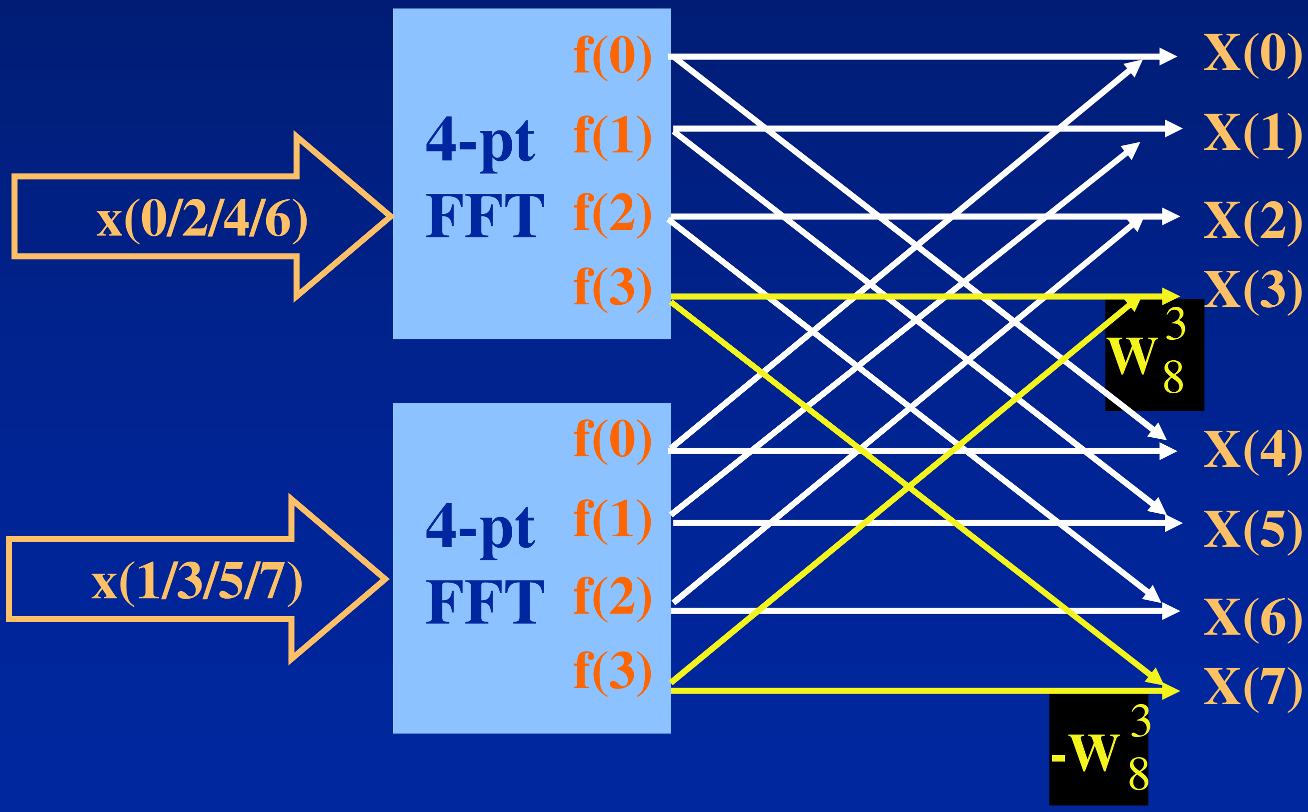
$$X_N\left(k + \frac{N}{2}\right) = G_{N/2}(k) + W_N^k W_N^{N/2} H_{N/2}(k)$$

$$X_N\left(k + \frac{N}{2}\right) = G_{N/2}(k) - W_N^k H_{N/2}(k)$$

Example: 8-point FFT

$$X_N(k) = G_{N/2}(k) + W_N^k H_{N/2}(k)$$

$$X_N\left(k + \frac{N}{2}\right) = G_{N/2}(k) - W_N^k H_{N/2}(k)$$



XILINX

4-point FFT (Decimation in time)

{0, 1, 2, 3} => {0, 2} {1, 3}

$$X_4(k) = \sum_{n=0}^{3} x(n) \cdot W_4^{kn}$$

$$= \sum_{r=0}^{1} x(2r) \cdot W_2^{rk} + W_4^k \sum_{r=0}^{1} x(2r+1) \cdot W_2^{rk}$$

$$= [x(0) + x(2) \cdot W_2^k] + W_4^k [x(1) + x(3) \cdot W_2^k]$$

$$= [x(0) + x(2) \cdot W_4^{2k}] + W_4^k [x(1) + x(3) \cdot W_4^{2k}]$$

4-point FFT (contd.)

- Expanded form

$$X_4(0) = [x(0) + x(2) \cdot W_4^0] + W_4^0[x(1) + x(3) \cdot W_4^0]$$

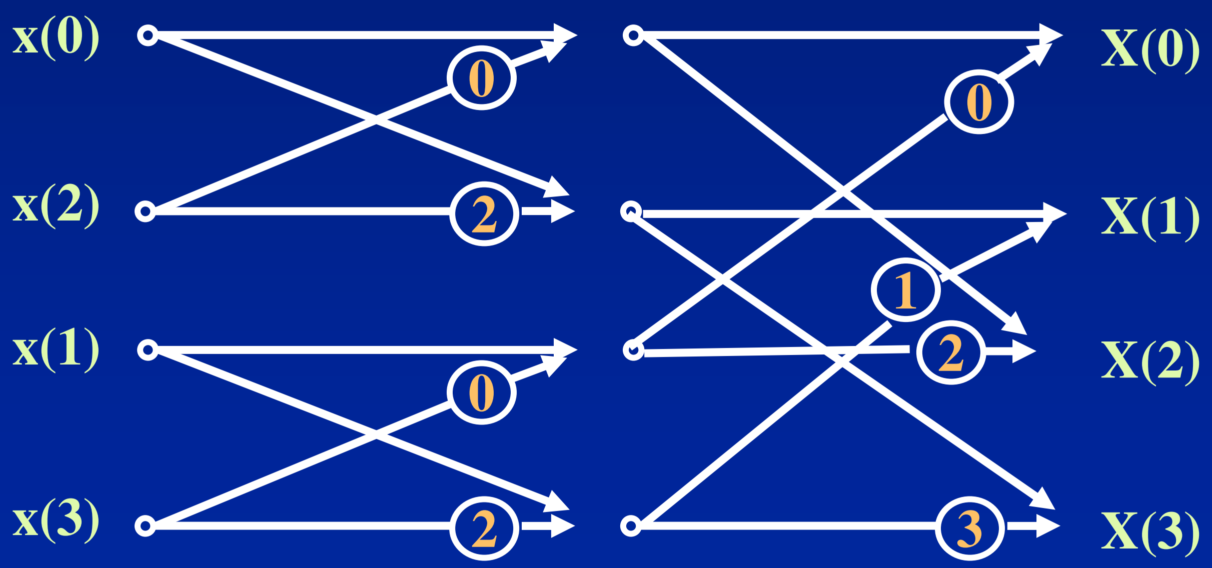
$$X_4(1) = [x(0) + x(2) \cdot W_4^2] + W_4^1[x(1) + x(3) \cdot W_4^2]$$

$$X_4(2) = [x(0) + x(2) \cdot W_4^0] + W_4^2[x(1) + x(3) \cdot W_4^0]$$

$$X_4(3) = [x(0) + x(2) \cdot W_4^2] + W_4^3[x(1) + x(3) \cdot W_4^2]$$

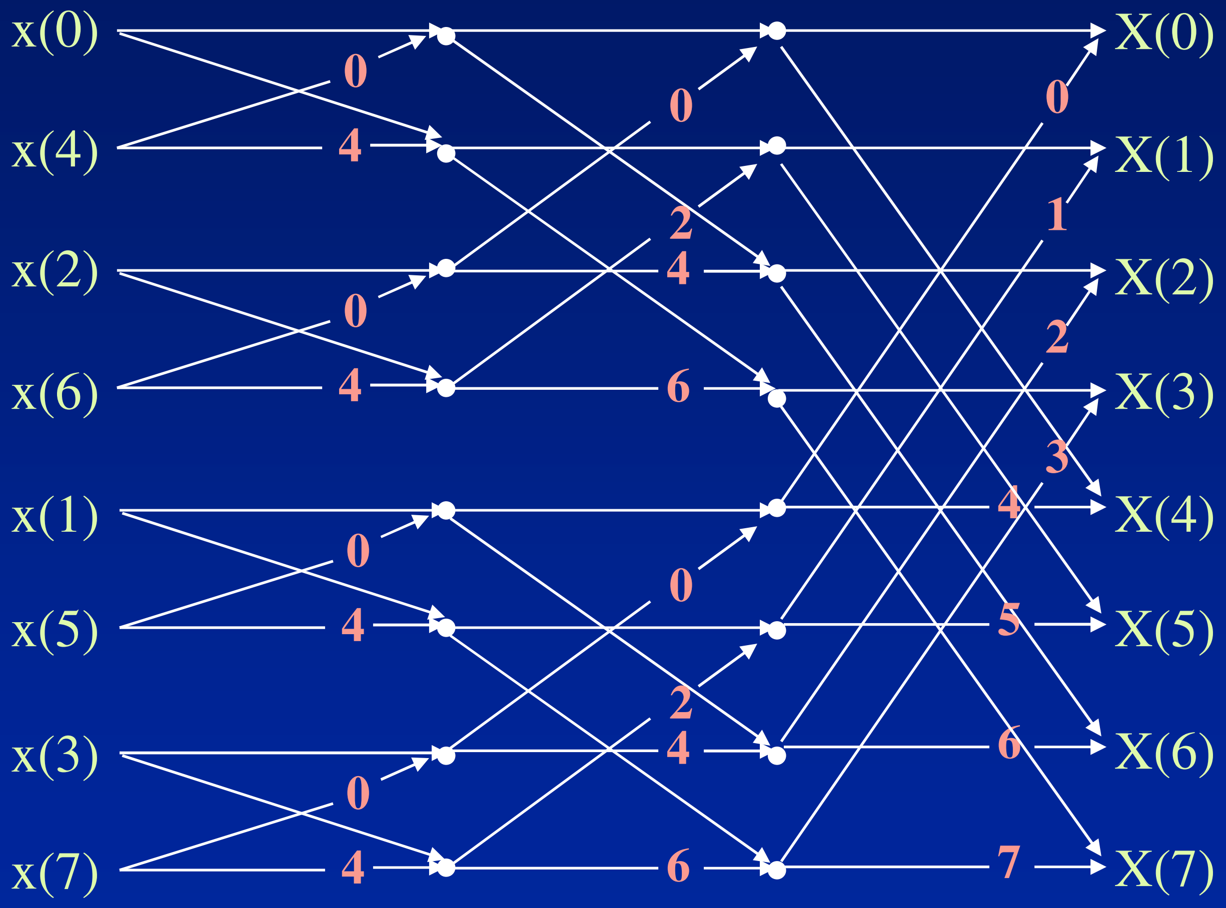
4-point FFT (contd.)

- Pictorial form for complete 4-point FFT
- Radix-2 structures form the core



$2 \Rightarrow W_4^2$

FFT structure



- 8-point FFT
- input bit-reversed
- 6 => W_8^6 $= e^{-j2\pi * 6/8}$

(FFT) Decimation in frequency

- Decimate frequency samples in odd and even groups
- Partition time samples into bottom and top halves

$$X_N(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{kn}$$

$$X_N(k) = \sum_{n=0}^{\frac{N}{2}-1} x(n) \cdot W_N^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n + \frac{N}{2}\right) \cdot W_N^{k(n+\frac{N}{2})}$$

Even Freq components

$$X_N(2v) = \sum_{n=0}^{\frac{N}{2}-1} x(n)\cdot W_N^{2vn} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right)\cdot W_N^{2v(n+\frac{N}{2})}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)\cdot W_{N/2}^{vn} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right)\cdot W_{N/2}^{vn}\cdot W_N^{vN}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) + x\left(n+\frac{N}{2}\right)\right)\cdot W_{N/2}^{vn}$$

$$= G_{N/2}(v)$$

Odd Freq Components

$$X_N(2v+1) = \sum_{n=0}^{\frac{N}{2}-1} x(n)\cdot W_N^{(2v+1)n} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right)\cdot W_N^{(2v+1)(n+\frac{N}{2})}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x(n)\cdot W_N^{n}\cdot W_{N/2}^{vn} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\frac{N}{2}\right)\cdot W_N^{n}\cdot W_{N/2}^{vn}\cdot W_N^{vN}\cdot W_N^{N/2}$$

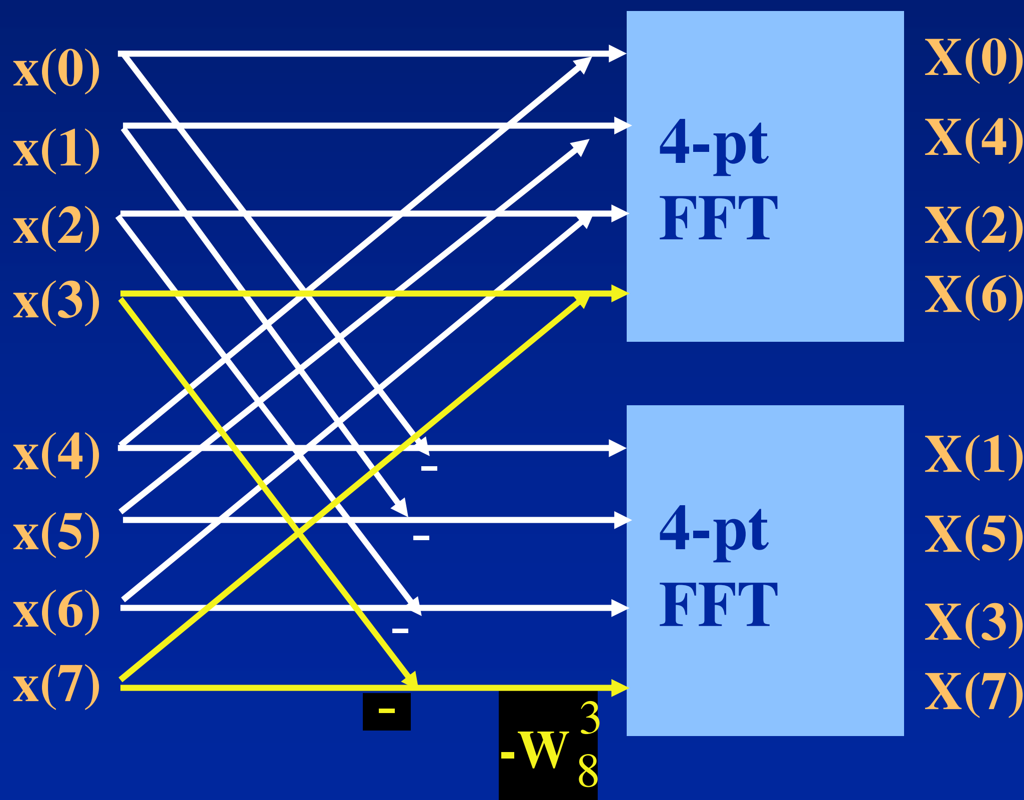
$$= \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) - x\left(n+\frac{N}{2}\right)\right)\cdot W_N^{n}\cdot W_{N/2}^{vn}$$

$$= H_{N/2}(v)$$

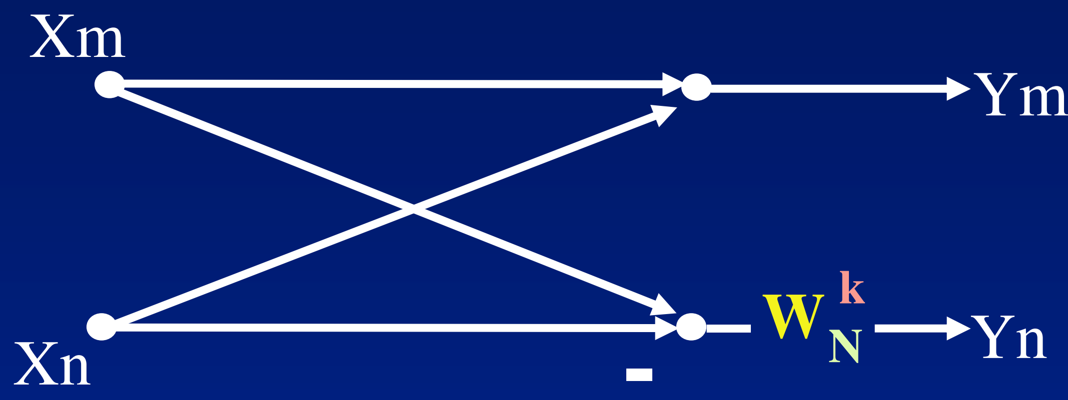
Ex: 8-point FFT

$$X_N(2v) = \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) + x\left(n+\frac{N}{2}\right)\right) \cdot W_{N/2}^{vn}$$

$$X_N(2v+1) = \sum_{n=0}^{\frac{N}{2}-1} \left(x(n) - x\left(n+\frac{N}{2}\right)\right) \cdot W_N^{n} \cdot W_{N/2}^{vn}$$



Radix-2 (decimation in freq)



$$y_m = x_m + x_n = x_{Rm} + x_{Rn} + j\left(x_{\mathrm{Im}} + x_{In}\right)$$

$$y_n = \left(x_m - x_n\right) \times W_N^k$$

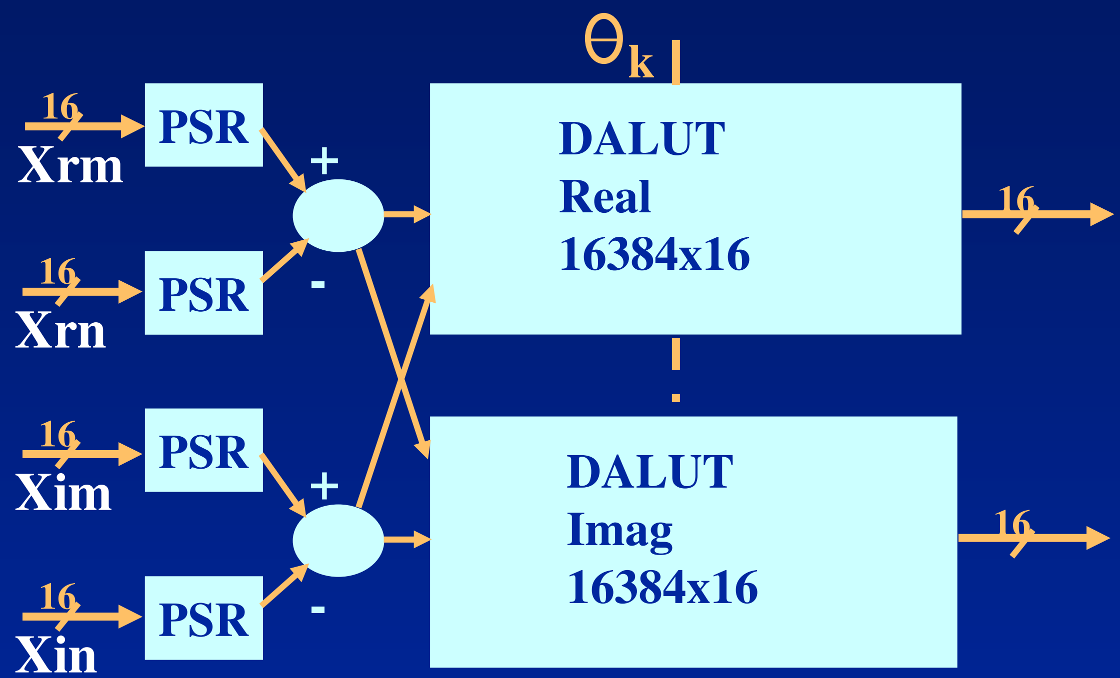
$$= \left(x_{Rm} - x_{Rn}\right) \times \cos\theta_k + \left(x_{\mathrm{Im}} - x_{In}\right) \times \sin\theta_k$$

$$+ j\left(x_{Rn} - x_{Rm}\right) \times \sin\theta_k + \left(x_{\mathrm{Im}} - x_{In}\right) \times \cos\theta_k$$

Radix-2 - using DA

- y_n implemented using distributed arithmetic
- DALUTs contain pre-computed sums of partial products
  - 3 variables: $\left(x_{Rm} - x_{Rn}\right), \left(x_{\mathrm{Im}} - x_{In}\right), \theta_k$
- $k = \log_2(N/2) \quad address\ space: 2^{(k+2)}$
- DALUT size increases exponentially
  - 8192-point FFT, 16 bit sine-cosine accuracy
  - will need 16384 deep, 16 bit wide DALUT

Radix-2 using DA



- N=8192, b=c=16, k=12
- Huge DALUT size

XILINX

Efficient DA implementation

- DALUT partitioning [Les Mintzer, ICSPAT'96]
  - uses the following:

$$\theta_k = \theta_1 + \theta_2 + \theta_3$$

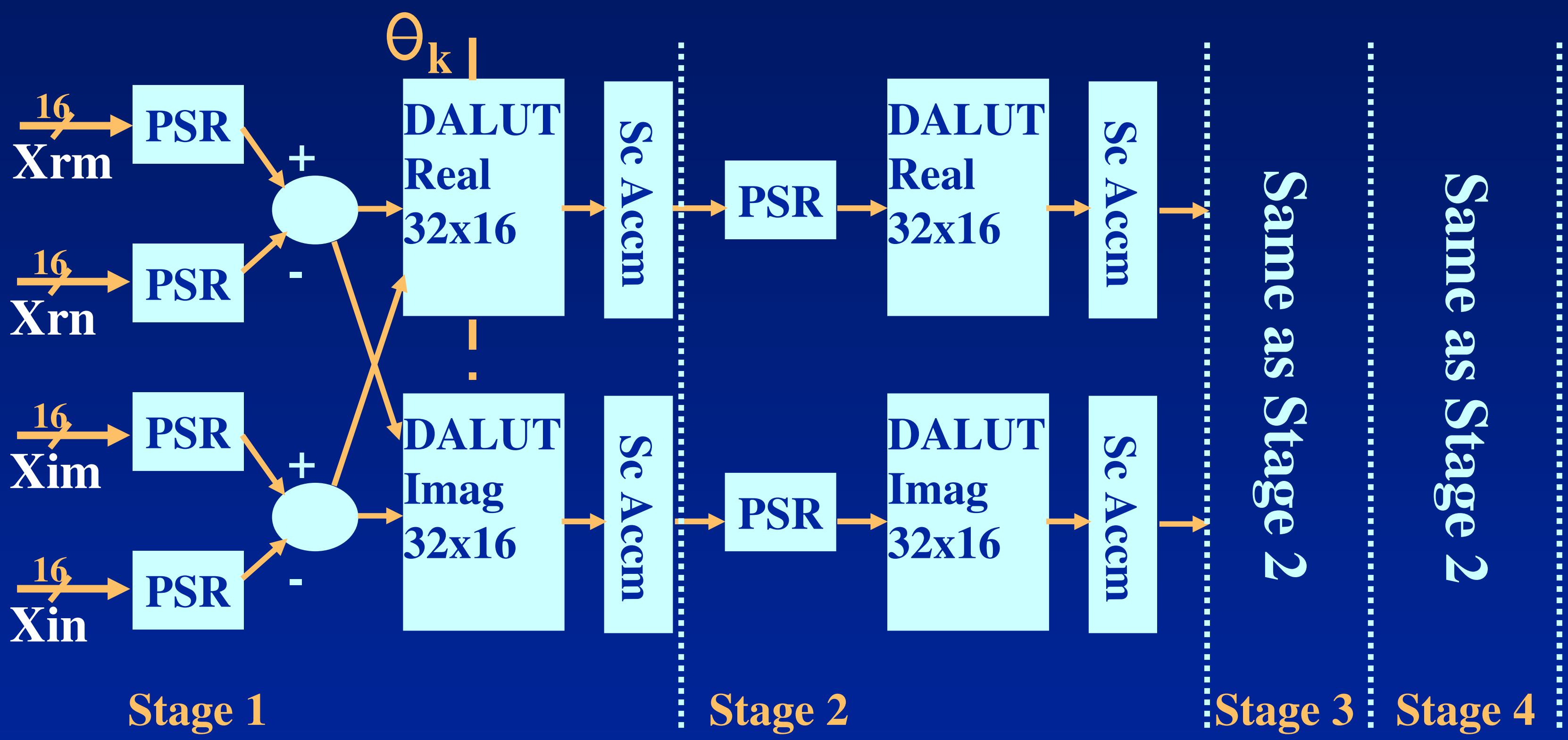
$$Ae^{\theta_k} = \left(\left(Ae^{\theta_1}\right)e^{\theta_2}\right)e^{\theta_3}$$

  - breaks $\theta_k = 110011100100$ as:

$$\theta_1 = 1100\ldots \quad \theta_2 = \ldots 1110\ldots \quad \theta_3 = \ldots 0100$$

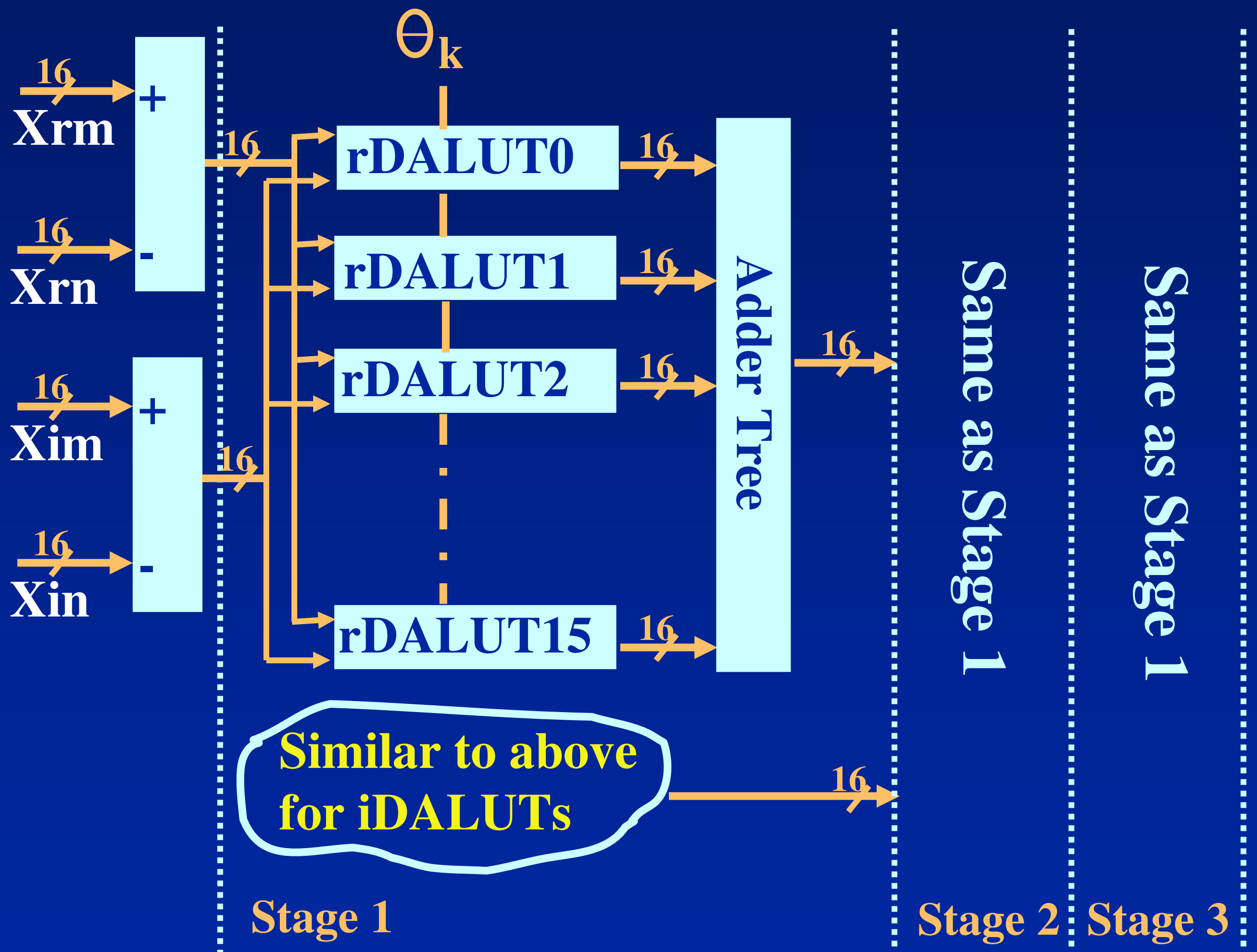
- reduces DALUT size from 16384 to 192
  - requires some additional logic

Radix2 using DALUT partitioning



- N=8192, b=c=16, k=12 (3+3+3+3)
- DALUT partitioning => Large area savings

Parallel implementation



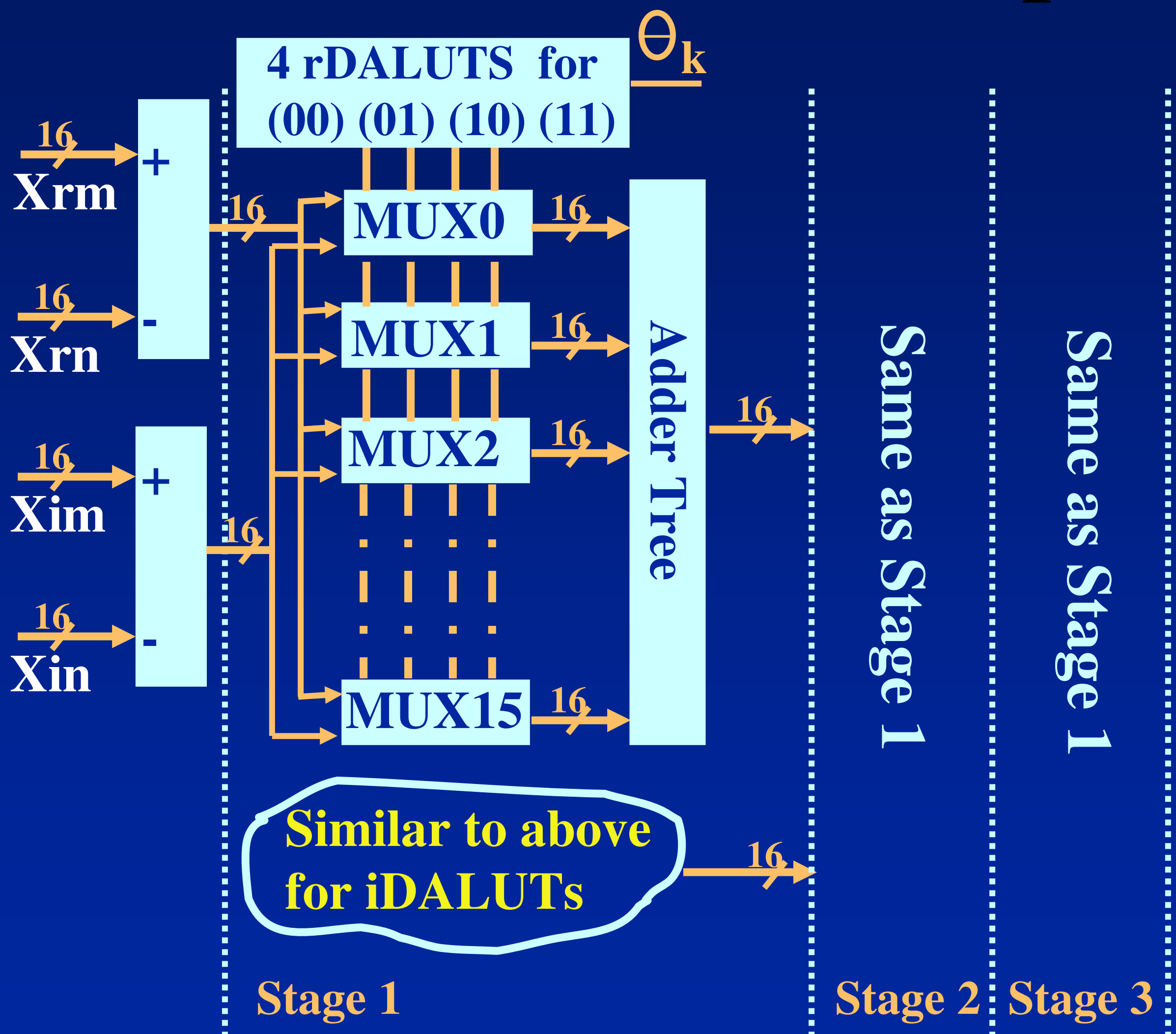
- b bit input
- b rDALUTs / stage
- bX speedup
- ~bX area

Redundancy in Parallel Implementation

- Observation
  - b DALUTs share same θ_k
  - b DALUTs can have 4 inputs: $\left(x_{Rm} - x_{Rn}, x_{\mathrm{Im}} - x_{In}\right)$
    - (0,0) (0,1) (1,0) (1,1)

- Improvement
  - can replace b DALUTs with 4 DALUTs and b Muxes
  - tremendous area saving with same speedup

Efficient Parallel Implementation



- b bit input
- 4 rDALUTs / stage
- b Muxes/stage
- bX speedup
- area incr << bX

Area Savings/Speedup

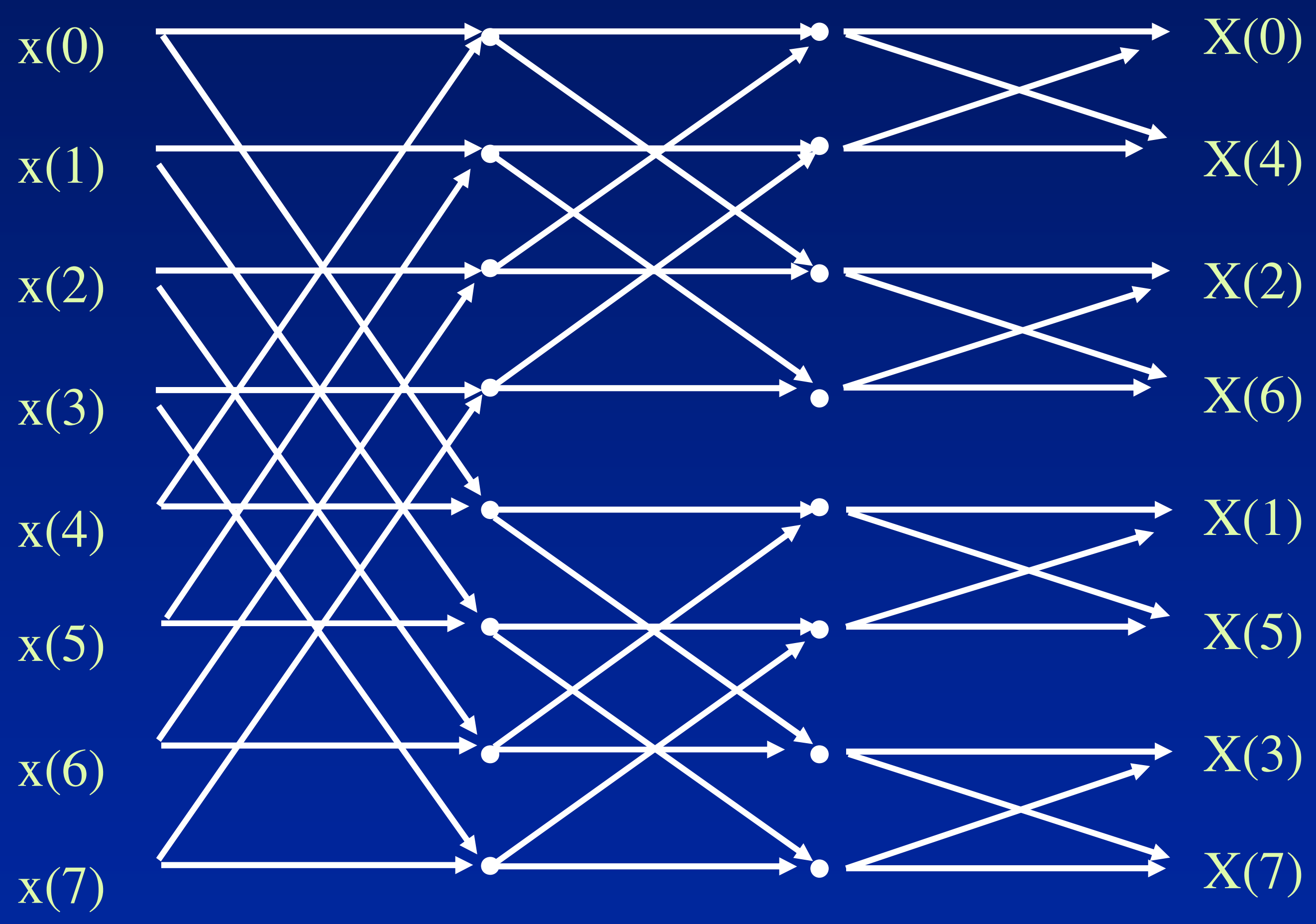
- Benefit of parallel implementation and using muxes
- Case A
  - N=8192 k=12 b=16 c=16

Impl type	CLBs reqd (Case A)		Speed (Case A)	
Serial	2ck/3 + c + 2b + 4	(180)	X	(X)
Simple Parallel	2bck/3 + (b-1)c + 2b	(2320)	bX	(16X)
Efficient Parallel	ck + bc + (b-1)c + 2b	(717)	bX	(16X)

FFT design with efficient radix-2s

- N=1024 k=8 b=c=16 => 5120 radix-2 operations
- Assuming 2 cycles/radix-2 operation => 10240 cycles
- How can we reduce # of cycles required?
  - Using more radix-2s alone will not help
  - Bottleneck is interaction with memory
  - Need to use memory-partitioning along with multiple radix-2s

Typical FFT structure



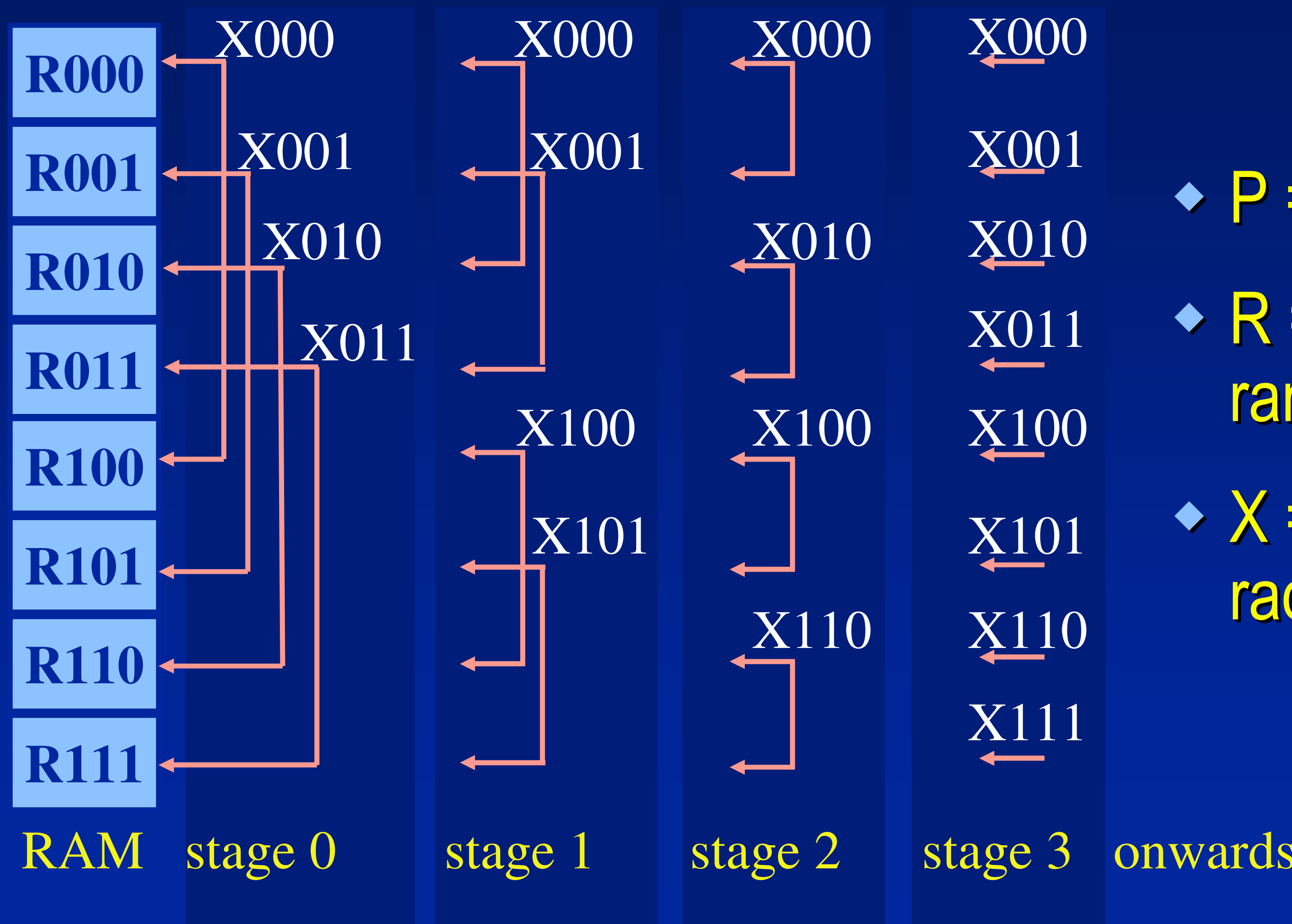
Memory partitioning

- Observation
  - radix-2 interacts with smaller memory part in advanced stages
  - Assume p memory-partitions
  - From after a critical stage, p radix-2s can interact independently with them in parallel
  - $p = 2^{cstage}$
  - Up to cstage, p/2 radix-2's can operate in parallel

Memory partitioning

- Partitioning Scheme
  - optimal # of partitions = # of radix-2s
  - Represent partitions as R_V and radix-2s as X_V
  - stage k (< cstage)
    - use radix-2s X_V such that kth bit of V is 0
    - R_V interacts with X_{V1} and X_{V2}
      - V1=V V2=V with kth bit reversed
  - stage k (>= cstage)
    - X_V interacts with R_V

Radix-2 Memory interaction



- P = 8
- R => ram-part
- X => radix-part

A design for maximal radix-2 utilization

- P=4
- Memory-partitions: R00 R01 R10 R11
- Radix-2 units: X00 X01 X10 X11

Stage 00	X00 <-> (R00, R10)	X01 <-> (R01, R11)		
Stage 01	X00 <-> (R00, R01)	X10 <-> (R10, R11)		
Stages 10/11	X00 <-> R00	X00 <-> R00	X00 <-> R00	X00 <-> R00

Area/speed effect with partitioning

- K=12 b=16 c=16

- N=8192

Radix-2 Units	CLBs	Cycles
1	723 (X)	106496 (F)
2	1446 (2X)	57344 (1.9F)
4	2892 (4X)	30720 (3.5F)
8	5784 (8X)	16384 (6.5F)

- N=1024

Radix-2 Units	CLBs	Cycles
1	640 (X)	10240 (F)
2	1280 (2X)	5632 (1.9F)
4	2560 (4X)	3072 (3.5F)
8	5120 (8X)	1664 (6.5F)

Conclusion

- New approach to efficiently design radix-2s for FPGAs
- Present a novel memory partitioning scheme for optimal usage of multiple radix-2s
- Provides a scheme to exploit area/speed tradeoff
- Method can be easily adapted for automatic FFT core generation based on user's requirement